TUGboat

Volume 33, Number 3 / 2012

General Delivery

LaTeX

Expanding Horizons

Philology

Fonts

Macros

Hints & Tricks

Book Reviews

Abstracts

Letters

News

Advertisements

TUG Business

TEX Users Group

TUGboat (ISSN 0896-3207) is published by the TEX Users Group.

Memberships and Subscriptions

2012 dues for individual members are as follows:

- Ordinary members: $95.
- Students/Seniors: $65.

The discounted rate of $65 is also available to citizens of countries with modest economies, as detailed on our web site.

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed, as well as software distributions and other benefits. 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.

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is $100 per year, including air mail delivery.

Institutional Membership

Institutional membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group, as well as providing a discounted group rate and other benefits. For further information, see `http://tug.org/instmem.html` or contact 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 trademarks which commonly appear in *TUGboat* should not be considered complete.

[printing date: November 2012]

Printed in U.S.A.

Board of Directors

See `http://tug.org/board.html` for a roster of all past and present board members, and other official positions.

Addresses

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

Telephone
+1 503 223-9994

Fax
+1 815 301-3568

Web
`http://tug.org/`
`http://tug.org/TUGboat/`

Electronic Mail
(Internet)

General correspondence, membership, subscriptions:
`office@tug.org`

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

Technical support for TEX users:
`support@tug.org`

Contact the Board of Directors:
`board@tug.org`

> [This] is what makes the history of technology interesting and relevant: it not only teaches us about the way things used to be done; it also gives us perspective on how things are done today — and how they most likely will be done in the future.
>
> Henry Petroski
> *The Book on the Bookshelf* (1999)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

Editor Barbara Beeton

Volume 33, Number 3 • 2012
Portland • Oregon • U.S.A.

TUGboat editorial information

This regular issue (Vol. 33, No. 3) is the last issue of the 2012 volume year.

TUGboat is distributed as a benefit of membership to all current TUG members. It is also available to non-members in printed form through the TUG store (`http://tug.org/store`), and online at the *TUGboat* web site, `http://tug.org/TUGboat`. Online publication to non-members is delayed up to one year after print publication, to give members the benefit of early access.

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

Submitting items for publication

The deadline for receipt of final papers for the next issue is March 1, and for the one after, July 1.

The *TUGboat* style files, for use with `plain` TeX and LaTeX, are available from CTAN and the *TUGboat* web site. We also accept submissions using ConTeXt. Deadlines, tips for authors, and other information:
`http://tug.org/TUGboat/location.html`

Suggestions and proposals for *TUGboat* articles are gratefully accepted. Please submit contributions by electronic mail to `TUGboat@tug.org`.

Effective with the 2005 volume year, submission of a new manuscript implies permission to publish the article, if accepted, on the *TUGboat* web site, as well as in print. Thus, the physical address you provide in the manuscript will also be available online. If you have any reservations about posting online, please notify the editors at the time of submission and we will be happy to make special arrangements.

TUGboat editorial board

Production team

TUGboat advertising

For advertising rates and information, including consultant listings, contact the TUG office, or see:
`http://tug.org/TUGboat/advertising.html`

TUG Institutional Members

American Mathematical Society,
Providence, *Rhode Island*

Aware Software, Inc.,
Midland Park, *New Jersey*

Center for Computing Sciences,
Bowie, *Maryland*

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

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

Masaryk University,
Faculty of Informatics,
Brno, *Czech Republic*

MOSEK ApS,
Copenhagen, *Denmark*

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

Springer-Verlag Heidelberg,
Heidelberg, *Germany*

StackExchange,
New York City, *New York*

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

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

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

Université Laval,
Ste-Foy, *Québec*, *Canada*

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

Ab Epistulis

Steve Peter

Since I last wrote at the beginning of the year, many things TEX have been transpiring, not the least of which (for the present group at least) was the annual meeting in Boston. I had an amazing time there, reconnecting with old friends, meeting new people, and having a great time talking TEX! (Have a look at `http://www.tug.org/2012` for a recap of what you might have missed.) Next year, the meeting will be in Tokyo, from October 23–26. The local organizing committee is hard at work making arrangements, and we will have more to say here and on the main TUG website and the dedicated `http://www.tug.org/2013` in the not-too-distant future (or RSN as they say in the software world).

Next year is also an election year for TUG. It will be a presidential election year, and several director's positions are also up for election. An official announcement appears elsewhere in this issue.

Another announcement from TUG is that we are now able to offer the Lucida fonts in OpenType format. The Type 1 Lucida fonts are frozen and will not be developed further, while the OpenType fonts continue to be actively maintained and developed. For instance, the OpenType fonts support more languages, with a variety of visual improvements to their glyphs and spacing. Lucida OpenType is also one of the few font sets currently supporting OpenType mathematical typesetting, and (uniquely, to our knowledge) provides a bold math variant. The regular-weight Lucida OpenType math font includes a newly-designed script alphabet design. Check out the TUG website to order.

The TEX Collection DVD for this year (which contains TEX Live, MacTEX, proTEXt, and a snapshot of CTAN) has shipped to TUG members since I last wrote. Nonmembers can also order the DVD from the TUG store at `http://tug.org/store`, and all the software may be freely downloaded (`http://tug.org/texlive`).

New, corrected printings of *Computers and Typesetting* and *Digital Typography* are available. Donald E. Knuth writes "…the books themselves are significantly better in hundreds of small ways. I went through every page very carefully and introduced many refinements, which have made me extremely happy with the result. I'm now able to replace my personal desk copies, in which hundreds of handwritten notes had been scrawled since the Millennium edition came out, by fresh versions that are essentially perfect (as far as I know). This is especially true of Volume B, because important updates to the TEX software that were made in 2002 and 2007 have never before been available in print."

The TUG website has a section (`http://www.tug.org/books`) that offers these and other books of either TEX or typography interest. A small portion of the sales benefits TUG so that we may continue to support various projects.

And speaking of books (I seem to write that a lot—maybe I should make a macro), CSLI Publications is now offering a 20% discount for TUG members on all their books, including the newly reissued (and corrected) *Digital Typography*. Such discounts are only one of the benefits available to you as a TUG member. Check out `http://tug.org/books/#discounts` for all the currently-available discounts.

And not to sound like a broken record (or a scratched CD) there are a number of worthy tasks looking for volunteers. Karl Berry maintains a list of projects in the TEX community looking for help at `http://www.tug.org/help`.

On a more somber note, the book world has lost two distinguished members in the last couple of months. Although neither was in the TEX world per se, their influence on the larger world of books merits their mention here.

Dan Carr was a punchcutter and printer operating in Ashuelot, New Hampshire, as the Golgonooza Letter Foundry & Press. Born in Rhode Island in 1951, he first came to my attention for his work on Greek typefaces, in particular Parmenides, but he cut the outstanding Regulus Latin typeface and, not being tied exclusively to the past, digitized his types as well. He is survived by his wife and business partner, Julia Ferrari.

Bill Hill was born in Scotland (hearing him speak, there was no doubt) and began in the traditional world of the newspaper industry. Upon seeing his first personal computer, he saw the future and joined Aldus to work on page layout before being hired by Microsoft in 1994. At Microsoft, he was one of the key figures behind ClearType and a strong proponent of onscreen reading. I knew Bill online through our various discussions of ebooks and typography. In fact, our last exchange was in the form of a (good-natured) argument over the meaning of margins in ebooks. Esoteric, to be sure, but each of us had deeply-felt opinions on the matter, and I will miss being able to bounce ideas off him. For a wonderful example of his worldview, see `http://youtube.com/watch?v=RP8D4uWEw5A`.

◇ Steve Peter
`president (at) tug dot org`
`http://tug.org/TUGboat/Pres`

Editorial comments

Barbara Beeton

New printings of *Computers & Typesetting*

Following on to Steve's comments, some additional information: The new printings of the *hardcover* editions of *C&T* volumes A (19th printing), B (9th printing), and C (8th printing) were released in April; new printings of volumes D and E are still "in press" but not yet released. The new printings were necessitated by the loss of the original films; work on the new versions has produced archival-quality PDF files that should withstand future reprintings without the danger of loss to which more tangible media are subject. The whole story is on Don's web page, `http://www-cs-faculty.stanford.edu/~knuth/abcde.html`, near the bottom under "Spiffy New Printings".

Since these are new *printings*, not new editions, it isn't clear from vendor listings (e.g., Amazon) whether what is being sold is the new printing or an old one. However, if you order through the member link on the TUG bookstore page, and specify that you want only the new printing but receive the wrong printing in error, a message to the publisher's customer service should result in a prompt correction.

The updated source files haven't been posted to CTAN yet, but we'll follow up to see if this can happen before TEX Live 2013 is produced.

Three printing-related symposia

Reading Digital was the topic of a symposium held in April at the Rochester Institute of Technology. Organized by Chuck Bigelow, his charge was stated thus: "I invited vision scientists to talk about scientific studies of reading in relation to digital screen displays." The presentations, while directed mainly at how to present material for best comprehension when read from a screen, also provided much food for thought regarding the possibilities for effective communication through all text-based media.

An overview and highlights of the symposium were posted by one of the speakers, Mario Garcia, on his blog: `http://garciamedia.com/blog/articles/the_rit_reading_digital_symposium_vision_studies_take_center_stage`.

Although the symposium was recorded, it is not yet clear if the videos will be posted for public access.

The symposium was held in conjunction with the presentation of the Frederic W. Goudy Award for Excellence in Typography. This year's recipient was Kris Holmes, co-creator of the Lucida family of typefaces. A reception and exhibit of Kris' work followed her lecture, "Moving Right Along". (The recipient of the first Goudy Award was Hermann Zapf; other recipients include Matthew Carter and Chuck Bigelow, names well known to TUG members.)

29th Isaiah Thomas Award On September 20, two anniversaries were celebrated — the 200th anniversary of the founding of the American Antiquarian Society, and the 75th anniversary of the RIT School of Printing (now the School of Media Sciences) — by presentation of 2012 Isaiah Thomas Award to the AAS. This award recognizes leaders in the newspaper industry, and earlier honorees represent a Who's Who of editors, Pulitzer Prize winners, and other practitioners. As noted by Dave Walden in his introduction at TUG 2012 to the printing and publishing world of Boston, Isaiah Thomas was the printer and publisher of *The Massachusetts Spy* (or the "Sedition factory", as it was also known to the British), and author of *The History of Printing in America*, first published in 1810; in 1812, Thomas founded the AAS in Worcester, where he had moved his press in order to take it out of range of the British fleet in Boston.

This year's award was both a celebration of the founder of the AAS and recognition of the Society's contribution to the preservation of historical documents and making them accessible in a digital age.

A panel discussion addressed the issue of preserving the history of news — the primary source from which scholars derive their material for study. The AAS is engaged in digitizing their holdings, but they also keep the originals. (At the Society's headquarters, after the celebration, the first number of the *Spy* printed in Worcester was on display, along with other materials published by Thomas and some of his tools of the trade. The paper on which this first edition is printed is in as good or better condition than newspapers printed in the middle of the last century, and eminently readable, although this reader's eyes did spot a typo on the front page.) Many newspapers that have "preserved" their archives on microfilm or other media more compact than paper, and then disposed of the paper copies, have discovered, to their distress, that the new images were unreadable. (This ignores the matter of different local editions, which often aren't saved at all; only the "city" edition usually warrants that consideration.) The obsolescence of each new medium is also a concern; who still has the equipment to read a 5-inch floppy disk?

A special treat was the appearance of Isaiah Thomas himself, portrayed by actor Neil Gustafson in appropriate 18th century garb. He gave a spirited first-person account of his life as a printer, publisher, and patriot. He is often a welcome visitor at local schools, where he helps make history come alive.

Coverage of the award ceremony can easily be found with the help of Google; the report by the Worcester *Telegram & Gazette* has a particularly fine photo of Thomas expounding on his life and times.

Oak Knoll Fest XVII The topic of this year's symposium was "The Fine Book in the Twenty-First Century — Yes, It Will Survive!". Books created in limited editions by private press printers, often using hand presses, are a special case. They are usually acquired by serious collectors and for library special collections. So the main question was not, is there enough interest in the physical books, but, will the tools and supplies continue to be available to the individuals who wish to create such books?

One of the speakers was Jerry Kelly, author of many books on typography (one on Zapf was reviewed in *TUGboat* **33**:2); his topic included the impact of technology on typography, one facet being the use of polymer plates created using computer type.

An open source font from Adobe: Source Sans Pro

Early in August, Adobe released their first open source font family, as reported by Paul Hunt in an Adobe type blog:

`http://blogs.adobe.com/typblography/2012/08/source-sans-pro.html`

This family of fonts is based on the gothic forms of designs by Morris Fuller Benton, in particular News Gothic and Franklin Gothic. The target applications include UI labels, where the need for legibility is paramount, and longer passages of text on screen and in print.

All the source files used in the production of Source Sans Pro are available "so that they can be referenced by others as a resource on how to build OpenType fonts with an AFDKO [Adobe Font Development Kit for OpenType] workflow". They have been released under the SIL Open Font License.

A followup blog post on September 24:

`http://blogs.adobe.com/typblography/2012/09/source-code-pro.html`

announced the release of a monospace font in the family — Source Code Pro.

All fonts and related software tools are available through the `Open@Adobe` portal on SourceForge. Details regarding design considerations and availability, as well as extensive discussion, can be had from the cited web pages.

LaTeX support for Source Sans Pro and Source Code Pro was posted to CTAN shortly before this issue went to press.

Errata: *TUGboat* 33:1 and 33:2

In the contents-by-difficulty listing on the inside back cover of **33**:1, the author of the "intermediate plus" article on page 54, "Generating barcodes with LuaTeX", was erroneously listed as Joseph Wright. The actual author is Patrick Gundlach. Apologies to both Patrick and Joseph. The on-line listing has been corrected.

The translation of Arno Trautman's article, "The `chickenize` package — fun with node manipulations in LuaTeX", alleged in the abstracts on page 121 of **33**:1, *Die TeXnische Komödie*, to be "published in this issue of *TUGboat*" wasn't. We hope to see it in the future.

In the proceedings issue for TUG 2012, **33**:2, the introductory article, which summarized all the rest of the program, failed to mention the presentation by Dave Walden, "My Boston: Some printing and publishing history".

Also in **33**:2, the article by Federico Garcia included a number of examples of musical notation generated with METAFONT. On page 161, in section 2.5, the triple-sharp key signature ♯♯♯ that should have appeared on every staff in the example and in text a few lines farther down was missing. It has now been restored to the on-line version of the article.

◇ Barbara Beeton
`http://tug.org/TUGboat`
`tugboat (at) tug dot org`

KOMA-Script comes of age

Markus Kohm

Abstract

In 1994 not only LaTeX 2ε but also KOMA-Script came into the world, thus having its 18th birthday this year. In contrast to LaTeX 2ε, which was completely developed at birth, KOMA-Script had to grow during the last 18 years. It started as a baby, not only complaining in each and every situation but getting on the nerves of its father and the environment. Thus an occasion for the father to write a short retrospective.

1 The conception

The exact date of the conception and the length of the pregnancy is hard to tell. More or less by chance I came across Frank Neukam's Script collection at

Editor's note: First published in *Die TeXnische Komödie* 3/2012, pp. 10–16; translation by Uwe Ziegenhagen.

the beginning of the 1990s. It contained essentially everything I needed for technical documents during my studies.

What I was missing were the adjustments needed for literary texts, for example support for DIN A5 format. Without further thought I added the most important changes into the style files. The licence of Script was somewhat unclear and the author hard to reach. Therefore I decided to publish my modified version via Maus KA[1] as an alternative.

Script was altered by others as well, so for a while there were various, incompatible versions floating around. When Neukam published Script 2.0 in December 1993 he rightly complained about the confusion. Script 2.0 contained all my changes.

Shortly afterward I heard for the first time that a successor of LaTeX 2.09 was on the way and that it would change everything. I started thinking about the future of my documents, which by that time completely relied on Script 2.0. Beginning in February 1994 I tried to get into contact with Frank Neukam to prevent a muddle of new versions. About that time — it is hard to tell when exactly — I managed to get a beta version of LaTeX 2$_\varepsilon$ on my computer.

Since I did not intend to poach again in somebody else's preserve — Neukam could not be reached either by mail or e-mail — I developed KOMA-Script 2.0 based on Script 2.0, the standard classes and my own ideas. The version number and the choice of the Grotesk font for the name were intended to create the link to Script 2.0. Primarily the whole thing was designed as a kind of "Script 2.0 for LaTeX 2$_\varepsilon$".

2 The birth

The baby was born on July 7th 1994 in the Maus KA. It consisted of just the three basic classes `scrbook`, `scrreprt`, `scrartcl`, the `typearea` package and a manual, which at that time was just a slightly modified version of the Script 2.0 manual.

Like most babies it could cry, demanded attention and was hungry. One could like it, get annoyed by it or just ignore it. It didn't play soccer, yet.

3 The suckling babe

Even in the early days of KOMA-Script the folks around hassled its father. Not only did everyone want to see the baby, they also wanted to touch it, some thought a finger was too small, others a toe too big, the eyes too blue, too green, too gray, they wanted to have a hat here and a jacket there. Daddy liked the attention and tried hard to please everybody. In the meantime the baby had to get patched up and its belly massaged to cover its deficiencies.

As early as October 1994 KOMA-Script switched to `docstrip`. This meant its own `dtx` files for the core as well as for the first additional packages.

4 The toddler

Since I rarely wrote business letters and preferred a fountain pen for my private communication I had put no energy into this area. This changed fast when Axel Kielhorn contributed a first adaptation of `script_l.sty` for LaTeX 2$_\varepsilon$ to KOMA-Script. Thus the first KOMA-Script letter class, `scrlettr`, was created.

After that was one critic demanding a proper manual. As a proud dad who was — besides his studies — quite busy with the KOMA-Script baby, some DVI-driver development for the Atari ST and the copublishing of a series of amateur anthologies I forbade those requests. So the LaTeX career of "Harald"[2] started with the creation of an updated manual for KOMA-Script.

Before its first birthday KOMA-Script became a mingle-mangle of classes, packages and example files. Especially varied were the packages for all the hats and shoes requested by people who had seen the baby crawling. I was not keen on creating new hats and shoes every week but `fancyheadings` — the only package in existence for this purpose — always caused trouble when interacting with my baby. So I decided to create `scrpage`. From then on all questions for new hats and shoes were answered.

Another big step consisted of some extensions proposed by Werner Lemberg. He dealt extensively with the so-called CJK languages.[3] To typeset these languages he needed a few extensions not available in the standard classes. In KOMA-Script they were implemented quickly but not fully sufficient. At this time it became clear that KOMA-Script had to become more international. Thanks to Werner Lemberg there was an abbreviated English translation of the German KOMA-Script manual.

5 The childhood

At the turn of the millennium KOMA-Script was not only again a collection of manifold classes and package but the rather monolithic structure increasingly complicated further development. Even when its dad

[1] Maus KA was a mailbox of MausNet, a mailbox network that was quite prominent in Germany, Austria and Switzerland when acoustic couplers and modems were state of the art. The name was derived from the local license plate; KA stood for Karlsruhe.

[2] Named Axel Sommerfeldt in the mundane world, known for his tremendous `caption` package and his help with many LaTeX issues. I got to know him as Harald.

[3] Chinese, Japanese, Korean

wanted to implement a minor change he had to make changes at three or four places in a file far away from each other. This was due to the source code being based on the logical processing of the class files: first the options, then the basic settings, building on the commands and environments. It was time for a more topic-based approach!

There were also — partly similar — issues with the manual. Basically it was still the same manual Axel Sommerfeldt had created years ago. It was not structured according to the classes and packages; rather, the topics came in an arbitrary sequence. Although mostly consistent they were hard to follow and understand. And although the manual had grown from 60 pages to almost 100 pages, most topics were handled briefly. While at the beginning only nerds like me had used LaTeX and technical, concise manuals were okay, the users of LaTeX now included even “ordinary” people. LaTeX had become a tool for everybody, and they wanted to play with my baby!

To cut a long story short it was time to teach the child some structure. Its sentences had to get connectives and were not to be thrown into the room unmotivated. The problem was: I didn’t have the time. I had a small (human) son now and shared work and school with my wife. Thus I looked for some help.

Help was found fast, or let’s say suspiciously fast. When it came to the distribution of work for a new version of the KOMA-Script manual, all help was suddenly gone. So I started on my own with the restructuring of the KOMA-Script source code. I had estimated it would take from mid-2001 to the end of 2003. By then I wanted to complete the new manual as well. When I pulled the KOMA-Script sources apart it became clear that the scrlettr package had flaws at every inch and corner. If the original design by Frank was to blame, Axel’s implementation or my extensions, no one knows. Probably it was some kind of “joint blame”.

To compensate for my frustration with the manual I created a completely new design of a letter class. I don’t remember how the contact to Torsten Krüger came about but he proved to be the best tester of scrlttr2 that I ever had. This positive experience gave me the kick to continue with the manual. This was additionally encouraged by Jens-Uwe Morawski who in a short time created various chapters of the new manual on the basis of explanations from the old one. He also created all the new examples for the scrpage2 package which was to replace the old scrpage.

6 Growing up

When KOMA-Script reached a two-digit age there was a new manual, a new package for headers and footers and a completely new letter class featuring a new user interface. However there was one thing missing: the new code base! What was planned originally for the end of 2003 had a significant delay. In the meantime I had become the head of development for a commercial company, had fathered a daughter and had quit my job to completely take care of my kids. All this had led to some turbulence affecting the development of KOMA-Script. Furthermore it wasn’t simple to work on two completely separate code trees.

But in 2006 it finally arrived! On July 5th 2006 KOMA-Script 2.95 was published, two days before its twelfth birthday and featuring the new code base!

After that the development gathered speed. On November 3rd 2008, with the publication of the 17th iteration of the KOMA-Script code, all changes I had planned were implemented, resulting in version number 3. All concepts I had tested with scrlttr2 and considered well done were not only implemented for all classes but also documented. This resulted in a newly restructured manual.

In the meantime there were 34 iterations of KOMA-Script on the basis of the new code. There were a few new packages, among them scrlfile, tocbasic and scrjura for special purposes. There is a printed, significantly extended manual in its 4th edition.[4] And there’s no end in sight!

7 Really grown up?

In July 2012 KOMA-Script had its 18th birthday. I asked myself if this makes it grown up and full-blown. It is surely grown up. But grown up does not mean either full-blown or finished. Nowadays nobody can stop learning only because we finished our years of apprenticeship. We continuously learn and evolve. So I see no end to the development of KOMA-Script. In fact I have more ideas in my drawer than I can ever implement. While in earlier times I was annoyed if someone had the same idea and finished a package faster than me, today I am happy since then I can do other things.

It may surprise some, but my intention was never to implement as many features as possible into KOMA-Script. I just wanted to simplify the

[4] Before the first official publication there was a preprint by DANTE for its members only. In cooperation with Christoph Kaeder and especially through his personal efforts the Lehmann’s edition of the manual was published. This book was also the beginning of the successful DANTE book series where mostly books from Herbert Voß have been published.

user's work. The work with KOMA-Script should force the user neither to go through meters of partly contradictory literature nor to dive into TeX internals to get the typography right. The user should also not be required to load potentially incompatible packages to make minor changes to the form of the document — an inevitable necessity with the standard classes. Typographically useful changes should be applicable with a minimum of work and a clear concept.

In the early days of KOMA-Script the resources of the TeX distributions, the computer itself or "best practices" often defined the limits. In the meantime new concepts, especially with regard to user interaction, were created and implemented by me on the long road to KOMA-Script 3. There is however still huge potential for further development.

Once upon a time I was a little pink baby with a squeaky voice myself. During puberty the voice went down and hair covered my face. From version 2 to version 3 KOMA-Script has undergone such a huge change as well. In the meantime my eyes have become worse, the hair thinner — and not only at places where it had annoyed me anyway. Maybe KOMA-Script will face the same destiny. But KOMA-Script does not need glasses. Maybe one can ignore the eyes and remove the hair permanently instead of always having to shave it. Maybe one can even renew or change huge parts of the body.

8 What else there is to say...

I started to work on KOMA-Script for two reasons: for my own purposes and because I wanted to give something back to the community that had brought me TeX. More or less overnight, however, the project gained an enormous level of attention. Faced with the decision between guarding my baby and ignoring all requests or responding to others' needs, I decided for the latter.

Still, not everything has found its way into KOMA-Script. Take for example an extended layout package: developed with much hard effort and then used only for a few weeks. In the end only a small portion of the original package found its way into KOMA-Script as scrpage2. Due to the complexity of the user interface the rest was thrown away.

Of course there were times during the last 18 years when I was close to quitting all work on KOMA-Script. Countless hours were spent supporting users, often a burden, partly harassment. People called me — me, the notorious phone hater! — at half past ten in the evening to discuss a LaTeX issue they had, unrelated to KOMA-Script. One evening a professor rang my doorbell and offered me a job. It embarrassed me then — today it rather amuses me. I'm even wondering that no one has yet banged on my door at four in the morning to get help, but who knows what the future may bring...

On the other hand I was glad to be able to help people. On various occasions this support inspired me and helped expand my horizons. Sometimes I had to remind myself that I dealt with strangers, to align my own priorities.

I would like to have avoided the whole infamous discussion about license breaches. In the toddler times of KOMA-Script I really could have used some help with these things. Even today I think it is a big mistake to annoy developers with the peddling and enforcement of their wishes and claims.

But when I look back I don't just see a mountain of code — several tens of thousands of lines now and a growing book — but also many nice encounters with people:

There was Luzia, who played an important role in the publication of early versions of KOMA-Script; Bernd and David who gave advice on the way TeX works; Ulrike[5] who is not tired of explaining to people the logic behind KOMA-Script and who still sends me bugs; Robin, who has provided my domain for years and had the idea for the KOMA-Script documentation project; Rainer, Alexander, Heiner and many more, who provided me with wine, hardware and consoling words; Rolf, who with his calming influence nevertheless has the ability to stimulate; all people who drove me, cheered me up, calmed me down, inspired me; the Wise Guys[6] who helped me to cope at various times of frustration, although they don't even know me and therefore have a pretty one-way nature of communication with me.

People from DANTE have consistently supported me. It is partly this support that has kept me from throwing everything down.

The most important thing was all these years, that I met people I could call friends. People that I consider as valuable to me, although I hardly have a chance to put the level of attention to these friendships that they deserve.

Just one more thing: KOMA-Script still can't play soccer. As ever, there is something to be done!

◇ Markus Kohm
Freiherr-von-Drais-Straße 66
68535 Edingen-Neckarhausen
Germany
komascript (at) gmx dot info
http://www.komascript.de

[5] I would like to use this occasion to congratulate her on receiving the DANTE Award 2012!

[6] http://www.wiseguys.de

Almost 30 years of using TeX

Christina Thiele

Abstract

It's not just TeX that's gotten older and more seasoned ... Reflections on changes in TeX and friends as used in a small typesetting company: software and hardware, of course, but also procedures and skills, resources that went from zero to virtually infinite, all of it interwoven with life and personal change. It's not earth-shaking news, but we've come far enough that looking back yields some interesting comparisons.

1 How it all began

I first came across TeX at Carleton University in Ottawa in 1983. I had returned to the university the year before, to undertake graduate work in linguistics, with a monthly $300 (CDN) grant, which didn't even cover rent. So, a job was needed and here was one that just fell into my lap: working for Bill Cowan, one of the department professors and his new editorial project, the *Canadian Journal of Linguistics*.

Since 1975, Bill Cowan had been editor of the long-standing *Papers of the Algonquian Conference*, and in fact remained its editor until 1993 (the 25th conference). His colleague, Jean-Pierre Paillet, a specialist in Inuktitut, was keen on computers and had convinced Bill that a really great program had just been installed at Carleton, and they were looking for a suitable project to use it on. So, Bill took the plunge, reassured that J-P would be around whenever we had problems.[1]

The first publication to give TeX and the AM fonts a trial run before committing the *Canadian Journal of Linguistics* to its use was the 1983 edition of the 14th *Papers*.[2] With this book of over 400 pages behind us, we were ready to tackle *CJL* the following year.[3]

CJL's pages are, in fact, a record of TeX's own evolution, from the Almost Modern AM fonts and in-house phonetic bitmaps, proofing to paper in the absence of screen previewers, mainframes and balky text editors, (snail-)mailing proofs and then camera copy, to today's plethora of fonts, emailing of PDF files, and CTAN as the motherlode of macros for almost anything conceivable.

2 Hardware/software journeys

Our overall plan was to get dot-matrix proofs out to authors early in the year, so that editorial work could progress in parallel with development of the TeX elements: macros and a phonetic font. By the time the editorial/proofing cycle was done, we would have a functioning TeX process, access to a 240 dpi laser printer, and have copies in time for the October annual meeting. Figure 1 shows the same page, from both processes.

2.1 Gutenberg-to-TeX, on a Peach

In 1983, TeX was just at v0.93 on the Carleton mainframe (a Honeywell, running CP-6).[4] We used a locally made Apple clone (called a 'Peach'![5]) to do the data entry; authors in the humanities weren't using computers much — and it never occurred to us that those who did might send us a disc. So everything was input from scratch.

We used a software program written by a fellow down in Kingston, Ont., called Gutenberg (I kid you not!).[6] And Gutenberg's great feature was that it used tags rather than function keys to format text — just like TeX! Another great feature was that you could design additional characters — so that took care of all our phonetic symbols.

We'd print those files on a dot matrix printer as proofs for authors, and input the changes into the same file (sub-figure 1a). Still no sign of TeX code at this point. Once the editorial work was done, it was time to prepare the files for transfer to the mainframe, where they'd be run through TeX. Our trusty J-P wrote up a conversion program (in LISP, I think it was) which did a pretty good job of translating Gutenberg's codes into basic TeX commands. Then I'd upload the files via modem and a program

Editor's note: From a presentation at TUG 2012.

[1] Much of this is described in a memorial piece I wrote after Bill's death (Thiele 2001). Where the stories diverge, I would be inclined to go with the 2001 text, as it was closer to the events.

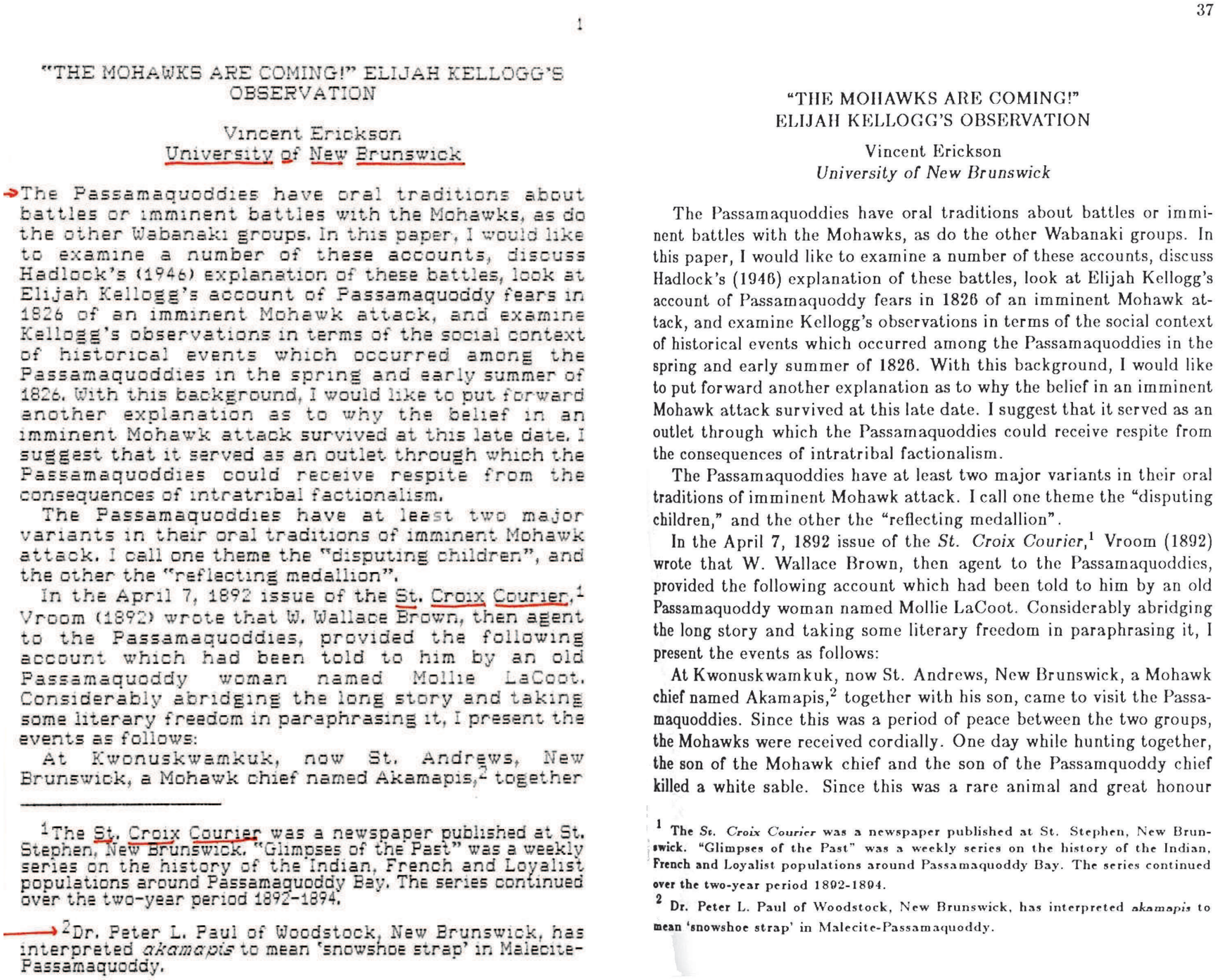
[2] Bill Cowan was editor of the *Papers* from 1975 till 1993; vols. 14 through 25 were done with TeX, and then LaTeX, at Carleton. Before TeX, these annual proceedings had been typescripts on 8.5 × 11 paper, which were then photo-reduced to the final trim size of 6 × 9.

[3] As with the *Papers*, *CJL* was also prepared with TeX, then LaTeX. The 1984 volume had only two issues, with roughly 230 pages. The next year, with the same budget, *CJL* became a quarterly, averaging well over 400 pages a year until 2006, when it changed to three yearly issues still averaging some 400 pages. Through four changes in editors, I continue to typeset *CJL* to this day.

[4] I've not been able to locate a `.log` file from that 'era'; however, I did find a printout of one from 1988, on CP-6: `This is TeX, DPS8/CP-6 Version 2.0 (preloaded format=lplain 87.6.25).`

[5] Wikipedia has a list of Apple `][` clones, and many are indeed 'fruity': Citron II, Golden, Orange, Pearcom, Pineapple 6502, ...

[6] I did a google search on this and, incredibly, came up with a review from 1983 (Glenn 1983), which included the name of the program's author: John Wagner.



1

"THE MOHAWKS ARE COMING!" ELIJAH KELLOGG'S OBSERVATION

Vincent Erickson
University of New Brunswick

The Passamaquoddies have oral traditions about battles or imminent battles with the Mohawks, as do the other Wabanaki groups. In this paper, I would like to examine a number of these accounts, discuss Hadlock's (1946) explanation of these battles, look at Elijah Kellogg's account of Passamaquoddy fears in 1826 of an imminent Mohawk attack, and examine Kellogg's observations in terms of the social context of historical events which occurred among the Passamaquoddies in the spring and early summer of 1826. With this background, I would like to put forward another explanation as to why the belief in an imminent Mohawk attack survived at this late date. I suggest that it served as an outlet through which the Passamaquoddies could receive respite from the consequences of intratribal factionalism.

The Passamaquoddies have at least two major variants in their oral traditions of imminent Mohawk attack. I call one theme the "disputing children", and the other the "reflecting medallion".

In the April 7, 1892 issue of the St. Croix Courier,[1] Vroom (1892) wrote that W. Wallace Brown, then agent to the Passamaquoddies, provided the following account which had been told to him by an old Passamaquoddy woman named Mollie LaCoot. Considerably abridging the long story and taking some literary freedom in paraphrasing it, I present the events as follows:

At Kwonuskwamkuk, now St. Andrews, New Brunswick, a Mohawk chief named Akamapis,[2] together

[1] The St. Croix Courier was a newspaper published at St. Stephen, New Brunswick. "Glimpses of the Past" was a weekly series on the history of the Indian, French and Loyalist populations around Passamaquoddy Bay. The series continued over the two-year period 1892–1894.

[2] Dr. Peter L. Paul of Woodstock, New Brunswick, has interpreted *akamapis* to mean 'snowshoe strap' in Malecite-Passamaquoddy.

(a) Gutenberg + dot-matrix printer

37

"THE MOHAWKS ARE COMING!"
ELIJAH KELLOGG'S OBSERVATION

Vincent Erickson
University of New Brunswick

The Passamaquoddies have oral traditions about battles or imminent battles with the Mohawks, as do the other Wabanaki groups. In this paper, I would like to examine a number of these accounts, discuss Hadlock's (1946) explanation of these battles, look at Elijah Kellogg's account of Passamaquoddy fears in 1826 of an imminent Mohawk attack, and examine Kellogg's observations in terms of the social context of historical events which occurred among the Passamaquoddies in the spring and early summer of 1826. With this background, I would like to put forward another explanation as to why the belief in an imminent Mohawk attack survived at this late date. I suggest that it served as an outlet through which the Passamaquoddies could receive respite from the consequences of intratribal factionalism.

The Passamaquoddies have at least two major variants in their oral traditions of imminent Mohawk attack. I call one theme the "disputing children," and the other the "reflecting medallion".

In the April 7, 1892 issue of the *St. Croix Courier*,[1] Vroom (1892) wrote that W. Wallace Brown, then agent to the Passamaquoddies, provided the following account which had been told to him by an old Passamaquoddy woman named Mollie LaCoot. Considerably abridging the long story and taking some literary freedom in paraphrasing it, I present the events as follows:

At Kwonuskwamkuk, now St. Andrews, New Brunswick, a Mohawk chief named Akamapis,[2] together with his son, came to visit the Passamaquoddies. Since this was a period of peace between the two groups, the Mohawks were received cordially. One day while hunting together, the son of the Mohawk chief and the son of the Passamquoddy chief killed a white sable. Since this was a rare animal and great honour

[1] The *St. Croix Courier* was a newspaper published at St. Stephen, New Brunswick. "Glimpses of the Past" was a weekly series on the history of the Indian, French and Loyalist populations around Passamaquoddy Bay. The series continued over the two-year period 1892–1894.

[2] Dr. Peter L. Paul of Woodstock, New Brunswick, has interpreted *akamapis* to mean 'snowshoe strap' in Malecite-Passamaquoddy.

(b) TEX + laser printer (reduced from 120%)

Figure 1: Comparison of outputs from Erickson (1983)

called Kermit. Transfer rate was 300 baud, from my apartment — one article would take about one supper (from prep to washing up) to complete.

Once on the mainframe, I'd run the file through TEX, ignoring all the screen-displayed log info, forcing TEX to eat through to the end of the file or, more commonly, until TEX threw up its hands and cried, "`TeX capacity exceeded`".

Editing on CP-6 was a bit of a trick. The `EDIT` program was a line editor with a maximum of 255 characters per line (Figure 2). Deleting text on a line was no problem but adding material (changes to the text or additional TEX code) meant that a line would easily reach its maximum capacity. So you had to insert a new line number (up to 999 'lines' between any two whole-numbered lines; in the sample shown, l. 1.010), then keyboard the new material. Copying the file over itself would re-write the line numbers.

We worked like this for several years (until 1991, when I got my very own UNIX box!). All in all, very labour-intensive, quite unsophisticated at times, and compared with today's workflow (the term didn't even exist in those days!), extremely inefficient. But we managed, like most everyone else who was trying hard to use TEX, especially to do more than just math and science.

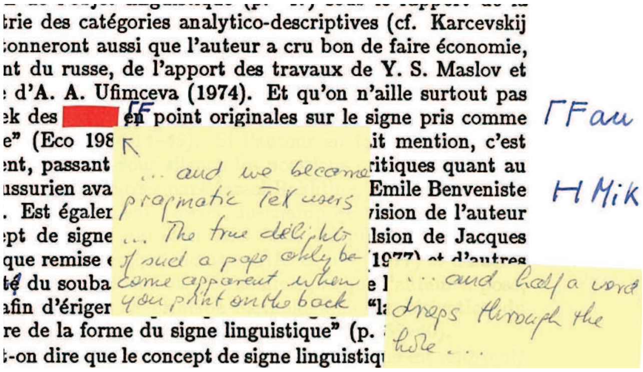
There was no screen-preview at the time so we proofed to paper — which cost us 10 cents per sheet. We had *many* defective pages of output, which we kept in a growing stack by the filing cabinet because they were the source of replacement letters and words for last-minute discoveries in what we thought was our perfect printed camera copy: the necessary 'repair' letters or words would be carefully

```
! EDIT
CP6 EDIT B00 HERE
* TY
1.000 The first publication to give \TeX\
2.000 before committing the
3.000 \textit{Canadian Journal of
4.000 Linguistics} to its use was the 1983
5.000 edition of the 14th \textit{Papers}.

* IN 1.010
1.010 and the \acro{AM} fonts a trial run
1.020 <return>

*TY
1.000 The first publication to give \TeX\
1.010 and the \acro{AM} fonts a trial run
2.000 before committing the
3.000 \textit{Canadian Journal of
4.000 Linguistics} to its use was the 1983
5.000 edition of the 14th \textit{Papers}.
```

Figure 2: EDIT session on CP-6



trie des catégories analytico-descriptives (cf. Karcevskij
:onneront aussi que l'auteur a cru bon de faire économie,
nt du russe, de l'apport des travaux de Y. S. Maslov et
: d'A. A. Ufimceva (1974). Et qu'on n'aille surtout pas
ek des en point originales sur le signe pris comme
e" (Eco 198 ... it mention, c'est
ent, passant ... ritiques quant au
ussurien ava ... Emile Benveniste
. Est égaler ... vision de l'auteur
ept de signe ... lsion de Jacques
que remise ... 1977) et d'autres
té du souba ... e l
afin d'ériger ... "l
re de la forme du signe linguistique" (p.
t-on dire que le concept de signe linguistiq

ΓF

ΓFau

HMik

... and we become pragmatic TeX users ... The true delights of such a page only become apparent when you print on the back

... and half a word drops through the hole ...

Figure 3: '... and we become pragmatic TEX users ... The true delights of such a page only become apparent when you print on the back ... and half a word drops through the hole ...'

excised with an Exacto knife and glued over the offending error on the camera copy (Figure 3; see also Thiele n.d., 2007).

Originally, all output was produced at 120%, then photo-reduced to the desired text size, to yield a crisper image (sub-figure 1b).[7] When the CM fonts arrived, we changed that, producing the journal text at the final design size — no more photo-reducing. Saved a step at the printers', which meant it also saved us a bit of money.

When we had tree diagrams, we worked out a two-stage process. I had figured out that a table structure was a good way to set the nodes and labels. Then Bill would carefully apply LetraLine in solid, dashed, or dotted versions, to the final paper copy to finish off the trees. The irony of using a tried-and-true traditional solution on pages produced with the very latest and hottest typesetting program around was not apparent to us for quite some time ...

Following the 1986 TUG meeting in Seattle, Dean Guenther introduced me to Janene Winter — and so began my involvement with developing the WSUIPA suite of fonts. At about the same time, a terrific phonetics book by Pullum and Ladusaw (1986) had been published, which was all about how to properly draw IPA symbols ... and each one was shown with horizontal rules for baselines, ascender and descender lines, and best of all, the equivalent of the x-height. Perfect descriptions to feed into Metafont, which she did — tirelessly, endlessly, repeatedly ... I proposed we borrow the names already in use in the Pullum and Ladusaw for macro names, and other than a hasty renaming of `\stop` to `\glotstop`, those remain in effect. From 1988 until 2005, the WSUIPA fonts were used. By then, Fukui Rei's TIPA/XIPA fonts had been in distribution for a few years, and I finally switched everything over to his, and they have become, I believe, the standard choice for most people doing linguistics with TEX.[8]

2.2 Building up the programs

2.2.1 `manmac` into `cjlmac`

J-P modified `manmac` to gradually arrive at *CJL*'s style and a psychology student (Mike Dunleavy) began to write documentation,[9] as J-P kept devising more macros to do what Bill wanted. As with all programming, initially the macros did only what Jean-Pierre's code told them to, with results that occasionally could be likened to a Venn diagram: there was an intersection area where what we wanted was what we got, but there seemed an awful lot of 'what we wanted' which was not 'what we got' outside of that area (Figure 4).[10]

2.2.2 Creating fonts

At the same time that `manmac` was being reshaped

[7] I believe subsequent testing showed there wasn't all that much of a gain but it was a working rule of thumb at the time.

[8] I also ran across some mail and files from Johannes Braams, where he'd sent me a sample of a phonetic font he'd devised (also from 2005) but I don't know how widespread its use became before TIPA.

[9] Actually, Mike began writing macros as well, and was the designer of the original `\diatop` macro, for multiple diacritics on a single character.

[10] This reminds me of something from the back cover of a book by Sebastian Rahtz (1987) and credited to Brian Kernighan: WYSIAYG, 'What You See Is All You Get' ... although perhaps one could adjust this one to WYCIAYG, 'What You Code Is All You Get' ...

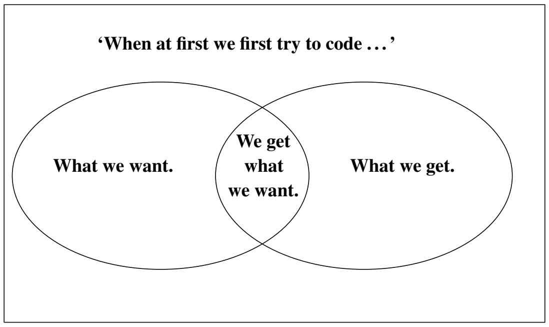


Figure 4: 'But that's not what I wanted!'

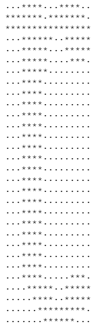
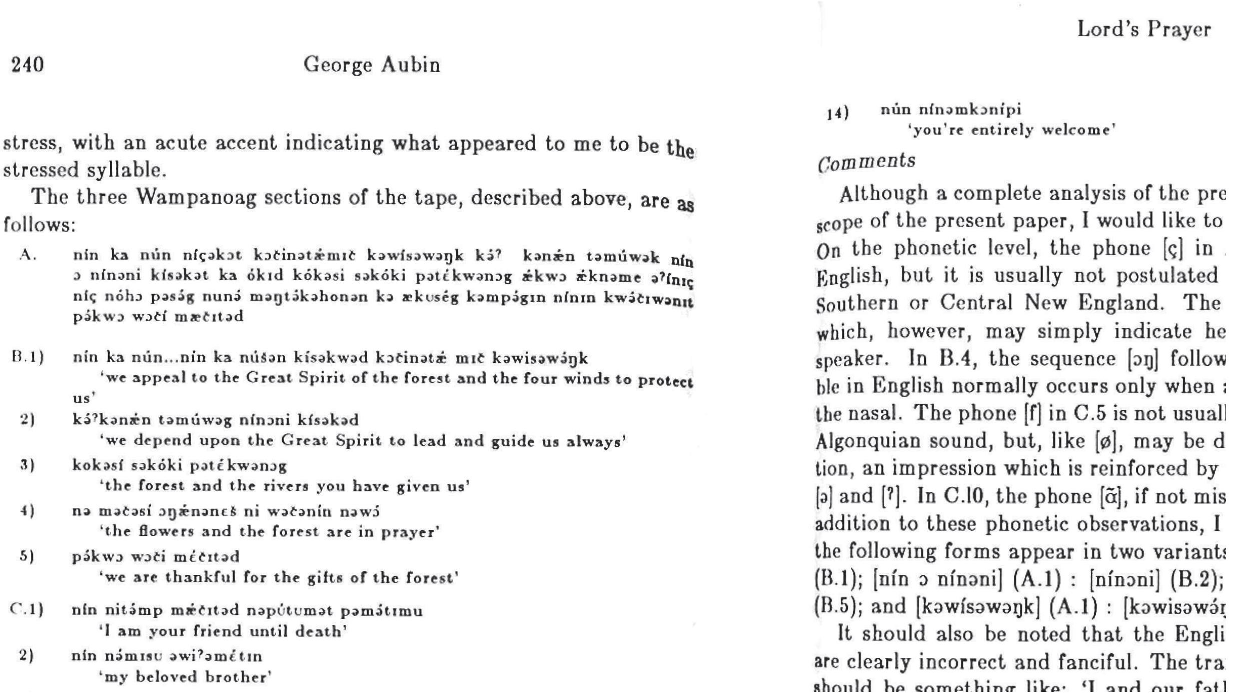


Figure 5: Verbatim copy of bitmap for `\rtail`



240 George Aubin

stress, with an acute accent indicating what appeared to me to be the stressed syllable.

The three Wampanoag sections of the tape, described above, are as follows:

A. nín ka nún níçəkɔt kɔčinətǽmıč kəwísəwəŋk kə́ʔ kənǽn təmúwək nín ɔ nínəni kísəkət ka ókıd kókəsi səkóki pətɛ́kwənɔg ǽkwɔ ǽknəme əʔínıç níç nóhɔ pəsə́g nunə́ məŋtə́kəhonən kə ækuség kəmpə́gın nínın kwə́čıwənıt pə́kwɔ wɔčí mæčıtəd

B.1) nín ka nún...nín ka núšən kísəkwəd kɔčinətǽ mıč kəwisəwə́ŋk
'we appeal to the Great Spirit of the forest and the four winds to protect us'

2) kə́ʔkənǽn təmúwəg nínɔni kísəkəd
'we depend upon the Great Spirit to lead and guide us always'

3) kokəsí səkóki pətɛ́kwənɔg
'the forest and the rivers you have given us'

4) nə məčəsí ɔŋǽnənɛš ni wəčənín nəwə́
'the flowers and the forest are in prayer'

5) pə́kwɔ wɔči mɛ́čıtəd
'we are thankful for the gifts of the forest'

C.1) nín nitə́mp mǽčıtəd nəpʊ́tumət pəmə́tımu
'I am your friend until death'

2) nín nə́mısʊ əwiʔəmɛ́tın
'my beloved brother'

Lord's Prayer

14) nún nínəmkɔnípi
'you're entirely welcome'

Comments

Although a complete analysis of the pre
scope of the present paper, I would like to
On the phonetic level, the phone [ç] in
English, but it is usually not postulated
Southern or Central New England. The
which, however, may simply indicate he
speaker. In B.4, the sequence [ɔŋ] follow
ble in English normally occurs only when
the nasal. The phone [f] in C.5 is not usuall
Algonquian sound, but, like [ø], may be d
tion, an impression which is reinforced by
[ə] and [ʔ]. In C.10, the phone [ɑ̃], if not mis
addition to these phonetic observations, I
the following forms appear in two variants
(B.1); [nín ɔ nínəni] (A.1) : [nínɔni] (B.2);
(B.5); and [kəwísəwəŋk] (A.1) : [kəwisəwə́
It should also be noted that the Engli
are clearly incorrect and fanciful. The tra
should be something like: 'I and our fat

Figure 6: Sample with `ph10` and `ph7` bitmapped fonts (Source: Aubin 1983, pp. 240–241)

into `cjlmac`, the design of a phonetic font — an IPA font — was also progressing, in bitmap form (Figure 5). J-P worked best at night; Bill best in the morning — almost every day began with newsprint fanfold printouts, one shape per sheet;[11] Bill would mark them for reworking, leave them for J-P, and new versions would appear the next morning. And so it went. We had `ph10` for regular text and `ph7` for the small text in footnotes, quotes, and the bibliography (Figure 6) — but neither had a matching italics.[12] So the upshot was that we tried to avoid having IPA symbols in titles. And of course, once the final pages were photo-reduced, the 'small' `ph7` font was painfully tiny, as shown in Figures 1 (footnotes) and 6 (examples with IPA).

2.3 Leaving our low-tech, low-TEX cocoon

We eventually left the Apple clone and Gutenberg, and moved on to the university's in-house DOS computer, called a Raven, where I learned to use this ASCII-based editor called Emacs. I have never changed — and brief bouts with WordPerfect and Word have never convinced me I was wrong. By then, our macros had been settled upon, we had

[11] Dean Guenther's article from the 1990 Cork meeting includes one of these, on p. 156. A complete chart of `ph10` can be found on p. 155 of Dean's article, as well as in Thiele (1987, p. 8).

[12] Normally, not an issue, as phonetic transcriptions are usually only in roman; on the other hand, if they're in an italicised title . . . — and it so happened that the *CJL* titleblock style put the title into italics. But that didn't much matter, as we didn't have a 'ph17' bitmapped phonetic font anyways! (At least, I don't remember that we did.)

our documentation, and we were inserting TeX into author files directly, leaving our crutches (Gutenberg, with J-P's conversion programs) behind.

We had also begun to receive submissions on disc (Thiele 1987, p. 24), and I would often wrestle with files, trying to get them up to the mainframe, hopefully losing all their hidden word processing codes, and then downloading essentially an ASCII file, which we'd code from scratch. But we were capturing author keystrokes, and on balance, came out ahead, in terms of accuracy and turn-around times. So that was good.

As well, I'd begun to finally read the `.log` info on-screen to try and catch obvious typos (`\itme` and `\exmaple` were two of my better ones), `hboxes` that were overfull by 213pt at times — and the more challenging searches for missing braces. At no time did we really consider moving to PCs, even though μTeX, PCTeX, and TurboTeX, to mention a few, had been coming out. We were on the mainframe from 1983 until 1991, in fact.

At some point, we decided we needed to have a laser printer of our own, in our building — did I mention that the dime-a-page charge was the prize/price we paid after hiking over to the other side of campus, to the computing services department on the 4th floor of the administration building?! A group of journal editors bought a 240 dpi Imagen printer by Ahearn and Soper (I'm amazed I can come up with that name so quickly!), with a driver devised by Rick Mallett, who also happened to be the TUG site coordinator for CP-6 (Mallett 1974).

3 There are journals everywhere . . . !

Other journal editors were becoming interested in how *CJL* was being produced. However, the queries would quickly lead to 'And you'll do the typesetting, right?', as opposed to 'And you can teach me how to do this?'. In the end, we actually did both.

As our work became known around campus, another editor (Toni Miller) and I realised there was an awful lot of scholarly editing going on, mainly but not exclusively in the humanities. We approached the library and asked if we could put together a display of scholarly journals. We devised a questionnaire, I typeset the results, and we mounted each journal and its description on large panels. In all, we had found over 50 publications whose editorial work was being done on campus. It was that display in early 1987 which also brought home the potential for typesetting on a much larger scale than we'd been planning.

Around 1988, with the urging of senior staff in the Computing Services division, I submitted a proposal to the university, suggesting that a centralised production centre would be a good way to consolidate the many disparate typesetting jobs (books and journals) which were humming along in relative obscurity across campus. Maybe make some money, raise the university's publishing/publications profile, as well as support its academic staff's initiatives.

THE SECRET LIFE OF AN EDITOR:
Scholarly Journals at Carleton University
REJECTED
ACCEPTED
An Exhibition presented by:
The Editors' Table and Carleton University Library
January 6 - 30, 1987
Main Floor - MacOdrum Library
Illustration by: GEORGE CHOUCHANI

Figure 7: The Secret Life of an Editor (drawn by George Chouchani)

3.1 The Journal Production Centre

The Journal Production Centre (JPC) existed from 1988 till 1991, and at times I had two employees, who mainly worked on keyboarding, coding, and initial processing, while I did debugging and final production runs. There were probably 6–8 journals which we worked on regularly, as well as a number of annual publications and one-off books.

We produced a lot of journal issues and books, but weren't making money to cover our expenses because I had no idea how to properly charge for our services. We were asking $3 and $4 per typeset page and it should have been more like $10 or $12, given the hours we were putting into them. Quite inefficient — and eventually the university grafted us onto Carleton University Press. That lasted about a year or so and we produced at least half a dozen books for them; eventually, though, we were cut loose

Table 1: The home network

Year	Machine	Speed	OS	Memory	Drive(s)
1991	IPC*	12.5 MHz	SunOS 4.1.3	8 MB	105 MB SCSI
2002	Sparc10*	2 × 40 MHz	Solaris 8	2 × 128 MB	105 MB SCSI
	Ultra1	170 MHz	Solaris 9	448 MB	4 GB SCSI
	Ultra10	440 MHz	Solaris 9	512 MB	9 GB SCSI
2008	Sunblade 1000	2 × 900 MHz	Solaris 10	4 GB	2 × 73 GB Fiber Channel
2009	Sunblade 2000	2 × 1.2 GHz	Solaris 10	4 GB	2 × 73 GB Fiber Channel
2012	Intel architecture†	2 × 3 GHz	Solaris 10	8 GB	2 × 80 GB SATA

* The IPCs and Sparc10 were 32-bit machines; all the rest have been 64-bit.

† The latest upgrade makes it possible to create virtual PCs running Windows 7. A cheap way to get a PC that can be rolled back to the last snapshot taken — which means not re-installing software all over again. As well, the underlying system is still UNIX and thus my sysadmin is still in complete control — and can fix what goes wrong. Not the case at all were we to go to real PCs.

- Back-ups have followed a similar ramping up over the years:
  - 5 MB tapes — until there was more data than any individual tape could hold
  - 5 GB Exabyte tapes
  - DTL tapes (10 and 20 GB)
  - currently: disc-to-disc back-up

  When my own work was in high production mode, back-ups were done every week, and a year's worth of back-ups were stored. Now it's more of an on-demand basis: when proofs and final production PDFs have been generated, I usually ask that one be done. However, with ZFS snapshots as backups, I have access to old versions of files back to Jan. 2008.

and I became a sole proprietorship, as they call it. A home-based business with one employee, in short. On April 1st, 1991, I filed the paperwork for my company, the Carleton Production Centre (CPC).

4 Freelancer at last — the JPC becomes my CPC

My fledgling office was marvellous! Large window, an enormous table surface, and all the comforts of home. Clients came to the house, or I cycled in to their university offices. Almost idyllic!

4.1 The home network

The inevitable shift away from mainframes to workstations in the late 1980s/early 90s on campus also had repercussions at home: we wired the house for UNIX workstations, and the home network was born (see Table 1). I had to learn, in order to do my work; and my husband Mike (not Dunleavy!) used the network to do beta-testing for the university.

The first machines were two Sun IPCs, at $5000 each (the educational discount at the time). Along with Sunblade 1000s, bought perhaps a decade later, these were the only machines we actually bought new; everything else has been on the trailing edge in hardware for us (EBay, re-sellers, university cast-offs), all repurposed for the home network.

We've gone through several machines since then, with changes in operating systems, hardware, back-up procedures (my husband swears by ZFS, for those who're interested) — but on only three occasions, upgrades to TEX.

Then there are the monitors — the change has been physically enormous. The usual tiny PC monitors had been replaced by the enormous cathode-tube monsters that came with workstations, as deep as they were wide, always taking up too much room. Flat-screen panels suspended on arms have made a huge change in the battle for real estate on the desk. But now the skirmish to move sideways with two such flat screens side-by-side is looming. I'm resisting . . . so far.

As for printers, my LaserJet 4M, bought in 1991 for about $2,500 finally died last year (2011). It's been replaced by a Lexmark X543d: it prints colour, it scans paper into `.pdf` files, it inhales toner of all colours. The LJ4 never did any of that — but it did give me 20 years of solid reliable service. I somehow don't see the Lexmark hanging on for that long.

4.2 Ahhh! This is much better!

The change from mainframe to workstation yielded a number of improvements right away:

- No longer hampered by finite disc storage space.
- Files were edited with Emacs and run through TeX *on the same machine!* Until the switch, files had been edited on a PC with Emacs, then uploaded to the mainframe, where they were processed by TeX.
- Saving old projects became feasible — all the work on the mainframe had to be deleted as each issue/book was published; there was no archiving possible (unless we wanted to pay for taped back-ups). Now I had — at that time — infinite storage … and 'infinite' seems to keep getting bigger, as the years go by.
- Printing was done on-site, in the same room as the computers.
- The monitor screen was enormous, compared with those used on either the PCs or the ones accessing the mainframe.
- Previewing — actually seeing the typeset results on-screen and not proof-to-paper — was a huge advance. Saved loads of trees, I'm sure!

4.3 Moving (slowly) through TeX versions

Well, in the beginning there was plain TeX and then I moved gingerly over to LaTeX around 1995.[13] Now, of course, it's LaTeX 2ε (Table 2). I don't use pdfTeX or pdfLaTeX since I'm totally dependent on pstricks — still one of the great tools from the early days.

Until recently, I had three TeX versions available: 'oldtex', 'newtex', and 'rnewtex' ('really new TeX'!) were command-line switches on my machine. When the second TeX Live was installed, I made Mike keep the older one around, as that's what all the archival files from before 1999 ran on and I've been very nervous that old plain files won't run the same; I'm even more nervous about the old LaTeX 2.09 files, mainly because their macros were already rather hybridised versions of the original plain macros. I actually had to use the 'oldtex' set-up a few years back, in order to generate `.pdf` files from very old `.tex` and `.dvi` files. And I'll be doing the same with even older files from *CJL*'s backlist, for the journal's 60th anniversary in three years' time.

My current TeX is from the 2003 TeX Live CD — 'rnewtex'. Once it was available, the previous 'newtex' became redundant, and was retired. And it's that old 2003 CD version which finally gave me access to LaTeX 2ε. However, all of these run on the one remaining piece of old hardware we have; all the other machines have been replaced with faster and more energy-efficient ones. The upgrade from Sparc to x86 has also made it possible to provide a platform for me to install PCTeX for client work.

Table 2: TeX versions I've gone through

original TeX on my UNIX box:

```
This is TeX, C Version 2.93 (preloaded
format=plain 90.2.15) 15 JUN 1993 17:53
```

oldtex

```
This is TeX, Version 3.1415 (C version 6.0)
(format=lplain 94.1.31) 8 MAR 2000 13:55
... LaTeX Version 2.09 <25 March 1992>
```

newtex

```
This is TeX, Version 3.14159 (Web2C 7.2)
(format=latex 1998.3.25) 18 NOV 2003
... LaTeX2e <1997/12/01> patch level 1
```

rnewtex

```
This is e-TeXk, Version 3.141592-2.1 (Web2C
7.5.2) (format=latex 2004.3.11) 10 JUL 2012
14:51 .. LaTeX2e <2001/06/01>
```

… and coming soon …

An x86 implementation TeX … perhaps by Christmas 2012.

After this conference, the testing begins, to ensure that old files using TeX from an old machine will still run on the new architecture, with a more current TeX implementation.

5 My favourites

5.1 Macros

My all-time favourite macros are `\llap` and `\rlap` ('print but don't measure') and `\phantom` ('measure but don't print'). They're everyday tools for dealing with things that won't fit: in tree diagrams, tables, faking decimal alignments, overwide column entries. These are invaluable to me. I learned them early, doing linguistic tree diagrams within `tabular`.

Early on in typesetting linguistics material, we needed super- and subscripts in text rather than math mode, where the vertical displacement (up or down) was preset. So we did a lot of `\raise`ing and `\lower`ing of `\hbox`es. The flexibility of assigning any value (we used `ex` as our unit of measure) was applied to everything from individual characters to whole words. And as this was TeX, one could then combine the up/down of the `\hbox` with the left/right shifting that the 'lap' commands afforded, to give a roughly centred combination of characters.[14]

[13] It was Mimi Burbank who convinced me that I didn't need to rewrite/have someone rewrite all the plain macros into LaTeX structures; that there were only a few lines to change near the top and that would be it. And she was right! Once that worry was laid to rest, I was very happy to move over to LaTeX, because that finally allowed me to access the ever-increasing number of packages coming on-line via CTAN.

[14] I know that `\kern`ing achieves much the same effect but never got the hang of it.

These are followed closely by negative `\hskip` and `\vskip` (and eventually their LaTeX counterparts `\hspace` and `\vspace`). As a non-mathematician, the idea of going up by using a negative value for going 'down' was quite arresting at the time. Same for having text pull to the left via negative horizontal values.

Nothing sophisticated about these — they just work without fail.

Ever since he wrote them up for *TTN*, I've been a firm devotee of Claudio Beccari's `\T` and `\B` (Beccari 1993), to insert struts into tables (usually above or below horizontal rules) — although now there's the `bigstrut` package, which has some advantages.

Several years ago, Paul Mailhot of PreTeX introduced me to the `\includeonly` approach for working from a single run-file to control an entire journal issue.[15] I just love it! I've expanded the contents now to include article-specific macros and parameters, so that just about everything is controlled via the run-file, and the `.tex` source files focus on the contents; it's now about 800 lines. A few extracts appear in Figure 8. I suppose I could move all the titleblock material into the run-file as well — it's only just now crossed my mind that perhaps that could be done. But then my source files would be devoid of any useful identifying content at the top ... no, I think it's better this way. The headers/footers are in the run-file, but the actual title, author, affiliation info is still in the source file. One of those 'just-'cause-I-can-doesn't-mean-I-should' decisions.

And finally ... although not a single macro: all the symbols in math mode! Ever since I learned to define a specific font (in the 'old' way, granted), in order to access a specific character, I've plundered the math fonts and many others, for unique and useful characters. These have been as varied as the `\widetilde` used here as a separator, and special symbols used in linguistics optimality theory tableaux: ☞ ✗ ☠ :

```
%% a. pointing hand:
\font\speciala=pzdr at 12pt
\newcommand{\Hand}{{\speciala\char'053}}
\newcommand{\notgood}{{\speciala\char'067}}

%% b. skull-and-crossbones:
%%    single-char font:
%%         Henrik Christian Grove
%%         grove@math.ku.dk
%%         [2002/01/23 v0.1]
\font\specialb=skull at 10pt
\newcommand{\death}{{\specialb\char'101}}
```

5.2 Utility programs

Let's start with xdvi — it goes without saying much more than that! Next — I could not function without dvips and ps2pdf!

To my mind, these three are the solid workhorses that have made it possible to stay with TeX for typesetting documents, and then smoothly interface with the wider world of authors, editors, printers (both human and mechanical), and publishers. They have gradually let go of being worried that something done in TeX will smell of CM fonts, and indeed have gradually let go caring about what's used before they receive `.pdf` files for their proofs, prefinals, and production runs. Which is as it should be — how I achieve the necessary end-product is not really something they need to know much about. One publisher did get very exercised about TeX being behind the `.pdf` files until we finally got him to understand that he wasn't going to be getting any `.tex` files to work on (that was my job) and no-one was expecting him to install and use TeX at any point — he was only doing the printing. And in the end, his production manager wrote to say the output, in Times Roman, did indeed look a lot better than any Times Roman they were able to generate. A nice, if small, victory for the TeX side.

In terms of utility programs, Emma Pease's package tree-dvips, which depends on pstricks, was an early winner, allowing me to typeset complete tree diagrams, using LaTeX's `tabular` environment for node placement and pstricks to draw in lines, and something that had never been possible via LetraLine — curves! Lines and curves with arrows, in solid, dashed, and dotted lines! Marvellous stuff. For now, this means that pdfTeX/pdfLaTeX are out of bounds, but I can still get to PDF via dvips + ps2pdf.

I encountered rtf2latex2e in the mid-2000s and used it extensively. But before even rtf2latex2e, I would regularly save all files in ASCII, getting rid of all coding in the files, to leave a clean playing field for the TeX — and then LaTeX — codes. It was a chance reference to rtf2latex2e that led me to poke around the web and then convince my husband/sysadmin to install it. Compared with coding from scratch, it was like heaven! Life was good, work was easy!

Then about a year ago, an upgrade to StarOffice came with an '`Export to LaTeX2e`' option, and now life's even better, work's even easier.

5.3 `cjlmac` spawns new packages

In the beginning, I didn't input TeX code at all. The basic formatting codes were inserted during the transfer of Gutenberg-encoded files up to the mainframe. I just ran the file and hoped it would get

[15] The basic idea is described in Mittelbach and Goossens (2004, pp. 18–20).

```
%% 22 MAR 07: began using a new approach to putting an issue
%%            together -- a master run-file, using \includeonly and
%%            \include -- based on a model provided by Paul Mailhot
%%            of PreTeX Inc. Copied cjal-run-file.tex from
%%            cjal/vol-10-1/ and customised for CJL (Ch.)
[...]
\documentclass[twoside]{article}
[...]
                                 %% 20 MAY 09: dimens changed:
\usepackage{cjlmac-l3}           %%    \textwidth  = 27 --> 28pc
                                 %%    \textheight = 43 --> 45pc
[...]
\usepackage{slashbox}   %% 24 JUL 09: for AKINLABI tables (in 54,2)
                        %% for FILE-MURIEL (in 55,1)
[...]
%
 art-baker-vinokurova, %% PROOFS sent: 31 MAY 12 %% CORR input: 14 JUN 12
 art-heycock,          %% PROOFS sent: 25 MAY 12 %% CORR input: 13 JUN 12
 art-massam,           %% PROOFS sent:  2 MAY 12 %% CORR input: 12 JUN 12
 art-mcconnell-ginet,  %% PROOFS sent: 18 APR 12 %% CORR input: 5 JUN 12
 art-nickel,           %% PROOFS sent: 13 APR 12 %% CORR input: 5 JUN 12
 art-gil,              %% PROOFS sent: 14 MAY 12 %% CORR input: 12 JUN 12
%
[...]
%%%%%%%%%%%%%%%%%%%%

%% NOTES:
%% a. for blank versos, use \watermark
%%
%% b. FOR MUSE version of .pdf, they don't want the watermark,
%%    so uncomment next line and reprocess the run-file:
%%
%%      \let\watermark=\blankverso
[...]
%% ARTICLES:

\pagenumbering{arabic}

%% For final camera run:
%%  a. uncomment all \crop[cam] lines
%%  b. uncomment next line, to set final pagestyle:
\pagestyle{cjlmac}
[...]
%% BAKER-VINOKUROVA:
\bgroup
   %%  \selectlanguage{english}
   %%  \pagestyle{proofs}
   %%  \pagestyle{secondproofs}
   %%  \pagestyle{prefinalproofs}
   %%  \setcounter{page}{1001}
   \pagerange{177--207}
%
      \def\DUP{\textsc{dup}\spacefactor = 1000 }
      \def\FACT{\textsc{fact}\spacefactor = 1000 }
      \def\NEUT{\textsc{n}\spacefactor = 1000 }
      \def\NSF{\textsc{nsf}\spacefactor = 1000 }
      \def\PUNC{\textsc{punc}\spacefactor = 1000 }
      \def\STAT{\textsc{stat}\spacefactor = 1000 }
      \def\TIME{\textsc{t}\spacefactor = 1000 }
%
   \markboth{\Leftheader{}}
            {BAKER and VINOKUROVA}
%%  \crop[cam]
   \include{art-baker-vinokurova}
   \watermark  %% = p.208
\egroup
[...]
```

Figure 8: Extracts from `cjl-57-2-runfile.tex`

Table 3: Macro packages galore!

algmac.sty	edmac.sty[17]	musicmac.sty
algmac-l.sty	escmac.sty	quemac.sty
burmac.sty	escmac-l.sty	revmac.sty
calsmac.sty	escmac-l2.sty	richemac.sty
campbell-l.sty	flormac.sty	rightmac.sty
cilob-l.sty	flormac-l.sty	ronimac.sty
cjal-l.sty	fosmac.sty	rsmac.sty
cjlmac.sty	gaumac.sty	rsmac-spanish.tex
cjlmac-l.sty	gilmac.sty	scamac.sty
cjlmac-l2.sty	kormac.sty	shortmac.sty
cjlmac-l3.sty	lathesis-newmacs.sty	tolleymac.sty
crispmac.sty	leftmac.sty	wademac.sty
critmac.sty	ling-atl-l.sty	walkmac.sty
demonmac.sty	lukas-l.sty	weber-l.sty

- The ones with `-l` are for LaTeX use — some old, some new.
- Many are simply cjlmac clones, with the much-mentioned 20–30 formatting revisions, to address layout differences between journals.

to the end. After several months, I began learning how to add TeX code, to address changes from the editor. The macros began proliferating, so we had documentation written up (I still have the binder), until we were inputting TeX codes into ASCII files from scratch.

And just as gradually, I began looking at those macros, leafing through *The TeXbook*, and making my first forays into tinkering with the definitions.

As other projects (first journals, then books) came along, I would modify `cjlmac`, renaming it something suitable: `escmac` for *English Studies in Canada*, `burmac` for an enormous book on 12th-century Spanish accounts,[16] and so on.

The proliferation of such `*mac` files (Table 3), where almost 95% of the contents were identical, eventually led to a more sophisticated solution: Macro packages would load `cjlmac` first, and then simply overwrite the 20 to 30 commands which were the significant differences from the linguistics journal's layout.[18] Much more streamlined, took up much less space, and the essential differences were all that the new macro package contained. Each subsequent variant was simply a modified version of the previous, with new macros being either additional reworked macros from `cjlmac` or entirely new ones I was gradually able to write.[19]

```
%% %% %% Production log:
%% 10 SEP 12: began file clean-up and encoding
%%            (to p.3) (Ch.)
%% 11 SEP 12: continued clean-up and coding (to p.24)
%%      NOTE: all former exx nos. are those found on
%%            the .pdf I printed as reference (Ch.)
%% 12 SEP 12: continued clean-up and coding (to
%%            p.34) (Ch.)
%% 13 SEP 12: finished coding file (Ch.)
%% 14 SEP 12: screen-proofed file; generated
%%            proofs (Ch.)
%% [...]
```

Figure 9: Sample of comments in `.tex` file

With the advent of LaTeX 2$_\varepsilon$, things changed again. Instead of cjlmac being the base file, it was article which did the job. Macros from previous packages were brought into the new version, adjusted where necessary, or simply replaced with publicly available packages and their special options or modifications; for example, titling, fancyhdr, titlesec, caption, . . .

And finally, once I began using `\includeonly` syntax, with a run-file for the entire publication, the journal-specific macro file became just another add-in to the run-file; the master file for everything.

One thing I have done from very early on is document my work, with a running commentary on what I'm doing. In my `.tex` files, I keep a dated log of work done (Figure 9).[20] In my macro files, in addition to the same style of entries at the top, I also make notes within the package on any changes,

[16] Part of 'Projecto Burriel'. Originally keyboarded with Nota Bene, auto-converted into TeX, with macros written by Andrew Dobrowolski.

[17] This is *not* the same package first designed by John Lavagnino and Dominik Wujastyk. For my first critical edition (1988–1993), I had heard of this package, which was still in beta form; I hired Andrew Dobrowolski to build on its base, to address all the needs of such a format for John Donne's *Pseudo-Martyr* (Raspa 1993). These were then extended even further to deal with a very large Spanish-language project (Hernandez, 1993). It was keyboarded in Nota Bene, which first required Andrew to write a conversion utility into TeX. After that opus, our edmac and ancillary files continued to be used in several series of Renaissance plays I typeset for Dovehouse Press, Ottawa, until 2005.

My failure to rename the file was inexperience — and also not knowing whether Dominik and John's package was ever going to go beyond beta. It eventually did, and so our home-grown package has remained in-house, to avoid confusion. And my impression is that the 'real' edmac has gone far beyond what ours had, so I don't think ours is a loss to the TeX community. But it is a bit of an object lesson in the need to adhere to the tenet that variations on an existing package, even if it's only in beta, *must* be renamed, even if you don't think the original will continue.

[18] I've often claimed there are no more than 20–30 journal-specific layouts that affect the main look of a document: headers and footers, titleblocks and opening page footers, lists, citation blocks, footnotes or endnotes, superscripts, bibliographies, contents page, font size, assigned font usage, text width and height, headings (from 2 to 5 levels), float parameters, indentions, and a few others. Some elements may take several macros to define (e.g., titleblock usually has title, author, affiliation, maybe also email) but for the most part, that's it.

[19] However, much of my macro-writing 'expertise' remains confined to modifying existing macros, copying them from others I've found via the web, asking for assistance on the `ling-tex` list, or as most often now happens, finding a new package on CTAN! I'm no Amy Hendrickson or Karl Berry!

[20] My log entries are always at the top of the file, so they're the first thing I see when I open it.

including which of my previous packages I've copied or modified a macro from. I think I realised that, as the only one working on a file, especially macros, there was no-one else to ask, and there were so many files, it was necessary to keep a record of such things.

5.4 Resources

Again, there are constant and true resources, without which I could not have done TeX typesetting for 30 years: these are the TeX community, email and the Internet, and courier services (!).

But more than any of these, the most important resource I have had has been my sysadmin husband, Mike McFaul. He's done all the computer work that allows me to just sit at my keyboard, and get on with my job. He doesn't use TeX, doesn't much like installing it (requires him to know stuff he doesn't, and I can't help as I don't know the stuff he needs). But he provides the entire infrastructure of a company's IT staff. From hardware/software upgrades to back-ups, from giving me the UNIX commands I need to do certain tasks to fixing files that go wonky, from email service to Internet connectivity, it's all there.

I am very fortunate.

There are many other great resources I've been able to tap and count on. To name a few:

1. Mining down into the TeX community itself, probably a good 20–30 percent of everyone at the conference has helped me in some way or another. Being an active member of TUG since 1986 helped build up those connections, opened doors in a few cases to typesetting contracts, and by osmosis (especially during the years spent editing conference proceedings) I learned and was taught a great many things that no classroom could have ever provided.

   Becoming further involved in that community, I started up the `ling-tex` list in 1994, which was a self-serving event at first. But it is still going, still viable, and many many people have asked far more questions than I! So I'm very pleased it's continued for almost 20 years.
2. Email, to stay connected with that community, and the Internet. Without these, no work in TeX could really be done, in my opinion.
3. CTAN's huge collection of all those wonderful contributions from people around the globe. The CTAN-ANN list has been a great way to keep up, and eventually remember that something came through a while back that might be what I need.[21]
4. Courier services were becoming more pervasive, targetting home businesses in particular. It was easier to get a 24-hour turnaround in documents (when a printed hardcopy was what printers wanted) using those courier services than any inter-departmental mail service or even personal delivery by car. Couriers made it possible to totally ignore where I was, and where my client was: we were both within 24 hours of one another for physical materials, and almost instantaneous contact via email and phone.

6 How things have changed

6.1 Exchanging and processing submissions

At the start, there were no author submissions on disc — everything was input from scratch. The *CJL* editor would edit the hardcopy ms., and I would input both the text and the edits. Everything was done on our Apple][clone, formatted with Gutenberg software, to generate proofs that were already roughly in the final layout (as described above).

Once author submissions began coming in on floppy, the files were converted to ASCII, for use with Emacs. We had both a PC and a basic Mac in the office and some conversion tools to bring the files into an un-encoded state, ready for inputting both edits and formatting codes.

By the late 1980s, author submissions increasingly came on disc (both PC and Mac) using a variety of software: Word, WordPerfect, Nota Bene, WordStar are the main ones that come to mind. The editors would send edited hardcopy and matching disc, and I'd input those edits.

Even after moving from the mainframe to the UNIX box, the same procedures were followed. When I had employees, they used WordPerfect for inputting edits and TeX code, avoided any of the WP function keys, and saved the files in 'Text only' format. The files were then saved in ASCII, I checked them in Emacs, and then ran them through TeX.

By the mid- to late 1990s, submissions seemed to converge on Word's `.doc` format, the PC and Mac versions having pretty much aligned.

By the early 2000s, editors were taking care of editorial changes themselves, sending only edited `.doc` files and a matching hardcopy by mail/courier. This was as much a cost-cutting choice as a practical one — editors were themselves becoming skilled users of the software, for their own academic work, and were transferring that skill to their journal editing.

This convergence on the `.doc` format made it possible to take advantage of RTF as a bridging mechanism: use StarOffice to save the file in `.rtf` and run that through the utility rtf2latex2e to yield

[21] The CTAN-ANN notices also appear on TUG's home page, `www.tug.org`.

a fairly accurate TEX-encoded file. Not only did this save time but it also ensured that all the care and time authors had taken, especially with their special characters and font changes, would be preserved.

In most cases, the cleaning out of extraneous TEX code was still more efficient than inputting it all from scratch. However, on occasion, a file would have a great number of special characters that failed to be picked up by the half-way step through RTF. The only way to deal with that — to make sure that all the original and unique keystrokes were retained — was to go into the .doc file and replace each one with a unique 'dummy code', some alphanumeric sequence that would pass through the conversion untouched.

As mentioned elsewhere, rtf2latex2e has now been replaced by StarOffice's own export utility into LATEX, which has been an even bigger time-saver.

At present, the only technical hiccup is with .docx submissions. Fortunately, editors have been most accommodating in resaving in the older format.

Somewhere in here, our Mac became obsolete: no Mac floppy had been seen in ages. And its removal cleared up desk space! Indeed, there were no discs from anyone — emailing files had become much easier and there were no more costs incurred either buying discs or shipping them along with their hardcopy mss. More cost savings for everyone. And I could get rid of some equipment from my work space.

The PC was still needed — its Acrobat Reader was better at viewing/printing .pdf files than the UNIX version, especially if font substitution was happening.

By the mid-2000s, I began sending .pdf proofs to editors, but still courier'd the marked-up mss, which had all my queries and comments marked.

Editors then began using the .pdf format to replace the hardcopy and both it and the .doc file would arrive via email. No more courier costs at either end.

However — I still work entirely from paper. I'm a firm believer in the paper trail. The manuscript .pdf is printed,[22] to get hardcopy for my queries and comments.

Corrections come back as scanned .pdf pages (which I then also print, to make sure everything's been taken care of). Some authors are sending back the proof .pdf, with annotations. Unfortunately, I can't print these pages, so those annotations often have to be written onto a hardcopy of the page needing correction.

Something I've been doing lately is printing the .pdf manuscript files in landscape, two pages per sheet — saves paper at my end. And when I scan the marked-up manuscript into a .pdf, it's fewer sheets for the recipient. Anything that's not quite clear can be zoomed in on via the Reader.

Now, everything is done in PDF. Even the author's marked-up manuscript is scanned into a .pdf and sent via email or left on an ftp server (file sizes are expanding, like the universe!). Mailing and courier costs have been reduced to almost nothing.

The physical paperwork that does remain at the end is then returned to the editors (by regular mail now, rather than courier). Where once I would go through a box of paper (5,000 sheets) every half year or so, a box now lasts me a couple of years at least. Yet another cost savings.

Hardcopy camera copy on high-contrast laser printer paper has similarly given way to production .pdf files for printers. And here, the cost of moving physical material around has been removed entirely.

Some printers (companies, I mean) were initially leery of my .pdf files, until it was clear that no font substitutions, even of the Base14, were happening. I had poked around the web, looking for ps2pdf commands in submission instructions for various publishers and publications and finally found the right syntax (all on one line):

```
ps2pdf -dPDFSETTINGS=/printer
   -dCompatibilityLevel=1.3
   -dMaxSubsetPct=100 -dSubsetFonts=true
   -dEmbedAllFonts=true
```

As a very small operation, I feel that I have to be as good as the 'big guys', and make sure no objections can be found for rejecting my files — and thus losing a client to someone using more widely-used — known software familiar to the printers. But soon there was a more concrete reason to have found this solution.

Publishers of books and journals now often require a matching .pdf file for web-based distribution of the publication. I am eternally grateful to the people who devised ps2pdf and the switches that fully embed all the fonts, include the current Base14 (used to be Base35, remember?). I have solid robust .pdf files — at least, I've never heard anything to the contrary.

The next step after .pdf files for the web has been the need (so far, only one journal) to have live links. So, back to CTAN for the solution — which is, of course, hyperref. As with a number of these utilities, I come to need them years after they've been developed, which means that most of the bugs and such wrinkles as cross-package interference issues have been found and usually resolved. Being behind

[22] The exception is files where so much font substitution has happened that it's more accurate to print the .doc file!

the curve has its advantages at times!

Graphics, illustrations — anything I can't create with TeX or xFig. In the beginning, `.tex` output had blank spaces where hardcopy provided by the authors would be cut-and-pasted into position. In those days, there were very few illustrations in *CJL*; most were for the *Algonquian Papers*.

It wasn't till the early 2000s that authors began offering up figures in various formats, often of a quality far inferior to their text. As I had none of the half-dozen or so popular software packages around at the time, any edits or changes needed had to be done by the author — and this didn't always work out.

A couple of years of this and I finally decided I had to find someone who could handle graphics files — I simply was not skilled enough, and it bothered me that great-looking text (which had been properly edited) was being accompanied by shabby figures (which often weren't consistent with that text). I think a small-time typesetting business like mine does have to weigh the costs (time and money and expertise) of doing it all, or paying better people to do things you're not so good at. Again, advances in technology kept making this more cost-effective: email renders geographic location irrelevant, attachments and ftp simplify file transfers both to and from the graphics person, payments made via PayPal (what a marvellous invention that's been, eh?!).

6.2 TeXnical enhancements over time

As mentioned earlier (Table 2), we've only upgraded TeX three times, none of them smooth ('Why can't this install like a normal UNIX program?!' is often heard around here). Based on what people were telling me at this summer's Boston meeting, things are quite different now, so I'm almost looking forward to having the new TeX Live DVD installed.

So, leaving aside such major upgrades, the progression of enhancements to TeX and company have been significant and noticeable, and the output of the past 30 years visibly reflects those changes:

- AM fonts to CM fonts for text: this is purely an appearance thing.
- The in-house bitmapped `ph10` and `ph7` were replaced by the Metafont suite of WSUIPA fonts from Washington State University, which at last made italic phonetic characters possible. The WSUIPA fonts were in turn replaced by Fukui Rei's TIPA/XIPA. This upgrade path is about more than just appearance.

  The TIPA fonts, under LaTeX, make the phonetics font as fully flexible as the regular text fonts. The greatly expanded repertoire of available characters, as well as the tools to build even more complex combinations, is a tremendous change. On the more mundane side of things, the coding intricacies have been greatly reduced. For example, looking at diacritics for these non-CM fonts, one can appreciate the simplicity which TIPA now affords:

`ph10`	`\v{\normalphn\char'162}`	n/a
WSUIPA	`\v\tailr`	ˇɽ
	`\v{\ipa\char'106}`	ɽ̌
TIPA	`\v\textrtailr`	ɽ̌

- plain TeX eventually gave way to LaTeX 2.09, and now to LaTeX 2ε.
- In-house macros based on `manmac` gradually hybridized into a combination of newer LaTeX-based syntax working alongside existing plain TeX definitions, supplemented with task-specific packages from CTAN: caption, tree-dvips, multirow, and so on.
- Every new layout or character in authors' `.doc` files has found a response in CTAN's ever-expanding collection of packages: that is, as authors' writings become more complex, there is often a package that already addresses that need. In the old days, a new layout was usually greeted with 'Why?' and 'Is there a significant difference in meaning by not doing this the old way?'. Now, it's usually just a matter of either finding an appropriate package or asking for assistance via the `ling-tex` list!
- Documentation on using TeX has gone from one, sometimes inscrutable source, *The TeXbook*, to an explosion of information, both printed and on-line, for all user levels and for far more than 'just math and science'.

7 ... and some things hardly change at all

In no particular order ...

- I work from paper — `.pdf`s with sticky notes are quite simply annoying: I can't print them to have hardcopy proof of what the author's corrections are, I can't cut-and-paste anything out of those sticky notes, and I can't compare pages easily. So I have no use for 'em. And of course, with the basic Acrobat Reader, I can't generate them either! So I'm a paper-based editor and typesetter and that's just the way it is — it's faster and more accurate for me to write notes by hand than to mess around with teeny tiny notes that wink out whenever my cursor goes out of the yellow box or into the yellow box or some place in between. Yuck.

- Annotations at the top of a file, showing its work history. I noticed this the first time I edited papers for the 1986 proceedings for the Montreal meeting. As I was exchanging files with the authors, it became a habit. Many prefer having such log files at the bottom, out of the way, but I prefer opening the file and seeing right away what's been worked on last, whether it was me or someone else, the commented history has been a terrific crutch, when there's just too much to keep track of or simply as memory fades …
- Pens — I depend on Pilot Hi-Techpoint V5 pens in extra fine turquoise to mark *all* my edits, notes, comments, queries on hardcopy. Early on, Bill and I used non-reproducing light-blue pencils (or markers): their marks wouldn't be noticed by photocopiers and, for the most part, by printers (at that time). So we were safe in making small annotations on final copy (instructions to the printer), or using a ruler to mark tree diagram lines on camera copy for placement of LetraLine.

  We had our pen colour assignments: blue was mine, his was green, and authors could have anything else (!). None of that happens anymore, and yet, I still use those turquoise markers, which are actually rather difficult to find these days. I suppose I could change but … These days too, I like corrections made in red as it's so much easier to spot than the ever-so-diplomatic lead pencil or black pen — useless when the eyesight starts to go!
- For tree diagrams, I still use `tabular` and `tree-dvips` rather than front ends such as `qtree`; I like to think I have more control over the appearance. And at this stage of the game, I'm too lazy to learn more than I have to (!).
- Daily tracking of hours per project — this has always been to the quarter-hour, marked on slips of paper. Then manually transferred to each project's production chart, tallied once a job's done, and then (eventually) an invoice is done up. Not efficient at all. But I don't see it changing at this point!
- Rates — I've increased these perhaps twice. Initially, it was per hour, then per typeset page, now a combination: typeset pages + per/hour when the hours exceed the norm. Eventually this builds up a track record of how much time a project (journal issue or book) ought to take, compared with what it actually took. Then I see how things look, on balance, over the long term.
- Production charts and account books — also done manually, but at least here I do the entries as invoices are sent and payments received. Needless to say, the forms are typeset in TeX.
- I still print on both sides of the page — in the days when proofs were on paper, my stack of recycle paper was much higher.
- Fonts — my knowledge of how to set up and access the many hundreds of fonts available these days continues to be hampered by being an end-user on the UNIX box, unable to explain exactly what goes where, when it comes to all the font files. I understand that there are good installation tools/scripts these days but … well, everyone's got a black box they sort of dread having to open, and this one's mine. I really and truly have had no feel for this subject, and it's been a real barrier.

  The one time I did need something special was Baskerville, for a collaborative project (Beecher and Ciavolella, 2010) where the main work had been done using Y&Y TeX, which I already had on my laptop (running Win98). In fact, the PC was networked to my workstation, with the big monitor, so I did all the editing in Emacs and then would turn to the laptop (and its tiny screen) to process the file. I found the program was really nice to use, but the fact that both the laptop's OS and the TeX implementation were ancient made it unrealistic to undertake new projects. But I have to say that the ease of use for installing fonts was a real treat. I just wish it were that easy with UNIX!

8 Passing TeX along

8.1 Within the university environment

Along with this student job working on the *Algonquian Papers* and *CJL*, I was also working on my master's — and of course, it was going to be typeset using TeX![23]

By 1985, along with the fellow who'd written the user guide for the *CJL* macros (Mike Dunleavy), I was asked by the university's Computing Services dept. to help write *The Local Guide to LaTeX* — how many of you have done the same thing at some point?![24] By virtue of our working on *CJL*'s TeX

[23] Using `cjlmac` and plain TeX and AM fonts.

[24] For those who don't know this bit of history … Both editions of Lamport's book *LaTeX: A Document Preparation System* tell the reader (on p. 2) that "how you actually run LaTeX depends upon the computer system … For each computer system, there is a short companion to this book, titled something like *Local Guide to LaTeX for the Kludge-499 Computer* (the 2nd ed. calls it the *McKludge PC*). After the initial confusion about the existence of such a document, people finally figured out that it meant the creation of a "local guide" had just been passed along to *them*!

macros and documentation — and apparently a fear that our expertise might lead us to leave the university — caused various administrators to find a way to keep us around; Computing Services got the mission.

While we wrote up this local guide, several other elements began to coalesce and converge on LaTeX:

- The idea of using LaTeX for thesis production was taking shape. I was just finishing mine, using plain TeX, for linguistics, so surely LaTeX would not only be easier but would also help move TeX out of the math/science sphere into the larger (and TeXnically less demanding) humanities faculties.[25]
- An institutional membership in TUG was taken out by the university, with copies (seven, I believe) being distributed across campus to likely hubs of interest (Math Dept., Engineering, Computing Services, *CJL*, and so on).
- The math department was pushing to have its secretarial staff work on faculty members' papers, notes, and so on, using TeX, so there was training that needed to take place.
- Various engineering departments were looking to do the same as well, and while some had a few secretaries already working with TeX, more formal courses were being sought.

The net result of all these threads was a series of mini-courses on using LaTeX for various purposes, a *Local Guide*, a macro package called lathesis, and the Journal Production Centre (JPC; see Section 3).

8.2 Beyond the university ... someone knows someone who ...

You never know when someone who knows what you do will act on that knowledge and pass your name on for potential work. I've never done any concerted advertising about what I do and yet, by word-of-mouth, many jobs have come my way. I think this is the way with many TeX users who either dabble as consultants or do it full-time. Our community is as much about recommending its fellows as it is about helping them solve problems.

A poster session at a 1993 SSP (Society for Scholarly Publishing) conference,[26] which focussed on TeX's suitability in a large-scale production environment, especially one with journals and/or series, made quite an impression. One interested fellow was from the NRC (see below) and even at that early date, thought the idea most applicable to the Research Press, although nothing came of it at the time.

On the other hand, it was about 18 months later when someone from the U.S.-based Oxford University Press office contacted me, saying they'd spent a long time trying to find me (how much of an ego-boost do you think that was!). They had a difficult text, done for the most part in TeX, which needed to be set in the house style. I worked for a very exacting but patient person, and eventually the book was done. A second followed — and that was it. Nevertheless, it a very good learning experience with a "professional" publisher and had come about by happenstance.

Similarly, one of the secretaries in the math department mini-course I'd taught moved on to a job at the NRC (National Research Council, Canada) in the Research Press around 1998. It was she who had them contact me to teach them how to use TeX for their journals. This became my first consulting job — with Robin Fairbairns doing all the heavy lifting, in terms of writing up macros to suit their one- and two-column journal formats. TeX was replaced by XML-focussed software around 2004; however, late in 2011 they decided to call us back in and move their physics journal back into TeX: it was just too hard to make physics fit into the XML mould.

Being rather specialised in linguistics (as is the current TUG president, Steve Peter, by the way) means it's easier for people to remember you, and contact you. Several years ago, the TUG office itself passed my name on to an American publisher, and again, it was a TeX-encoded project which needed to be adapted to their specifications. Curiously, something very similar came my way a couple of years ago (files which had been worked on since the early 1990s!) and that 1245-page volume is now in print, the last in its series.

It's always a challenge taking TeX-encoded files that have been worked on for a long time, because they show the evolution of the author's skills over time — as well as availability of packages that may or may not have undergone changes and/or upgrades.

The long and short of it is — as expertise increases, and the client base expands, opportunities arise from unexpected quarters.

8.3 Joining the TeX community

While my work using TeX and my involvement with TUG were deeply intertwined on a daily basis, lengthy reminiscences about the latter should wait for an-

[25] I'm sure the associated 'discussions' on how to bend, adjust, or simply silently circumvent some of the university's typewriter-based requirements are familiar to many who've done the same thing. I smiled when I saw that two presentations at this conference (Flynn, Veytsman) would be about using LaTeX for thesis preparation. I suspect there are many of us out there who've been down a similar path.

[26] This is one organisation (`sspnet.org`) I would strongly encourage joining; they are a year or so older than TUG, and represent a very broad range of fields related to publishing. The exposure to potential clients is one not to ignore.

other time, another article. Some highlights might, nevertheless, be pertinent at this point.

Only a few years after I began working with TeX, I joined TUG (1984), and then became active on the board, from 1987 till 1995, serving on many committees and in several positions, culminating in president. I'm still a member — just a lot less active. Almost everything I know today about meeting procedures and about collaborating with very different people (TeX expertise, languages, academic backgrounds) stems from my years on TUG's board and continues to serve me well.

I attending the annual meetings for TUG for many years, as well as a few hosted by various European user groups. Often the committee and board work limited the time spent listening to presentations but there was always lots of time spent talking and listening outside of the formal schedule — useful and indeed invaluable, especially in the period from 1986 till 1993, when one's fellow attendees were the main source of information, support, and teaching.[27]

As well, for many years, I edited our conference proceedings, from 1988 and 1989 alone and was grateful when others joined, as of 1990. Building on this, it eventually lead to the creation of the *TUGboat* production team.

For a time I was a member of the original production team, along with Barbara Beeton, Mimi Burbank, Robin Fairbairns, Sebastian Rahtz, and Michel Goossens. Being part of that group gave me exposure to a different level of production values, definitely a very different level of TeX programming and skill levels, and models of how to make collaboration over long distance (and time zones) work smoothly and productively. That education, along with all the time spent on TUG's board, has come back repeatedly in all my subsequent activities. So I am extremely grateful for having had these opportunities, very rarely conducted in person, to expand and extend my own abilities.

From 1991 till 1995, I was involved in getting TUG's first small-scale newsletter, *TTN* (*TeX and TUG News*); Peter Flynn carried on with *TTN* for several years after that, until *TTN*'s contents were folded back into both *TUGboat* and Lance Carnes' on-line *PracTeX Journal*.

While I think I'm still listed as the contact for a Technical Working Group in linguistics, it's the `ling-tex` list 1994 which has been the more successful mechanism for the community using TeX for linguistic material.

I started the `ling-tex` list in 1994, and it's still going strong. It's just about the best resource for me and my linguistics typesetting — and it seems to serve that same purpose for a great many other people. I have to admit that I'm quite proud of the list, that it continues to function, providing users of all skill levels the help and encouragement they need. Since 1995 or so, it's been generously housed at the University of Oslo and is still being maintained by Dag Langmyhr.[28]

From around 2000 till 2003, I was part of a technical support group for users of Y&Y TeX, along with Mimi Burbank and Robin Fairbairns. As often as not, I found myself learning a lot about PCs and how TeX ran on them. We wrote documentation, and did an awful lot of testing. Eventually that came to an end, but the collaborative atmosphere was a very positive and productive one.

The details of all the above can be found in the pages of TUG publications and its website.

Much of this activity has now ceased — except for `ling-tex`. And it's been 13 years since I last came to a TUG meeting. But perhaps because so much communication has been done via email that it never seems that long. Friendships and acquaintances, never really gone, are picked up again with an email query; familiar faces are seen again, if a little older — and, of course, so much wiser `:-)`.

9 Life and personal change

9.1 Then ...

I was single when this whole adventure began. In time, my thesis was completed, so I no longer had to split my time between earning money and doing research. I could stay late any night of the week, working in my office, dawdling about with paperwork for the journal or single-mindedly plugging away at coding or debugging. I'd be there most Friday nights till late,[29] order pizza, and be happy as a clam. I had a computer in my apartment as well, and would do the same there.

Eventually, I had a boyfriend, and work was confined more to regular hours and more to the

[27] Unlike today's annual meetings, this was when vendors were showing their newest software for PCs and Macs, improving the user interface, selling books that were a precious source of information ... the web hadn't yet really turned up so all we had was *The TeXbook*, our *TUGboat* issues, and one another.

[28] However, I want to point out that the list was first hosted by George Greenwade at Sam Houston State University (SHSU); I'm grateful to George for that generosity in taking an idea and making it reality.

[29] Not something one probably needs to do these days, given the portability of devices as well as the current system of connectivity everywhere.

office. In 1991, we had rented a house, and it became possible to consider moving my work place from the university to home, as an independent business.

First thing — Mike wired the house to network our brand-new IPC computers (wires strung through the walls or just lying along the baseboards). I had an upstairs bedroom for the office, which clients reached by coming up quite steep and narrow stairs. I got a new LaserJet 4M printer, which put out 600 dpi as opposed to the old one's jagged 240 dpi. I looked like a pro!

Next was finding ways to getting paperwork to and from my clients still at the university — increasingly I depended on couriers, rather than cycling in (not so good in an Ottawa winter!). It soon became clear that the courier services took away one last hurdle — what happens when a journal editorship moves away from Carleton and Ottawa? It was virtually a seamless shift, given the 24-hour turnaround that most companies were offering. A seemingly trivial point — but with almost 8 years of being on campus, sneaker-netting between client offices and my own, the client possibilities beyond the university suddenly became very real.

The downside to this was, of course, a significant decrease in cycling to university, as a car was so much easier ... and clients so much further (the NRC, for example, were on the other side of the city!) and the gradual decline in my level of fitness ... !

We married in 1998 and added our lovely daughter in 2000. I stopped working till 2002, starting up again when Anna went to home care, just across the backyard. Productivity jumped right back up there and remained fairly steady as school became a full-day routine.

Journals in Canada are usually funded partly by memberships in associations, partly by government grants. As money became tighter in the late 2000s, some of my client journals left, either to be produced only in on-line versions (usually via Word) or they simply faded away. Table 4 shows the ebb and flow in pages and projects over time.

Some of the small publishers whose books I had typeset for years also ceased operations: one sold his backlist to a larger publisher, another finally retired. I got a few not-quite-finished projects, where I had to find my way through someone else's TeX code — like doing historical research! There have been various editing jobs, with or without the subsequent typesetting, and several very long-term book projects.

9.2 ... and now

Now I have only two journals: *CJL*, the old stalwart, is still with me,[30] and an annual collection of papers from the University of Ottawa's OLBI (Official Languages and Bilingualism Institute); and there's the renewed consulting work for the NRC.

Table 4: Page counts 1992–2011

Year	Pages	Projects	Year	Pages	Projects
1992	2,365	13	2002	3,152	18
1993	3,609	19	2003	2,077	10
1994	3,240	16	2004	715	4
1995	2,737	17	2005	976	6
1996	3,643	21	2006	937	5
1997	3,528	19	2007	1,009	5
1998	2,435	13	2008	1,357	9
1999	2,984	15	2009	1,396	8
2000	3,490	21	2010	613	4
2001	1,023	6	2011	630	4

From time to time I lend a hand to others, editing texts and/or inputting edits to TeX files. I don't mind bibliography work either, which is often where editors have difficulties.

It's been a good run so far. Many inventions and opportunities, coming together at the same time, have made it all work out quite well, I think.

TeX has allowed me to have my own small business and work from home, close to my family. It has made it possible for me to have a job where I know I have contributed to the larger world of scholarly research and publication, as well as passing my skills and enthusiasm on to others of the same mind.

TeX — and TUG — have led to long-standing friendships in many places far beyond Ottawa, experiences in working on a variety of boards and committees and lists, and an appreciation of the role each of us can have in making things better for one another, and for everyone.

References

[1] Aubin, George F. 1983. A Lord's Prayer in Wampanoag? *Actes du quatorzième congrès des algonquinistes*, ed. William Cowan, 239–244. Ottawa: Carleton University.

[2] Beccari, Claudio. 1993. "Hey — it works!". *TeX and TUG News* (*TTN*), 2,3, pp. 10–11.

[30] Having started with vol. 29 in 1984, I'm currently finishing vol. 57. For its 60th anniversary, my proposal to have the entire backlist put into PDF files for its MUSE collection has been accepted — which means some new work over the next year or two. I have files back to vol. 35 (1990) and will generate true searchable PDF files through vol. 47 (2003). Everything before that (1954–1989) will be scanned into PDF, from originals held by the University of Toronto Press.

[3] Beecher, Donald, and Massimo Ciavolella, eds. 2010. *De la maladie d'amour ou mélancolie érotique*. Paris: Éditions classique Garnier.

[4] Carleton University. 1987. *Local Guide to LaTeX*. Ottawa: Computing Services, Carleton University. (Written by Mike Dunleavy and Christina Thiele.)

[5] Erickson, Vincent. 1983. "The Mohawks are Coming!" Elijah Kellogg's Observation. *Actes du quatorzième congrès des algonquinistes*, ed. William Cowan, 37–47. Ottawa: Carleton University.

[6] Glenn, Patricia D. 1983. "Gutenberg (evaluation)". *Creative Computing* Vol. 9, No. 6, p. 64 (June). `www.atarimagazines.com/creative/v9n6/64_Gutenberg.php`.

[7] Guenther, Dean, and Janene Winter. 1990. An International Phonetic Alphabet. *TUGboat* 12:1, pp. 149–156.

[8] Hernandez, Fernando. 1993. *Las Rentas del Rey: Sociedad y fisco en el reino castellano del siglo XII*. Madrid: Ramón Areces.

[9] Lamport, Leslie. 1986. *LaTeX: User's Guide and Reference Manual*. Reading: Addison-Wesley.

[10] Lamport, Leslie. 1994. *LaTeX: A Document Preparation System*, 2nd ed. Reading: Addison-Wesley.

[11] Mallett, Rick. 1974. TeX82 on CP-6. *TUGboat* 4:2, pp. 73–74. Site coordinator report. `tug.org/TUGboat/tb04-2/tb08site.pdf`.

[12] Mittelbach, Frank, and Michel Goossens. 2004. *The LaTeX Companion*. 2nd ed. Boston: Addison-Wesley.

[13] Lukasiewicz, Julius. 2011. *Rue Lukasiewicz: Glimpses of a Life*. Ottawa: Golden Dog Press.

[14] Pullum, Geoffrey K., and William A. Ladusaw. 1986. *Phonetic Symbol Guide*. Chicago: University of Chicago Press. (2nd ed. published in 1996.)

[15] Rahtz, Sebastian P.Q. 1987. *Information Technology in the Humanities: Tools, Techniques, and Applications*. London: Ellis Horwood Ltd.

[16] Raspa, Anthony. 1993. *Pseudo-Martyr*. Kingston: McGill–Queen's University Press.

[17] Thiele, Christina. 1987. TeX, Linguistics, and Journal Production. *TeXniques* no. 5, pp. 5–26. Conference Proceedings from the 8th Annual TeX Users Group Meeting, Seattle, Aug. 24–26, 1987.

[18] Thiele, Christina. 1991. "Hey — it works!". *TeX and TUG News* (*TTN*), 0,0, pp. 30–31.

[19] Thiele, Christina. 1992. So we know what we're doing, eh? Paper presented at UKTeX Users Group meeting, "Book and journal production", London (February). Unpublished.

[20] Thiele, Christina. n.d. '... from the deepest archives ...'. Ottawa: Carleton University. [An in-house 'survey' of our progress using TeX.]

[21] Thiele, Christina. 2001. In Memoriam: William G. Cowan (1929–2001). *Canadian Journal of Linguistics* 46:291–306. [Not yet available on-line via MUSE.]

[22] Thiele, Christina. 2007. Interview Corner. `tug.org/interviews/thiele.html`.

◇ Christina Thiele
15 Wiltshire Circle
Nepean, Ont. K2J 4K9
Canada
`cthiele (at) ncf dot ca`

Changing the font size in LaTeX

Thomas Thurnherr

Abstract

Changing the font size in LaTeX can be done on two levels, affecting either the whole document or elements within it. Using a different font size on a global level will affect all normal-sized text as well as the sizes of headings, footnotes, etc. By changing the font size locally, however, a single word, a few lines of text, a large table, or a heading throughout the document may be modified. Fortunately, there is no need for the writer to juggle with numbers when doing so. LaTeX provides a set of macros for changing the font size locally, taking into consideration the document's global font size.

1 Changing the font size on the document-wide level

The standard classes article, report and book support three different font sizes: 10pt, 11pt, 12pt. By default, the font size is set to 10pt and can be modified by passing any of the previously-mentioned value as a class option. As an example, suppose you want to change the font size for normal text to 12pt throughout the document. For the class report, this is how you would do that:

```
\documentclass[12pt]{report}
```

In most cases, the available font sizes for the standard classes are sufficient and you do not have to bother about loading special packages that provide more options.

1.1 Extended font sizes for basic classes

Should you ever require a different font size, however, the extsizes package comes in handy. Along with the standard font sizes mentioned above, it provides the following additional options: 8pt, 9pt, 14pt, 17pt, and 20pt. As these font sizes require a reimplementation of the document classes, names are slightly different from the standard classes article and report:

```
\documentclass[9pt]{extarticle}
\documentclass[14pt]{extreport}
```

1.2 KOMA-script and memoir classes

The KOMA-script document classes work very much the same in terms of font size as the standard classes. The only difference is the default font size which is 11pt for all classes except scrlettr. The latter has a default size of 12pt.

The memoir class, however, is more flexible when it comes to font sizes. It provides additional sizes ranging from 9pt all the way to 60pt. These options are available: 9pt, 10pt, 11pt, 12pt, 14pt, 17pt, 20pt, 25pt, 30pt, 36pt, 48pt, and 60pt. The following example illustrates their usage:

```
\documentclass[60pt,extrafontsizes]{memoir}
```

The example illustrates a common problem with fonts larger than 25pt and the standard LaTeX font Computer Modern (in OT1 encoding). They cannot exceed 25pt since larger sizes are not defined and therefore not available. The memoir class solves this problem with the extrafontsizes option. It changes the standard font to the scalable Latin Modern in T1 encoding. This is equivalent to the following two lines of code in the document preamble:

```
\usepackage{lmodern}
\usepackage[T1]{fontenc}
```

1.3 Other classes

The AMS document classes have a few more font sizes than the basic classes, though not as many as extsizes. It's always good to check the class documentation to see what's supported — not all classes are the same.

2 Changing the font size locally

A common scenario is that the author of a document needs to change the font size for a word or paragraph, decrease the font size of a large table to make it fit on a page or increase the size of a heading throughout the document. LaTeX implements a set of macros which allow changing font size from Huge to tiny, literally. That way, the author does not have to worry about numbers. The macros, including the exact font size in points, are summarized in table 1.

A good rule of thumb is not to use too many different sizes and not to make things too small or too big.

LaTeX provides two different ways to use these font size modifier macros: inline or as an environment using \begin...\end:

```
{\Large This is some large text.\par}
\begin{footnotesize}
This is some footnote-sized text.
\end{footnotesize}
```

The \par command at the end of the inline example adjusts baselineskip, the minimum space between the bottom of two successive lines.

2.1 More sizes: \HUGE and \ssmall

The moresize package adds two additional options to the list of macros above, \HUGE and \ssmall. The first provides a font size bigger than the largest

Class option	10pt	11pt	12pt
`\Huge`	25pt	25pt	25pt
`\huge`	20pt	20pt	25pt
`\LARGE`	17pt	17pt	20pt
`\Large`	14pt	14pt	17pt
`\large`	12pt	12pt	14pt
`\normalsize` (default)	10pt	11pt	12pt
`\small`	9pt	10pt	11pt
`\footnotesize`	8pt	9pt	10pt
`\scriptsize`	7pt	8pt	8pt
`\tiny`	5pt	6pt	6pt

fontsize	`\Huge`
fontsize	`\huge`
fontsize	`\LARGE`
fontsize	`\Large`
fontsize	`\large`
fontsize	`\normalsize`
fontsize	`\small`
fontsize	`\footnotesize`
fontsize	`\scriptsize`
fontsize	`\tiny`

Table 1: Font sizes available in standard LaTeX.

available by default, whereas the latter fills the gap between `\scriptsize` and `\tiny`.

Since `\HUGE` changes the font size to a number bigger than `25pt` and, as mentioned above, the standard font is not scalable, LaTeX displays a warning saying the font size is not available and that it was replaced by the next smaller (`\Huge`). Again, one needs to use another font type, such as the Times Roman equivalent available in the PSNFSS package (see example below). This way, you can benefit from that "HUGE" font size provided by the `moresize` package. Here is an example:

```
\documentclass[11pt]{report}
\usepackage{mathptmx}
\usepackage[11pt]{moresize}
\begin{document}
{\HUGE HUGE text}
{\ssmall Can you see a ‘‘ssmall’’ text?}
\end{document}
```

HUGE text Can you see a "ssmall" text?

2.2 Not enough?

There is an alternative, completely flexible approach. The `anyfontsize` package scales the closest bigger or smaller font size available to any size.

The usage is very similar to the inline example shown before. The package implements the `\fontsize` command which takes two arguments, the new font size and the size of the `baselineskip`.

`\fontsize{`⟨*size*⟩`}{`⟨*baselineskip*⟩`}`

It is recommended to use a `baselineskip` of roughly 1.2 × font size in order to get a reasonable space between two successive lines. Of course the best value depends on the document and font design.

The following example shows font sizes `50pt` and `5pt` and compares them with `\Huge` and `\tiny`. The difference between `5pt` and `\tiny` (`6pt` for the `11pt` class option) is barely visible.

```
\documentclass[11pt]{report}
\usepackage{mathptmx}
\usepackage{anyfontsize}
\usepackage{t1enc}
\begin{document}
{\fontsize{50}{60}\selectfont Foo}
  {\fontsize{5}{6}\selectfont bar!}
{\Huge Foo}
  {\tiny bar!}
\end{document}
```

Foo bar! Foo bar!

Again, this only works with a scalable, non-standard font.

2.3 Memoir classes

As with font size class options, the memoir class also provides additional font modifier macros at the extreme ends of the scale, `\minuscule` and `\HUGE`. These macros use font sizes of `4pt`, `20pt` respectively, for the `9pt` class option and `20pt`, `132pt` respectively, for the `60pt` class option. Usage is exactly the same as for the standard LaTeX classes.

◇ Thomas Thurnherr
texblog (at) gmail dot com
http://texblog.org

The `calculator` and `calculus` packages: Arithmetic and functional calculations inside LaTeX

Robert Fuster

Abstract

The `calculator` package allows us to use LaTeX as a calculator, with which we can perform many of the common scientific calculations (with the limitation in accuracy imposed by the TeX arithmetic). The `calculus` package uses `calculator` to compute simultaneously a function and its derivative.

1 Introduction

The packages presented in section 2 define several commands to realize calculations within a LaTeX document. The `calculator` package introduces several new instructions that allow you to calculate with integer and real numbers using LaTeX. As well as add, subtract, multiply and divide, `calculator` computes powers, square roots, exponentials, logarithms, trigonometric and hyperbolic functions, and performs usual operations with integer numbers such that integer division (quotient and modulo), greatest common divisor, fraction simplification, ... In addition, the `calculator` package supports some elementary calculations with vectors in two and three dimensions and with 2×2 and 3×3 square matrices.

The `calculus` package adds to the `calculator` package several utilities to use and define elementary real functions and their derivatives, including operations with functions, polar coordinates and vector-valued functions.

Several packages can realize some arithmetic operations in TeX and LaTeX, but as far as I know, only the `calculus` package has the ability to calculate derivatives.

These two packages are designed to perform the calculations needed in the package `xpicture` (Fuster, 2012b), so the precision it gets is usually sufficient. But if we see TeX as a programming language, why should not we use it to make our calculations? In fact, we can use the `calculator` package as a length calculator, an alternative to the `calc` package; and, as another possible application, we can calculate and print the value of a mathematical expression, without using any external application. In this sense, these packages are appropriate if high precision is not required.

In section 3 we review some packages offering similar functionality to `calculator`. Section 4 describes the main algorithms used by `calculator` and, finally, in section 5 we explain our conclusions and future improvements of these packages.

2 The `calculator` and `calculus` packages

The `calculator` package defines a large set of commands intended to use LaTeX as a scientific calculator. Its companion package, `calculus`, gives us some tools to define, manipulate and operate with functions and derivatives. These packages are available, together, from CTAN (Fuster, 2012a).

2.1 `calculator`

This package operates with *numbers* (at least from the standpoint of the user),[1] not lengths, as is usual within other packages. The operations implemented by the `calculator` package include routines for assignment of variables, arithmetical calculations with real and integer numbers, two and three-dimensional vector and matrix arithmetic and the computation of square roots, trigonometrical, exponential, logarithmic and hyperbolic functions. In addition, some important numbers, such as $\sqrt{2}$, π and e, are predefined.

The names of all these commands are spelled in capital letters (with a very few exceptions) and, in general, they all need two or more mandatory arguments, one (or more) of which is a number and one (or more) the name of a command where results will be stored, as shown in the examples below.[2]

The new commands defined in this way work in any LaTeX mode.

For example, this instruction

```
\MAX{3}{5}{\solution}
```

stores `5` in the command `\solution`. Similarly,

```
\FRACTIONSIMPLIFY
   {10}{12}{\numerator}{\denominator}
```

defines `\numerator` and `\denominator` as `5` and `6`, respectively. Moreover, some of these commands support a first optional argument.

The *data* arguments need not be explicit numbers; they may also consist of commands expanding to a number. This allows us to chain several calculations, as in the following example:

Example 1

$$\frac{2.5^2}{\sqrt{12}} + \mathrm{e}^{3.4} = 31.7685$$

```
% store 2.5^2 in \tempA
 \SQUARE{2.5}{\tempA}
```

[1] Internally, numbers are converted into lengths, but a user need not be aware of this.

[2] Logically, the control sequences that represent special numbers (such as `\numberPI`) do not need any argument.

```
% store sqrt(12) in \tempB
 \SQUAREROOT{12}{\tempB}

% store e^3.4 in \tempC
 \EXP{3.4}{\tempC}

% \division:=\tempA/tempB
 \DIVIDE{\tempA}{\tempB}{\division}

% \sol:=\division+\tempC
 \ADD{\division}{\tempC}{\sol}

% round to 4 decimal places
 \ROUND[4]{\sol}{\sol}

\[
 \frac{2.5^2}{\sqrt{12}}+\mathrm{e}^{3.4}
      =\sol
\]
```

It does not matter if the *results* arguments are previously defined. But these commands act as declarations, so the scope is local.

The `calculator` and `calculus` user manual, embedded as usual in the source file `calculator.dtx` and also accessible on CTAN as `calculator.pdf`, describes all the commands in that package. We include below an overview and (incomplete) summary.

2.1.1 Predefined numbers

A few numbers are predefined: π and some of its multiples and divisors, the square roots of the first natural numbers (2, 3 and 5), e, 1/e, e^2 and $1/\mathrm{e}^2$, the useful cosines of $\pi/6$ and $\pi/4$ (or 30^{o} and 45^{o}), the golden ratio and its inverse, the logarithm of 10, and assorted others. Every predefined number is directly accessible calling the command `\number`*`XXX`*, where *XXX* is a reasonable name of the number (for example, `\numberPI`, `\numberSQRTTWO`, `\numberE`, `\numberCOSXXX` or `\numberGOLD`).

The choice of these numbers is obviously arbitrary, but you can define any number, either directly, using the command `\COPY`,

```
\COPY{12.56637}{\numberFOURPI}
```

or as the result of an operation

```
\SQUAREROOT{7}{\numberSQRTSEVEN}
```

You can use any admissible command name in place of `\numberSQRTSEVEN`.

2.1.2 Real arithmetic

The four basic operations are implemented as `\ADD`, `\SUBTRACT`, `\MULTIPLY` and `\DIVIDE`. As a special case, with the `\LENGTHDIVIDE` command we can divide two lengths and obtain a number.

Example 2 *One inch equals 2.54 centimeters.*

```
\LENGTHDIVIDE{1in}{1cm}\sol
One inch equals \sol{} centimeters.
```

Other implemented operations include integer powers, maximum and minimum of two numbers, absolute value, integer and fractional parts, truncation and rounding.

Example 3

$$\sqrt{2}+\sqrt{3}\approx 3.1463$$

```
\ADD{\numberSQRTTWO}
    {\numberSQRTTHREE}
    {\temp}
\ROUND[4]{\temp}{\sol}
\[
   \sqrt{2}+\sqrt{3}\approx\sol
\]
```

2.1.3 Integer numbers

We can compute the integer quotient and remainder of integer division, greatest common divisor and least common multiple of two numbers, and the irreducible fraction equivalent to a given fraction.

Example 4

$$\frac{4255}{4830}=\frac{37}{42}$$

```
\FRACTIONSIMPLIFY{4255}{4830}{\num}{\div}
\[
   \frac{4255}{4830}=\frac{\num}{\div}
\]
```

2.1.4 Elementary functions

The following real functions are defined: square root, exponential and logarithm, trigonometric (sine, cosine, tangent and cotangent) and hyperbolic (hyperbolic sine, cosine, tangent and cotangent).

Example 5

$$\mathrm{e}^2\cos\pi/3=3.69453$$

```
\EXP{2}{\exptwo}
\COS{\numberTHIRDPI}{\costhirdpi}
\MULTIPLY{\exptwo}{\costhirdpi}{\sol}
\[
   \mathrm{e}^2\cos \pi/3=\sol
\]
```

The exponential and logarithm functions admit bases other than e. Trigonometric functions allow radians or degrees as arguments (and also an arbitrary number of circle divisions).

Example 6

$$\log_{10}2=0.30103\quad\cos 72=0.309$$

```
\LOG[10]{2}{\logtwo}
\DEGREESCOS{72}{\cosseventytwo}
\[
   \log_{10}2=\logtwo\quad
   \cos 72=\cosseventytwo
\]
```

2.1.5 Vectors and matrices

This package operates only with two and three-dimensional vectors and 2×2 and 3×3 square matrices.

Within that limitation, it can add and subtract two vectors, compute the scalar product of two vectors, scalar-vector product, the norm of a vector, normalized vectors and absolute value (in each entry) of a vector.

Example 7

$$\|(1,2,-2)\| = 3$$

```
\VECTORNORM(1,2,-2)\sol
\[
   \left\|(1,2,-2)\right\|=\sol
\]
```

With matrices, the implemented operations are addition, subtraction, matrix product, scalar-matrix product, matrix-vector product, transposition, determinant, inverse matrix, absolute value (in each entry) of a matrix, and solution of a linear (square) system.

The `bmatrix` environment in the following example requires the `amsmath` package.

Example 8

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & 1 & 1 \\ 3 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 0 & 0.33333 \\ 1 & -1 & -1 \\ -1 & 2 & 1.66666 \end{bmatrix}$$

```
\INVERSEMATRIX(1, 2, 1;
              -2, 1, 1;
               3, 0, 0)(\aUU,\aUD,\aUT;
                        \aDU,\aDD,\aDT;
                        \aTU,\aTD,\aTT)
\[
\begin{bmatrix}
   1 & 2 & 1 \\ -2 & 1 & 1 \\ 3 & 0 & 0
\end{bmatrix}^{-1}=
\begin{bmatrix}
   \aUU & \aUD & \aUT \\
   \aDU & \aDD & \aDT \\
   \aTU & \aTD & \aTT
\end{bmatrix}
\]
```

2.2 `calculus`

The `calculus` package computes simultaneously the values of an elementary function and its derivative. It includes some predefined functions and diverse tools to define new functions, operating with either pre-existing functions or programming the required operations. Moreover, we can also define vector-valued functions and, in particular, curves referring to polar coordinates.

Example 9 *If* $f(t) = \cos t$, *then*

$$f(\pi/4) = 0.7071 \quad f'(\pi/4) = -0.70709$$

```
\COSfunction{\numberQUARTERPI}
            {\cosine}{\Dcosine}
If $f(t)=\cos t$, then
\[
   f(\pi/4)=\cosine\quad
   f'(\pi/4)=\Dcosine
\]
```

For each *function* defined here, you must use the following syntax:

> \function{*num*}{\cmd1}{\cmd2}

where *num* is a number (or a command expanding to a number), and \cmd1 and \cmd2 two control sequences where the values of the function and its derivative (in this number) will be stored.

2.2.1 Predefined functions

The following functions are defined:

- zero ($f(t) = 0$) and one ($f(t) = 1$) constant functions,
- identity ($f(t) = t$),
- reciprocal ($f(t) = 1/t$),
- square ($f(t) = t^2$),
- cube ($f(t) = t^3$),
- square root ($f(t) = \sqrt{t}$),
- exponential ($f(t) = \exp t$),
- logarithm ($f(t) = \log t$),
- trigonometric ($f(t) = \sin t$, $f(t) = \cos t$, $f(t) = \tan t$ and $f(t) = \cot t$),
- hyperbolic ($f(t) = \sinh t$, $f(t) = \cosh t$, $f(t) = \tanh t$ and $f(t) = \coth t$),
- and the Heaviside function ($f(t) = 0$, if $t < 0$; $f(t) = 1$, if $t \geq 0$).

All these functions can be used as in example 9.

2.2.2 Operating with functions

The easiest way to define new functions is to perform some operation with already-defined functions. Available operations allow us to define constant functions, to add, subtract, multiply or divide two functions, scale variable or function, raise a function to an integer power, compose two functions and make a linear combination of two functions.

Example 10 *If* $f(t) = (1 + \cos t)^2$, *then*

$$f(\pi/3) = 2.25 \text{ and } f'(\pi/3) = -2.59804$$

```
% g(t)=1+cos(t)
\SUMfunction
    {\ONEfunction}{\COSfunction}
    {\gfunction}

% F(t)=g(t)^2
\COMPOSITIONfunction
    {\SQUAREfunction}{\gfunction}
    {\Ffunction}

% sol=F(pi/3), Dsol=F'(pi/3)
\Ffunction{\numberTHIRDPI}{\sol}{\Dsol}

\noindent If $f(t)=(1+\cos t)^2$,
then $f(\pi/3)=\sol$ and $f'(\pi/3)=\Dsol$.
```

2.2.3 Polynomials and low-level function definition

Although the polynomials can be defined as linear combinations of powers, `calculus` includes some commands to directly define linear, quadratic, and cubic polynomials. For example, we can define the polynomial $p(t) = 2 - 3t^2$ by typing

```
\newqpoly{\mypoly}{2}{0}{-3}
```

Also, low-level commands exist to define a function by programming it and its derivative.

2.2.4 Vector-valued functions and polar coordinates

A vector-valued function can be identified with a pair of ordinary functions.[3] If the functions `\Xfunct` and `\Yfunct` are already defined, then

```
\VECTORfunction{\Ffunct}{\Xfunct}{\Yfunct}
```

declares the new vector-valued function `\Ffunct` with component functions `\Xfunct` and `\Yfunct`. For example, we can define the function $f(t) = (t^2, t^3)$ by typing

```
\VECTORfunction{\Ffunction}
  {\SQUAREfunction}{\CUBEfunction}
```

The `xpicture` package uses vector-valued functions to plot parametrically defined curves.

In this respect, curves defined in polar coordinates are a particularly interesting case. To define the polar curve $\rho = f(\phi)$ (where ρ and ϕ are the polar radius and arc), the `calculus` package includes the command

```
\POLARfunction{\ffunction}{\Pfunction}
```

where `\ffunction` is an already defined function and `\Pfunction` is the new polar function. In the following example, we define the *five-petal* curve, $\rho = \cos 5\phi$.

Example 11 *The polar curve* $\rho = \cos 5\phi$ *passes through the point* $(0.24998, 1.73204)$. *At this point, its tangent vector is* $(0.43298, 3.99986)$.

```
\SCALEVARIABLEfunction{5}{\COSfunction}
                        {\myfunction}
\POLARfunction{\myfunction}{\FIVEROSE}
\FIVEROSE{\numberTHIRDPI}{\x}{\y}{\Dx}{\Dy}

\noindent The polar curve $\rho=\cos 5\phi$
passes through the point $(\x,\y)$. At this
point, its tangent vector is $(\Dx,\Dy)$.
```

3 Other arithmetic-related packages in TEX

The limitations of classic TEX arithmetic are well known (Knuth, 1990). In short, TEX can operate with integer numbers n restricted by the relation $|n| \le 2^{32} - 1$.[4] Noninteger arithmetic is performed on lengths via conversion to a whole number of scaled points (`sp`). The largest admissible length is $16383.99998\,\mathtt{pt} \approx 2^{14}\,\mathtt{pt} = 2^{30}\mathtt{sp}$ (a point equals 2^{16} scaled points). Therefore standard TEX can not manage real numbers greater than 16383.99998. Moreover, considering that the smaller length is one scaled point ($1\,\mathtt{sp} \approx 0.000015\,\mathtt{pt}$), TEX cannot distinguish between two lengths differing in less than 0.00002 points. With the standard TEX behavior, this is the maximum level of accuracy we can expect.

These restrictions are absolutely negligible if you consider the main aim of TEX: for fine typesetting of text documents, TEX arithmetic is more than sufficient. But for some questions (e.g., composing quality graphics in high definition) more exacting arithmetic is required. Either for this reason or to implement a more user friendly syntax, a few authors have worked on issues related to the TEX arithmetic, as we can see from a look at the literature or in packages on CTAN. Some representative examples:

- Probably the most widely used package in this matter is `calc` (Thorup, Jensen, and Rowley, 1998). This package introduces a friendly syntax for the end user, reimplementing the LATEX length and counter manipulating commands and adding the ability to manipulate counters and lengths with infix notation. Among other improvements, this package allows you to multiply and divide by real numbers.
- In fact, calculating divisions is a notable problem for a TEX user, because the `\divide` command only supports integer divisors. Claudio Beccari

[3] Only two-dimensional vector functions are defined.

[4] In other words, the absolute value of an integer must not be greater than 2147483647.

introduced a nice algorithm for division (Beccari, 2006) in his `curve2e` package, which was subsequently adopted by Gäßlein, Niepraschk, and Tkadlec (2011) in `pict2e`.

- For obvious geometrical reasons, we often need to calculate trigonometrical ratios; the `trig` package, distributed within the `graphics` bundle (Carlisle, 2005), solves this problem by approximating the sine function with the Taylor-McLaurin series.[5] Beccari, in `curve2e`, uses a continued fraction approximation for the tangent function.
- The `pgf/tikz` bundle (Tantau, 2010) includes a *mathematical engine* to compute calculations needed by `tikz` pictures, including real and integer operations. This engine may be used separately by loading the `pgfmath` package.

  For the calculations it can perform, this package is the closest to `calculator`. Table 1 compares the results obtained using `calculator` and `pgf` with some of the examples included in this document. The selection is obviously arbitrary, but it shows that performance of calculations in these two packages is similar.

Regarding the problem of precision, the `fp` package (Mehlich, 1999) extends the fixed-point arithmetic to the interval

$$[-999999999999999999.999999999999999999,\\ 999999999999999999.999999999999999999]$$

which is more than enough in many cases. The `fltpoint` package (Guthöhrlein, 2004) provides similar results.[6]

To deal with numbers of arbitrary magnitudes, however, fixed-point arithmetic is always limited. The future LaTeX3 (Mittelbach et al., 2009) will support a floating point arithmetic that is already available as a part of the experimental package `expl3` (The LaTeX3 Project, 2012).

The `pgf/tikz` system includes, as an alternative to its mathematical engine, a fixed point library (using `fp`) and also a floating point library, to work with very large or very small numbers.

Another approach to arithmetic problems is to call an external application to take care of the calculations; this option has the obvious advantage of avoiding the restrictions imposed by the TeX arithmetic, both in efficiency and accuracy. The `sagetex` package (Drake, 2009)[7] or the commercial product *Scientific WorkPlace*[8] are well-known examples of these ideas. In a more general way, LuaTeX (Hoekwater, 2009) integrates an interface between TeX and the programming language Lua (Ierusalimschy, Celes, and de Figueiredo, 2012).

4 The algorithms

`calculator` performs additions, subtractions and products using ordinary TeX arithmetic on lengths. Division is essentially identical to Beccari's algorithm (adapted to the `calculator` syntax) with only one small improvement: as in Beccari's approach, each decimal place is calculated by multiplying by 10 the last remainder and dividing it by the divisor; but when the remainder is greater than 1638.3, an overflow occurs, because 16383.99998 is the greatest number. So, instead, we multiply the divisor by 0.1.

To calculate square roots and logarithms, we use Newton's method: $\sqrt{x}$ is the limit of

$$x_0 = x, \quad x_{n+1} = x_n - \frac{x_n^2 - x}{2x_n}$$

and $\log x$ is the limit of

$$x_0 = x - 1, \quad x_{n+1} = x_n + \frac{x}{\mathrm{e}^{x_n}} - 1$$

(this iterative method is used for $x \in [1, \mathrm{e}^2]$; otherwise, the relation $\log xy = \log x + \log y$ is used).

Trigonometric and exponential functions use generalized continued fractions: trigonometric functions are computed by reduction to sine or tangent,

$$\sin x = \cfrac{x}{1 + \cfrac{x^2}{2\cdot 3 - x^2 + \cfrac{2\cdot 3x^2}{4\cdot 5 - x^2 + \cfrac{4\cdot 5x^2}{6\cdot 7 - x^2 + \cdots}}}}$$

$$\tan x = \cfrac{1}{\cfrac{1}{x} - \cfrac{1}{\cfrac{3}{x} - \cfrac{1}{\cfrac{5}{x} - \cfrac{1}{\cfrac{7}{x} - \cfrac{1}{\cfrac{9}{x} - \cfrac{1}{\cfrac{11}{x} - \cdots}}}}}}$$

(for $|x| < \pi/2$; otherwise, angles are reduced to this case). The exponential applies this approximation

[5] For small values of x, the seven-degree McLaurin polynomial of $\sin x$ is a very accurate approximation.

[6] Despite its name, this package works with fixed point arithmetic.

[7] Sage (The Sage Project, 2012) is a free, open-source math software that supports research and teaching in algebra, geometry, number theory, cryptography, numerical computation, and related areas. The `sagetex` package acts as an interface between Sage and LaTeX.

[8] *Scientific WorkPlace* (MacKichan Software Inc., 2012) is a LaTeX typesetting and computer algebra integrated system.

		`calculator`		`pgf`	
expression	`scilab`	result	rel. error	result	rel. error
$\dfrac{2.5^2}{\sqrt{12}}+\mathrm{e}^{3.4}$	31.76831964	31.76854	0.000007	31.76741	0.000029
$\sqrt{2}+\sqrt{3}$	3.14626437	3.14627	0.000002	3.14627	0.000002
$\mathrm{e}^2\cos\pi/3$	3.694528049	3.69453	0.000001	3.69435	0.000048
$\log_{10}2$	0.301029996	0.30103	0.000000	0.3001	0.003089
$\cos 72^{\mathrm{o}}$	0.309016994	0.309	0.000055	0.30902	0.000010

Table 1: Comparison between `calculator` and `pgf`. The second column shows results obtained with `scilab`; columns four and six contain the relative errors when using `calculator` and `pgf`.

(Lauschke, 2009):

$$\exp x \approx 1+\cfrac{2x}{2-x+\cfrac{x^2/6}{1+\cfrac{x^2/60}{1+\cfrac{x^2/140}{1+\cfrac{x^2/256}{1+\cfrac{x^2}{396}}}}}}$$

(for $-6 < x < 3$; otherwise, the relation $\exp(x+y) = (\exp x)(\exp y)$ is applied).

5 Conclusions and future work

The packages `calculator` and `calculus` were born as working tools for the `xpicture` package, and they provide sufficient accuracy for the requirements of this package.

The package `calculator` is also appropriate to do geometrical calculations related to page composition and, in general, this package can be used for any scientific calculation which does not require overmuch precision. In fact, the performance of this package is similar to those of other packages that use the arithmetic of TeX. Only high precision packages give good results in complex calculations.

With respect to the calculations it can make, the performance of `calculator` is similar to those of `pgfmath`; all other packages offer a very limited set of functions.

The `calculator` package uses the typical TeX syntax: calculations are performed by calling the `calculator` commands and results are stored in new commands. The `fp` package behaves similarly. Other packages, like `pgfmath` and experimental `expl3`, admit infix mathematical notation; this is a nice feature not supported by `calculator`.

The `calculus` package provides a user friendly method to define functions and simultaneously calculate the values of a function and its derivative. The accuracy of the calculations is related to the accuracy of the `calculator` package, so if this is improved, the `calculus` package will become a solid tool to evaluate functions and derivatives. No other package has the ability to calculate derivatives.

There are some improvements that we hope to add soon, such as the implementation of additional functions (the inverse trigonometric and hyperbolic functions, and maybe some boolean functions), the inclusion of more utilities related to the definition of polynomials (definition of polynomials of arbitrary degree, construction of polynomials from their roots, implementation of the interpolating polynomial), ...

On the other hand, to make these packages truly interesting, we will attack the issue of accuracy. We will study the possibility of incorporating the option to obtain more accurate results, perhaps by loading the `fp` package, or exploring the possibility of using floating point arithmetic.

References

Beccari, Claudio. "LaTeX 2_ε, `pict2e` and complex numbers". *TUGboat* **27**(2), 202–212, 2006.

Carlisle, D. P. "Packages in the 'graphics' bundle". Available from CTAN, `/macros/latex/required/graphics`, 2005.

Drake, Dan. "The SageTeX package". Available from CTAN, `/macros/latex/contrib/sagetex`, 2009.

Fuster, Robert. "The `calculator` and `calculus` packages: Use LaTeX as a scientific calculator". Available from CTAN, `/macros/latex/contrib/calculator`, 2012a.

Fuster, Robert. "The `xpicture` package: A LaTeX package intended to extend the picture environment". `http://www.upv.es/~rfuster/xpicture`, 2012b.

Gäßlein, Hubert, R. Niepraschk, and J. Tkadlec. "The `pict2e` package". Available from CTAN, `/macros/latex/contrib/pict2e`, 2011.

Guthöhrlein, Eckhart. "The fltpoint package". Available from CTAN, macros/latex/contrib/fltpoint/, 2004.

Hoekwater, Taco. "The LuaTEX program". Available from CTAN, /systems/luatex/base, 2009.

Ierusalimschy, Roberto, W. Celes, and L. H. de Figueiredo. "Lua: The programming language". http://www.lua.org, 2012.

Knuth, Donald E. *The TEXbook*. Addison Wesley, Reading, Massachussets, 1990.

Lauschke, Andreas. "Convergence Acceleration with Canonical Contractions of Continued Fractions". http://216.80.120.13:8080/webMathematica/LC/general.jsp, 2009.

MacKichan Software Inc. "Scientific WorkPlace. The integration of LATEX and Computer Algebra". http://www.mackichan.com, 2012.

Mehlich, Michael. "The fp Package". Available from CTAN, /macros/latex/contrib/fp, 1999.

Mittelbach, Frank, R. Schöpf, C. Rowley, D. Carlisle, J. Braams, R. Fairbairns, T. Lotze, W. Robertson, J. Wright, and B. Le Floch. "The LATEX3 Project". http://www.latex-project.org/latex3.html, 2009.

Tantau, Till. "The Ti*k*Z and PGF Packages. Manual for version 2.10". http://sourceforge.net/projects/pgf. Also available from CTAN, /graphics/pgf, 2010.

The LATEX3 Project. "The expl3 package and LATEX3 programming". Available from CTAN, /macros/latex/contrib/expl3, 2012.

The Sage Project. "Sage: Open source mathematics software". http://www.sagemath.org, 2012.

Thorup, Kresten Krab, F. Jensen, and C. Rowley. "The calc package. Infix notation arithmetic in LATEX". Available from CTAN, as part of the tools bundle, /macros/latex/required/tools, 1998.

◇ Robert Fuster
Universitat Politècnica de València
Departament de Matemàtica Aplicada
Camí de Vera, 14
València E46022
Spain
rfuster (at) mat dot upv dot es
http://www.upv.es/~rfuster/

The xtemplate package: An example

Clemens Niederberger

Abstract

One of the most important points in the development of LaTeX3 is — roughly speaking — the separation of implementation, layout design and user interface. The package xtemplate connects the first two — it is part of the *Designer Interface Foundation Layer*. This article tries to demonstrate the idea behind the package and its usage with a (not necessarily practical) example.

1 Introduction

Not too long ago I had a first look at the xtemplate package which is part of the l3packages bundle.[1] After I understood the idea behind it I was immediately excited. So: what idea am I talking about?

The underlying structure for LaTeX3 has been discussed, for instance, by Frank Mittelbach at the TUG 2011 conference [1]. Of course I'm not going to repeat that.[2] An important part — if not the main idea — is the strict separation of different so-called layers. I'm confident you've already heard about the *Core Programming Layer* — `expl3`.

The xtemplate is part of a different layer, the *Designer Interface Foundation Layer*. So principally it is directed at package and class authors but I believe it will also play a big role in a LaTeX3 kernel, at least conceptually. The idea behind it isn't new, as one can read in "New Interfaces for LaTeX Class Design" (1999) [2].

Roughly speaking, the idea is this: a class has to provide a suitable design for different objects, such as section headings or lists. Preferably this would be done via a simple interface that would allow authors to adjust certain parameters to their own wishes or requirements. In other words (from xtemplate's documentation [4]):

1. semantic elements such as the ideas of sections and lists;
2. a set of design solutions for representing these elements visually;
3. specific variations for these designs that represent the elements in the document.

One should be able to determine the number and possibly the kind of arguments from both the definition and the interface.

xtemplate now allows one to declare objects, and so-called templates for these objects. For every object instances can be defined (figure 1). The user interface is then defined with the help of xparse [3].[3]

I for my part learn best through examples and I'm guessing I am not alone with that. So, I am going to demonstrate the concept and the different commands using an example that is not necessarily a practical application of xtemplate. Inspired by a question on `tex.stackexchange.com` [5], I will declare an object `names` and two templates `fullname` and `initial`. Declaring the instances will then show how flexible and easily extendable the concept is.

In the end, the code

```
\name{Jack Skellington} \par
\name[init-it-rev]{Jack Skellington}
```

will give:

Jack Skellington
Skellington, *J.*

A small warning: if you're not familiar with `expl3` — the "programming language" of LaTeX3 — details of the code might seem cryptic. But I will keep the example short so you should be able to follow the important parts.

2 The important commands

Basically there are four commands that are important for the definition of the structures:

1. `\DeclareObjectType`
   {⟨*object*⟩}
   {⟨*number of args*⟩}
2. `\DeclareTemplateInterface`
   {⟨*object*⟩}
   {⟨*template*⟩}
   {⟨*number of args*⟩}
   {⟨*interface*⟩}
3. `\DeclareTemplateCode`
   {⟨*object*⟩}
   {⟨*template*⟩}
   {⟨*number of args*⟩}
   {⟨*parameter*⟩}
   {⟨*code*⟩}
4. `\DeclareInstance`
   {⟨*object*⟩}
   {⟨*instance*⟩}
   {⟨*template*⟩}
   {⟨*parameter*⟩}

The first command, `\DeclareObjectType`, declares the object and specifies how many arguments it gets.

Then the object can be specified with the second command, `\DeclareTemplateInterface`. More precisely an interface is declared, i.e., the number

[1] From directory `macros/latex/contrib/l3packages`

[2] I wouldn't be qualified anyway.

[3] Also part of the l3packages bundle.

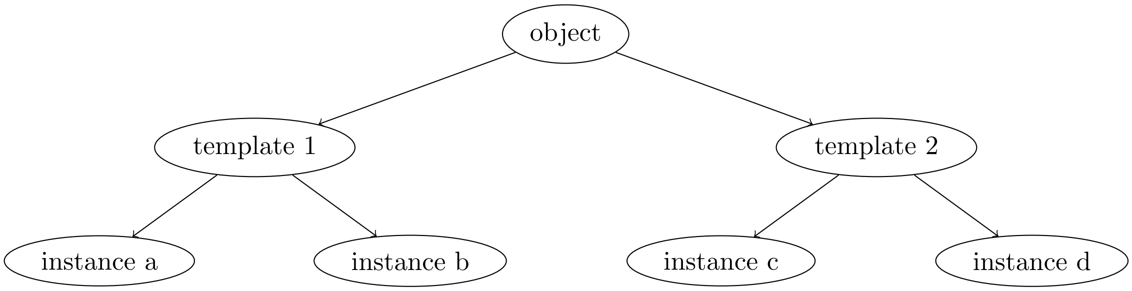


Figure 1: Schematic figure of the relationships between object, templates and instances.

and type of parameters are declared with which the template can be customized later.

The third command, \DeclareTemplateCode, is where the actual definitions are made. The parameters defined in the interface get variables assigned, and code is defined that determines the behaviour of the template.

The fourth command, \DeclareInstance, at last defines an instance that instantiates a template with concrete values for the parameters. Each of these instances can then be used in the user command with \UseInstance.

3 An example

Now let's consider an actual example.

3.1 The object

The first thing to do is to think about the basic interface. The user command of our object names is going to be \name, with one argument taking the lastname and firstname separated with a space. At a level deeper, though, we want to handle lastname and firstname as two separate arguments. Thus the object is going to get two arguments:

```
\usepackage{xtemplate,xparse}
% we use expl3, so activate the expl3 namespace
\ExplSyntaxOn
% #1: lastname, #2: firstname
\DeclareObjectType { names } { 2 }
```

The next thing to do is to specify the templates.

3.2 The templates

Templates are declared for an existing object. First the interface has to be specified. The number of arguments of the template and a possible list of parameters has to be declared. Every parameter is given a certain type and can get a default value.

```
% the interface for the template ‘fullname’:
\DeclareTemplateInterface{names}{fullname}{2}
{
  reversed           : boolean   = false ,
  use-last-name      : boolean   = true  ,
  use-first-name     : boolean   = true  ,
  last-name-format   : tokenlist         ,
  first-name-format  : tokenlist         ,
}
% the interface for the template ‘initial’:
\DeclareTemplateInterface{names}{initial}{2}
{
  reversed           : boolean   = false ,
  use-last-name      : boolean   = true  ,
  use-first-name     : boolean   = true  ,
  last-name-format   : tokenlist         ,
  first-name-format  : tokenlist         ,
  last-name-initial  : boolean   = false ,
  first-name-initial : boolean   = true  ,
}
```

The parameters that are defined here can easily be extended and are determined by the type of object and of course the desired degree of complexity. Here we have just a few to demonstrate the concept.

After the interfaces for the templates have been declared the actual code can be defined. Let's start with fullname. We're going to need suitable variables or functions for the parameters. In the ⟨*code*⟩ part \AssignTemplateKeys will define them with actual values and activate them.

Our code now only tests the values of the boolean variables and writes the names in the given order. The solution is not the most elegant one but will do for this demonstration:

```
% variables first:
\bool_new:N \l_names_reversed_bool
\bool_new:N \l_names_use_last_bool
\bool_new:N \l_names_use_first_bool
\tl_new:N   \l_names_last_format_tl
\tl_new:N   \l_names_first_format_tl

% now the template code:
% #1: lastname, #2: firstname
\DeclareTemplateCode {names} {fullname} {2}
{
  reversed          = \l_names_reversed_bool   ,
  use-last-name     = \l_names_use_last_bool   ,
  use-first-name    = \l_names_use_first_bool  ,
  last-name-format  = \l_names_last_format_tl  ,
  first-name-format = \l_names_first_format_tl,
}
{
  \AssignTemplateKeys
  \bool_if:NTF \l_names_reversed_bool
  {
    \bool_if:NT \l_names_use_last_bool
      {{\tl_use:N \l_names_last_format_tl #2}}
    \bool_if:nT
      {
        \l_names_use_first_bool &&
        \l_names_use_last_bool
      }
      {,~}
    \bool_if:NT \l_names_use_first_bool
      {{\tl_use:N \l_names_first_format_tl #1}}
  }
  {
    \bool_if:NT \l_names_use_first_bool
      {{\tl_use:N \l_names_first_format_tl #1}}
    \bool_if:nT
      {
        \l_names_use_first_bool &&
        \l_names_use_last_bool
      }
      {\tl_use:N \c_space_tl}
    \bool_if:NT \l_names_use_last_bool
      {{\tl_use:N \l_names_last_format_tl #2}}
  }
}
```

We can reuse most of the variables for the template `initial` but we're going to need a helper function that gets all but the initial of a name. The code is going to become a little bigger. Of course it could be written in a more elegant way but again, this will suffice for our demonstration purposes.

```
% two additional variables:
\bool_new:N \l_names_last_initial_bool
\bool_new:N \l_names_first_initial_bool

% helper function:
\cs_new:Npn \names_get_initial:w #1#2\q_stop
  {#1.}

% the template code:
% #1: lastname, #2: firstname
\DeclareTemplateCode {names}{initial}{2}
{
  reversed           = \l_names_reversed_bool      ,
  use-last-name      = \l_names_use_last_bool      ,
  use-first-name     = \l_names_use_first_bool     ,
  last-name-format   = \l_names_last_format_tl     ,
  first-name-format  = \l_names_first_format_tl    ,
  last-name-initial  = \l_names_last_initial_bool  ,
  first-name-initial = \l_names_first_initial_bool,
}
{
  \AssignTemplateKeys
  \bool_if:NTF \l_names_reversed_bool
  {
    \bool_if:NT \l_names_use_last_bool
      {
        \group_begin:
          \tl_use:N \l_names_last_format_tl
          \bool_if:NTF
            \l_names_last_initial_bool
            {\names_get_initial:w #2\q_stop}
            {#2}
        \group_end:
      }
    \bool_if:nT
      {
        \l_names_use_first_bool &&
        \l_names_use_last_bool
      }
      {,~}
    \bool_if:NT \l_names_use_first_bool
      {
        \group_begin:
          \tl_use:N \l_names_first_format_tl
          \bool_if:NTF
            \l_names_first_initial_bool
            {\names_get_initial:w #1\q_stop}
            {#1}
        \group_end:
      }
  }
  {
    \bool_if:NT \l_names_use_first_bool
      {
        \group_begin:
          \tl_use:N \l_names_first_format_tl
          \bool_if:NTF
            \l_names_first_initial_bool
            {\names_get_initial:w #1\q_stop}
            {#1}
        \group_end:
      }
    \bool_if:nT
      {
        \l_names_use_first_bool &&
        \l_names_use_last_bool
      }
      {\tl_use:N \c_space_tl}
    \bool_if:NT \l_names_use_last_bool
      {
        \group_begin:
          \tl_use:N \l_names_last_format_tl
          \bool_if:NTF
            \l_names_last_initial_bool
            {\names_get_initial:w #2 \q_stop}
            {#2}
        \group_end:
      }
  }
}
```

We're nearly there. For every template we need to declare at least one instance. And of course we need to define the user command.

3.3 The instances

Declaring the instances is not complicated at all. You choose the template and assign values to the parameters. Here we will make three instances for each template:

```
% a few instances, starting with ‘fullname’:
\DeclareInstance {names}{standard}{fullname}{}
\DeclareInstance {names}{it-rev}{fullname}
  {
    first-name-format = \itshape ,
    reversed          = true
  }
\DeclareInstance {names}{first-only}{fullname}
  {use-last-name = false}
% and now ‘initial’:
\DeclareInstance {names}{init-first}{initial}{}
\DeclareInstance {names}{init-it-rev}{initial}
  {
    first-name-format = \itshape ,
    reversed          = true
  }
\DeclareInstance {names} init-all}{initial}
  {last-name-initial = true}
```

Defining more instances wouldn't be any problem. With every additional instance the user command we're going to define next would get another option.

3.4 The user command

For the definition of this command we're going to use the package xparse. This makes it easy to define the wanted argument input: lastname and firstname separated with a blank space.

The command is going to get an optional argument with which the instance to be used can be specified. It should test if the chosen instance exists and if not, use the `standard` instance. Of course it could also raise a warning or an error.

```
% yet more variables:
\tl_new:N  \l_names_instance_tl
\tl_set:Nn \l_names_instance_tl { standard }

% the internal command:
\cs_new:Npn \names_typeset_name:nnn #1#2#3
  {
    \IfInstanceExistTF {names} {#1}
      { \UseInstance {names} {#1} }
      { \UseInstance {names} {standard} }
    {#2} {#3}
  }
\cs_generate_variant:Nn
  \names_typeset_name:nnn {V}

% the user command:
\DeclareDocumentCommand \name
  {o>{\SplitArgument{1}{~}}m}
  {
    \group_begin:
      \IfNoValueF {#1}
        {\tl_set:Nn \l_names_instance_tl {#1}}
      \names_typeset_name:Vnn
        \l_names_instance_tl #2
    \group_end:
  }
\ExplSyntaxOff
```

Now we're ready to comprehend the examples from the beginning (and a few others):

```
\name{Jack Skellington} \par
\name[it-rev]{Jack Skellington} \par
\name[first-only]{Jack Skellington} \par
\name[init-first]{Jack Skellington} \par
\name[init-it-rev]{Jack Skellington} \par
\name[init-all]{Jack Skellington}
```

And the output:

Jack Skellington
Skellington, *Jack*
Jack
J. Skellington
Skellington, *J.*
J. S.

4 Before the end

I hope this little excursion can provide a first insight into the functionality of xtemplate. My own knowledge is not much deeper. In my opinion the idea behind xtemplate has a great future and I am excited to see how it will be used in LaTeX3.

References

[1] Frank Mittelbach. LaTeX3 architecture and current work in progress, 2011. `http://river-valley.tv/latex3-architecture-and-current-work-in-progress`.

[2] Frank Mittelbach, David Carlisle, and Chris Rowley. New interfaces for LaTeX class design. 1999. `http://www.latex-project.org/papers/tug99.pdf`.

[3] The LaTeX3 Project. The xparse package. Available from CTAN, `macros/latex/contrib/l3packages/xparse.dtx`.

[4] The LaTeX3 Project. The xtemplate package. Available from CTAN, `macros/latex/contrib/l3packages/xtemplate.dtx`.

[5] Emit Taste. Macro for formatting names (initials or full name). `http://tex.stackexchange.com/q/57636/5049`, 2012.

◇ Clemens Niederberger
Am Burgrain 3
71083 Herrenberg
Germany
`contact (at) mychemistry dot eu`
`http://www.mychemistry.eu/`

A patent application design flow in LaTeX and LyX

Peter J. Pupalaikis

Abstract

I describe a design flow for beautiful and consistently formatted U.S. patent applications using our favorite typesetting tools LaTeX and LyX along with a newly-created LaTeX class file and an accompanying LyX layout file.

1 Introduction

The patent process is often thought of as a thing of mystery cloaked in arcane terminology and rules. It is also thought to be very expensive. This is surprising because these thoughts are often held by inventors who are performing in highly technical fields where research itself is expensive. It turns out that the patent process is actually fairly straightforward and can be relatively inexpensive when the inventor takes an active role. It is helpful to stay grounded in some simple concepts:

- The patent process is one that preferably ends with a document describing certain intellectual property rights.
- The property rights obtained are the rights to exclude others from practicing the *claims*.
- In exchange for these rights, an inventor is obligated to disclose the best way he knows how to practice an invention.
- The document containing the disclosure is the patent application.
- In order to obtain patent protection, an invention must be novel, unique, and useful.

Some information that many are surprised to know:

- Anyone may file a patent application.
- With sufficient effort and narrowing of claims scope, a patent can almost always be obtained.
- The actual protection granted by a patent is discovered during litigation.

I have made some rather broad statements here. While anyone can file an application, it's advisable to get help from an attorney. The process of narrowing the scope of claims during prosecution can provide for a patent with very limited usefulness to the patent owner. Finally, a poorly written application can lead to poor protection and as I said, the quality of the protection is found out too late.

I have been inventing and patenting for about twenty years now. When I started out, I knew next to nothing about the patent process. The process (at medium to large companies) from an inventor's perspective is to fill out a *patent disclosure form.* This is a form that describes the invention and is used to supply necessary information to a patent attorney. These forms are very common at big companies. Then, a few months after supplying this information and a few discussions with an attorney, a patent application arrives. That's the theory anyway. I found my early patenting experiences very difficult and unsatisfying. It was a painful process of transmission of highly technical information from inventor to an (albeit technically trained) attorney and legal information from attorney to inventor. Over time, I found that the choices with this process are between expensive, time consuming and low (or at least unclear) quality on the one hand and very expensive, very time consuming with reasonable quality on the other.

My goal, therefore, was to lower expense and effort and improve quality. This led to the creation of patent application drafts and culminated in the creation of tools for improving this flow. Today, I write all of my patent applications myself in a form ready for filing. My application is not filed until a patent attorney has cleaned it up and rewritten the claims after many discussions. I've discovered that getting a higher quality patent disclosure into the hands of a patent attorney is the right way to ensure that money spent on legal fees is spent wisely. Working with me, the attorney concentrates on using her time and legal knowledge to get me the right protection for the right price.

No matter your opinions, training or systems for drafting patent applications, it is clear that the process benefits extremely from improved tools for producing the application itself. Improved tools can solve many of the mechanical problems with patent drafting and can ensure consistently generated applications. This article is about how LaTeX and its more user-friendly cousin LyX can be used to streamline the patent application process.

2 Patent writing tools

After my first frustrating experiences of rewriting patent applications, my first attempts to improve the process began with efforts at better patent disclosures using well-known word processing tools from well-known software companies. Using these tools, I would provide a disclosure filled out under the same headings as the patent application. Over time, I learned the intent of each section of the application and after each patent was filed, I compared what I provided to what ended up in the attorney-generated application and honed my skills; this culminated in the Knuthian leap of controlling the final output and properly formatting and typesetting the final application. This is where I ran into difficult tool problems. Let me outline some of them:

1. The major problem — drawings! The patent office has rather arcane rules about drawings and specific standards for drawings. One requirement is that drawings contain elements that are *annotated* (i.e. a numbered arrow or line on the drawing pointing to an element) where the drawing-element name and accompanying number are used within the patent application to refer to the element. Despite looking bad, this is a cross-referencing nightmare as no tools (other than LaTeX, eventually) could be made to use variable names that get replaced with numbers in both the drawings and the patent specification.
2. Drawing and drawing-element name and number agreement. The problem is that the drawing-element name and the number must be consistent. If I refer to widget [23] in my specification, there had better be a drawing with a line pointing to the widget element and it had better be numbered with 23. Keeping the name of the element consistent and matching the number was a problem. Also, it would be nice if the next time I refer to element 23, I call it the same thing.
3. Figure numbers. There is one section in an application where the drawings are to be described *in order*. Most cross-referencing tools get the number right, but cannot place the drawing description in the correct order and cannot make the number agree with the description.

Other problems include:

1. The formatting capabilities of various word processors, believe it or not, are too flexible. One can edit the application, change the fonts, styles, headers, whatever one wants. They're flexible while missing key features.
2. Have you ever tried to get drawings in the right place using the world's most popular word processor? Enough said.
3. Consistency of claim language and support in the specification.

I was finding that in some cases it was nearly impossible to control the format of the final application the way I wanted. When I saw the paralegal at my company cut out a drawing I made and tape it onto a sheet of paper to get the drawings right, I determined that enough was enough. I set out to solve these cross-referencing and formatting problems.

All of the above problems benefit from a programmatic approach; an approach with variables and macros that can be used for cross-referencing, consistency of language, and consistency of format.

```
\documentclass[english]{uspatent}
...all front-matter definitions ...
\include{Drawings}% figure information
\maketitle
\patentSection{Field of the Invention}
\patentParagraph text ...
\patentSection{Cross Reference to Related
   Applications}
\patentParagraph text ...
\patentSection{Background of the Invention}
\patentParagraph text ...
\patentSection{Objects of the Invention}
\patentParagraph text ...
\patentSection{Summary of the Invention}
\patentParagraph text ...
\patentDrawingDescriptions
\patentSection{Detailed Description of the
   Preferred Embodiments}
\patentParagraph text ...
\patentClaimsStart
...
\patentClaimsEnd
\patentSection{Abstract}
\patentParagraph text ...
\patentDrawings
\end{document}
```

Figure 1: LaTeX Patent Application Structure

3 Patent writing in LaTeX

Anyone reading this *TUGboat* article is likely to be acquainted with LaTeX, so I'll dive right in. The solutions to some of the previously mentioned problems are solved through the incorporation of various macros and page settings into a LaTeX class file `uspatent.cls`. The class finally developed and published is based on the `memoir` class. The declaration of this class appears at the top of the LaTeX patent application file as shown in figure 1. Here you can see the layout of the document, which includes:

- various front-matter material. These are macros which define the title, inventor name, assignee information, patent attorney information, etc. These assignments are used in header, footer and title page creation.
- inclusion of the drawings file (which I will talk about in a bit).
- various sections of the patent which will be properly formatted via the `\patentSection` macro and where each paragraph is preceded by the `\patentParagraph` macro that causes the paragraphs to be numbered.

```
\figureDefinition{VisioDrawing}
\figureExtension{pdf}
\figureDescription{example drawing made in Visio}
\annotationDefinition{Widget}
\annotationName{widget}
\annotationDescription{a widget in the drawing}
\figureDefinition{TpXDrawing}
\figureExtension{tpx}
\figureCaption{PRIOR ART}
\figureDescription{example drawing made in TpX}
\annotationDefinition{input}
\annotationName{input}
\annotationDescription{the input}
\annotationDefinition{output}
\annotationName{output}
\annotationDescription{the output}
\annotationDefinition{mathProcessor}
\annotationName{math processor}
\annotationDescription{math processor on left}
```

Figure 2: Drawing definitions in LaTeX

- a patent claims area (which is actually just an enumerated environment).
- the \patentDrawingDescriptions macro that automatically emits the description of the drawings.
- the \patentDrawings macro that automatically emits the drawing pages.

The drawings file contains a list of drawings and annotations, as shown in figure 2. Here you see a list of macros that will define everything about the figures and annotations. Each figure's information is listed, in order starting with a \figureDefinition macro that defines the name of the file and how it is referenced from within the application. It is assigned a number from an internal counter. This is followed by a \figureExtension macro that defines its extension and the \figureDescription macro which provides a concise description. Usually patent drawings do not contain captions, but one can be optionally provided using the \figureCaption macro. You might be wondering why I didn't use a multiple argument macro. I found it difficult to remember the order of the arguments but more importantly, macros which have multiple arguments cannot be used within LyX.

After each figure definition its annotations are listed, each one beginning with a call to the macro \annotationDefinition. This specifies the name used to reference the annotation within the document. The macro also assigns a unique annotation number from an internal counter. Each annotation definition is followed by calls to \annotationName and \annotationDescription. The name is the word associated with the element that you want printed in your document when you refer to it. The description is a longer description that helps find the annotation in the drawing and is used with drawing packages that cannot make use of LaTeX cross-referencing. I will explain its use further when I discuss the Annotation List section.

The figure and annotation definitions can appear inline if you want, but it gets very long and bothersome to see all these definitions when you are editing the application so it's better to include them in a separate file. After the drawings file is processed, it enables the following features:

- the figures and annotations can be referred to using several macros:
  - \referencePatentFigure expands to the formatted patent figure (e.g. FIG 2.).
  - \annotateWithName is the normal way to reference the annotations in the application and maintain agreement between element name and number. It expands to the annotation name and formatted annotation number joined (e.g. widget [3]).
  - \annotate expands to the formatted annotation number (e.g. [3]). It is used when the name and number would appear awkward (e.g. . . . is connected to the two widgets [3] and [4]).
  - \annotationNumberReference is used in drawings to point to the drawing elements. LaTeX friendly drawing formats (like Ti*k*Z) can be made to draw the correct number pointing to a drawing element.
- The \patentDrawingDescriptions macro expands into a section that automatically prints a section that looks like:

  > **Brief Description of the Drawings**
  >
  > For a more complete understanding of the invention, reference is made to the following description and accompanying drawings, in which:
  > FIG. 1 is aaaa;
  > . . .
  > FIG. x is bbbb; and
  > FIG. y is cccc.

  Here the figure's descriptions are the exact text in the descriptions provided, match the figure numbers in order, and are even punctuated properly according to how the patent office would like to see them.
- The \patentDrawings macro expands into a section that brings in all of the drawings and places them appropriately on numbered pages. If the \annotationNumberReference macro is used and the drawings have been produced with a LaTeX friendly tool (like TpX) and/or in a

. . . front-matter. . .

Include: Drawings.lyx

Field of the Invention

text . . .

Cross Reference to Related Applications

text . . .

Background of the Invention

text . . .

Objects of the Invention

text . . .

Summary of the Invention

text . . .

—BRIEF DESCRIPTION OF THE DRAWINGS—

Detailed Description of the Preferred Embodiments

text . . .

—START OF PATENT CLAIMS—

. . .

—END OF PATENT CLAIMS—

Abstract

text . . .

—PATENT DRAWINGS—

Figure 3: LyX Patent Application Structure

LaTeX friendly format (like Ti*k*Z), then the annotations are also numbered properly. I also include a printing mode switch via the macro `\setPrintingModeDraft`. Among other things, this macro causes an Annotation List section to precede the drawing pages. This section is not meant for filing; it lists all of the figures, their names, annotations, and their names and descriptions. This is so that they can be matched up with the drawings to ensure that the right numbers are in the right place. I also provided this for users who use drawing tools that are not LaTeX friendly and do not allow the referencing of the annotation with the `\annotationNumberReference` macro. These users must manually place the numbers on the drawing using the information in the annotation list after finishing writing the application.

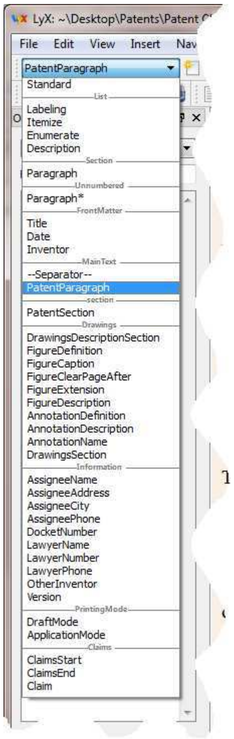


Figure 4: LyX Custom Environments

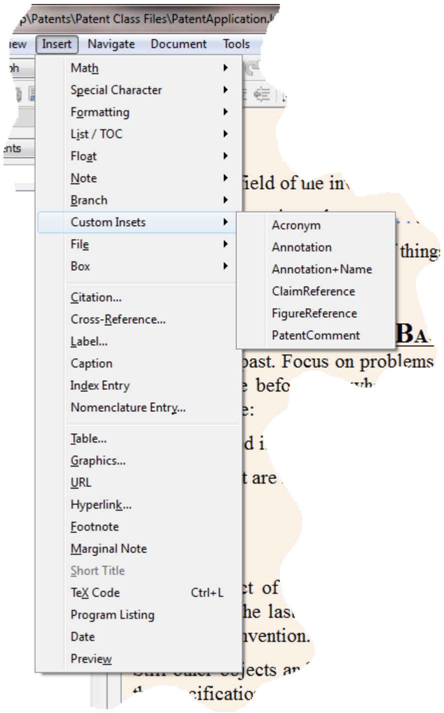


Figure 5: LyX Custom Insets

```
\def\annotationDefinition#1{%
 \expandafter\ifx\csname anonum#1 \endcsname\relax
 \global\advance\@annotationnumber by 1
 \expandafter\edef\csname anoele \the\@annotationnumber\endcsname{#1}%
 \expandafter\edef\csname anonum#1 \endcsname{\the\@annotationnumber}%
 \expandafter\edef\csname anofignum \the\@annotationnumber\endcsname{\the\@annotationfigurenumber}%
 \else % error handling here
 \fi}
\def\annotationDescription#1{\expandafter\def\csname anodesc \the\@annotationnumber\endcsname{#1}}
\def\annotationName#1{\expandafter\def\csname anotext \the\@annotationnumber\endcsname{#1}}
\def\annotationReference#1{[\thinspace\annotationNumberReference{#1}\thinspace]}
\def\annotationNameAndReference#1{\annotationTextReference{#1}~\annotationReference{#1}}
\def\annotationDescriptionReference#1{\csname anodesc \annotationNumberReference{#1}\endcsname}
\def\annotationTextReference#1{\csname anotext \annotationNumberReference{#1}\endcsname}
\def\annotationNumberReference#1{\csname anonum#1 \endcsname}
\def\annotationListVariableName#1{\csname anoele #1\endcsname}
\def\annotationListText#1{\csname anotext #1\endcsname}
\def\annotationListDescription#1{\csname anodesc #1\endcsname}
\def\annotationListFigureNumber#1{\csname anofignum #1\endcsname}
```

Figure 6: Annotation Macros in LaTeX

4 Patent writing in LyX

LyX has been written about in many *TUGboat* articles, but briefly, it is a front-end for LaTeX. LyX is a WYSIWYM editor, i.e. "What You See Is What You Mean". LyX takes the guesswork out of what equations will look like in the end, removes some of the verbosity of LaTeX from the user, and can be used to limit and target the formatting capability that the user has (he can always enter raw LaTeX in the end, so the full capability of LaTeX is never completely out of reach).

With a properly written layout file, LyX attempts to mimic to some degree what the document will mostly look like. With the `uspatent.layout` file that goes along with the `uspatent.cls` LaTeX class file, it provides the patent writing capability through the use of *environments* and *custom insets*. A typical LyX patent application looks much like the analogous LaTeX application, only the formatting mimics that of the final PDF and the user can choose not to see any LaTeX code, as shown in figure 3. After the LaTeX explanation, I think this figure does not need much explanation.

The environments are accessed in LyX using a drop-down box in the upper left corner of the editor. The customized environment list for patent applications is shown in figure 4. Here we see all of the environments for the front-matter, patent sections, patent paragraphs, claims and the drawings and annotations. Mostly, the use of these environments saves on typing the LaTeX macros and shows some sort of indication on the screen that the macro arguments are in that environment. Even more important, it restricts the user's choice of formatting.

The custom insets are how all of the references are made to claims, figures and annotations as previously described. This is shown in figure 5. These put small boxes inline with the text that can be expanded to show the arguments. Here you see the addition of an Acronym custom inset (which uses the LaTeX `acro` package) which is also useful in patent applications.

5 A few TeX tricks

The main tricks used in dealing with the figures and annotations are based on the use of `\csname`.[1] I want to explain the mechanics here because they might be useful for solving other similar problems. Please refer to the TeX listing in figure 6.

First, a brief summary of these macros:

- The first three macros are *Assignment* macros: `\annotationDefinition`, `\annotationDescription`, and `\annotationName` refer to the definition of an annotation.
- The next five macros are *Referencing* macros and are suffixed by "Reference": `\annotationReference`, `\annotationNameAndReference`, `\annotationDescriptionReference`, `\annotationTextReference`, and `\annotationNumberReference` provide reference of an annotation by a *VarName*.
- The last four macros are *List* macros and are prefixed by "annotationList":

[1] Amy Hendrickson, "The wonders of `\csname`", TUG 2012, Boston MA; *TUGboat* 33:2, pp. 219–224, `http://tug.org/TUGboat/33-2/tb104hendrickson.pdf`.

`\annotationListVariableName`, `\annotationListText`, `\annotationListDescription`, and `\annotationListFigureNumber` provide reference to an annotation by *Number*. The reason they are called *List* macros is because they are most useful for generating an annotation list.

The annotation is defined by first using the macro `\annotationDefinition`, where the argument supplied becomes the *VarName* for the annotation. In figure 6 you can see that an annotation counter is advanced and two new control sequences are defined. One control sequence is *anoele* followed by the annotation counter value. It is assigned to *VarName*. The other control sequence is *anonum* followed by *VarName*. It is assigned to the annotation counter value. These two assignments allow one to obtain the *VarName* given an annotation number and to obtain the annotation number given the *VarName*. The key *List* and *Referencing* macros are such that:

`\annotationListVariableName{`*Number*`}` = *VarName*
`\annotationNumberReference{`*VarName*`}` = *Number*

The macro `\annotationDescription` defines a control sequence named `anodesc` followed by the annotation counter value. Similarly, `\annotationName` defines a control sequence `anotext` followed by the annotation counter value. In this way, given an annotation number, it is easy to find the *VarName*, and the description and text name associated with an annotation through the use of the *List* macros provided. When we generate the annotation list in the draft mode patent application, we simply loop over the annotation numbers and list all of this information.

The *Referencing* macros are used within the patent application and within drawings. They are used to reference the annotation number, a formatted number (e.g. [*Number*]) and the annotation text that goes with that number. The annotation text is an interesting example because the `\annotationName` associated the text with the annotation number and we are referencing this text with the *VarName*. If you examine the `\annotationTextReference` macro, you see that the `\annotationNumberReference` macro is used to obtain the *Number* associated with the *VarName* supplied and this *Number* is used in conjunction with the associated *anotext* to form the control sequence that defines the text. In this manner, any of the annotation control sequences that define the annotation can be formed from either the annotation *Number* or *VarName*.

6 Summary and future plans

I've presented a method, using the `uspatent.cls` file in LaTeX and the `uspatent.layout` file in LyX, for typesetting patent applications that produce consistent and good looking results. Look for these files along with a patent writing guide on CTAN at `http://ctan.org/pkg/uspatent`. There are a number of things I'd like to improve in future versions. Some of these are:

- Usage of the `aux` file so that the figure and annotation definitions can go anywhere in the document (currently they need to go at the top before they are referenced).
- Elimination of dependence on the `memoir` class. The `memoir` class is great but heavy-handed for this application. I make use of very little of this class's capability and know that I can remove it and make use of a few smaller packages.
- I'd like to become more adept at my usage of `\csname`, which I still haven't quite mastered.
- I'd like to add better error handling and undefined reference handling capability so that users don't get a cryptic (LA)TeX error message when things fail. Currently, if an undefined annotation is referred to, it simply doesn't print; this is obviously a poor way to handle this type of error.
- I was too lazy to figure out how to parse the figure names to extract the extension.
- I'd like to eventually develop an offline piece of software for managing drawings and annotations. While the tools provided here fix some of the most difficult problems and are a step in the right direction, I know of many more ways to improve the patent writing process. A small step in this direction would be a drawing package that can import a PDF and, through a graphical interface, add TikZ code that draws a line pointing to drawing elements and the macro that expands to the element number.

This has been a fun and fruitful experience and is my first contribution to CTAN and the LaTeX community. I hope to make many more in the future!

Happy TeX patenting!

◇ Peter J. Pupalaikis
Ramsey, NJ USA
pete_pope (at) hotmail dot com
http://mysite.verizon.net/
petepope/id6.html

Page style tricks with scrpage2

Markus Kohm

Abstract

For the fourth edition of the KOMA-Script book I've used a special style for the page header, in which the page number is separated by a small black bar from the running head and moved out into the margin. This was accomplished by using the KOMA-Script package scrpage2. Although the KOMA-Script book was created using a KOMA-Script class, use of the scrpage2 package is not limited to those classes. It may also be used with several other classes, e.g., the standard classes.

1 Issue

Kohm and Morawski (2012) was produced using a layout with a long, narrow text column and a relatively wide, and heavily used, column for margin notes. For this design, the default page style of KOMA-Script would have been problematical. On the one hand, the page header with running head aligned to the outer edge of the text area would have accented the text column too much. On the other hand, the page number, if aligned to the outer edge of the text area at the foot of the page, would have seemed lost in space; this might even have caused the page proportions to seem wrong.

It would usually be suggested to extend the page header across the entire text and margin note columns, aligning the running head text to the outer edge of the margin note column. Similarly, the footer should also extend across both columns. But on pages with few margin notes, the page number at the outer edge of the margin note column would look lost. To avoid squeezing the page number into the lower outside corner of the page, the bottom margin would have to be very large.

To avoid such problems, the page header and footer are often separated from the rest of the page by horizontal lines. These rules form a kind of frame. Personally I don't like this, because such frames optically constrict the page body. Also, the book already contains several elements with frame effects, e.g., tables, syntax frames, figures of pages, and all of those may be close to the page head or foot. Such a concentration of horizontal lines is best avoided.

Because of this, I've decided not to use horizontal lines below the head or above the foot. The foot should in any case remain empty, to avoid the problem with the page numbers. So the pagination must be somewhere in the head. But where? Moving the page number from the foot to the head only moved the problem.

I reject page numbers at the inner margin. I think that the design of a page layout is not only about the appearance, but also about utility. If one is searching for a page based on the table of contents or an index, or even a cross reference, the page number is the object of the search. Placing it in the inner margin or the gutter makes it nearly useless; somewhere in the outer margin is much better, so one doesn't need to open the book completely to find the number. Then the search will be much faster.

2 Solution

The final choice was a classic design with the title on the inside, and the page number on the outside of the running head. Because of the prominent border for margin notes, heavily used also for figures and tables or their captions, the page number was aligned not at the edge of the text, but at the outer edge of the margin note column.

Of course, the margin note column isn't always occupied; on some pages this column is empty, and sometimes the main text is even indented on the left. On such pages, the page number may seem a bit lost. However, as mentioned earlier, I didn't want to accentuate the header with a horizontal rule. Instead, I introduced a somewhat heavier vertical rule to the inside of the page number, as shown in figure 1.

To implement this heading, first the package scrpage2 is loaded, and the page style scrheadings activated. This page style is configured to set the running head text flush to the inner edge of the main text area. In addition, the page number is aligned to the outer edge of the margin note column, even on plain pages, which will be used, e.g., for starting new chapters. The footer is configured to be empty:

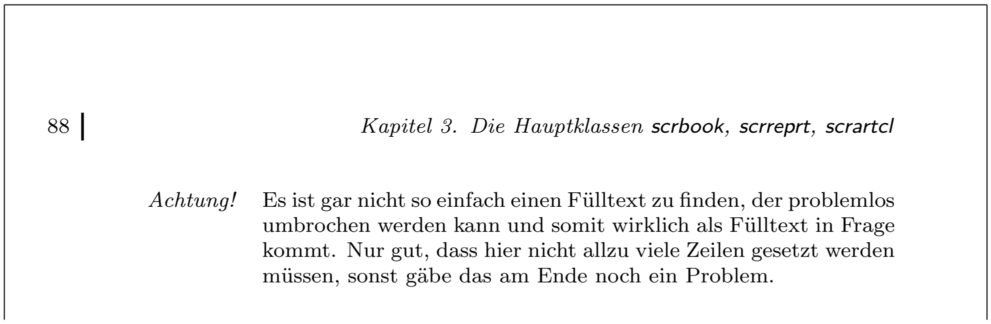
```
\usepackage{scrpage2}
\pagestyle{scrheadings}
\clearscrheadfoot
\ihead{\headmark}
\ohead[\pagemark]{\pagemark}
```

Note that scrpage2 uses the optional argument of commands like \ohead to configure the page style scrplain, the plain page style used together with scrheadings.

After this, the pagination routine is redefined to add the vertical rule beside the page number, distinguishing between odd and even pages:

```
\renewcommand*{\pagemark}{%
  \usekomafont{pagenumber}%
```

Editor's note: First published in *Die TEXnische Komödie* 3/2012, pp. 17–21; translation by the author and Barbara Beeton.



88 | *Kapitel 3. Die Hauptklassen* *scrbook, scrreprt, scrartcl*

Achtung! Es ist gar nicht so einfach einen Fülltext zu finden, der problemlos umbrochen werden kann und somit wirklich als Fülltext in Frage kommt. Nur gut, dass hier nicht allzu viele Zeilen gesetzt werden müssen, sonst gäbe das am Ende noch ein Problem.

Figure 1: Page head used in Kohm and Morawski (2012)

```
  \ifodd\value{page}\pnumbar\enskip\fi
  \thepage
  \ifodd\value{page}\else\enskip\pnumbar\fi
}%
\newcommand*{\pnumbar}{%
  \raisebox{0pt}[\ht\strutbox][\dp\strutbox]{%
    \rule[-\dp\strutbox]
         {1.5pt}
         {1.1\baselineskip}%
  }%
}
```

So printing the rule beside the page number isn't part of the definition of the page style itself, but part of the definition of the pagination macro `\pagemark`. The height, alignment, and width of the rule were determined by experiment. The rule has been set to the height and depth of a normal text line using `\raisebox`; this ensures proper alignment with the page number and running head text.

Last but not least the header has been enlarged to span not only the text area but also the margin note column. To make sure that recalculation of the page area will also adjust the width of the header, the `\AfterCalculatingTypearea` hook of the typearea package has been used, with the command `\recalctypearea` activating the adjustment.

```
\AfterCalculatingTypearea{%
  \setheadwidth[0pt]{%
    \dimexpr\textwidth
            +\marginparsep
            +\marginparwidth\relax
  }%
}%
\recalctypearea
```

If you don't use the typearea package from KOMA-Script, you may omit `\AfterCalculatingTypearea` and `\recalctypearea`; just use `\setheadwidth` directly and avoid further changes to the page area.

3 Usage with standard page areas

I've already mentioned that Kohm and Morawski (2012) uses a special page area with a wide margin note column. Optically this will be part of the type area and of the margin. But what about when using a standard layout without margin notes or only some small margin notes? Would the same page header definition blend well in this case? See figure 2 for an example.

In my opinion, the distance from the page number to the left edge of the text area is too large. An alternative would be to align the page number at the inner edge of the margin column instead of the outer edge. Figure 3 shows an example of this suggestion.

Only a small change to the `\pagemark` command is needed to implement this alternative:

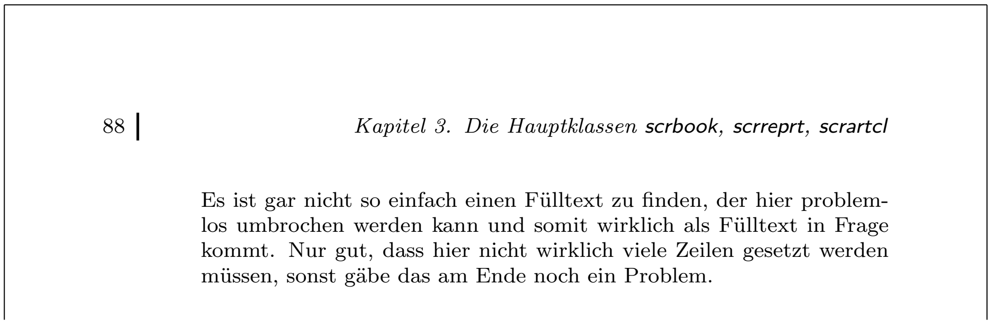
```
\renewcommand*{\pagemark}{%
  \usekomafont{pagenumber}%
  \ifodd\value{page}%
    \makebox[\marginparwidth][l]
            {\pnumbar\enskip\thepage}%
  \else
    \makebox[\marginparwidth][r]
            {\thepage\enskip\pnumbar}%
  \fi
}
```

4 More modifications

The page head shown is suitable for double-sided printing. For single-sided printing the distinction between odd and even pages may be dropped; in this case only the variant for right-hand (odd) pages is needed.

The running head can be aligned not only to the inner edge of the text area but alternatively to its outer edge. To do so, remove the `\ihead` commands from the first listing and extend the `\ohead` command appropriately. Alternatively you may substitute the commands `\rohead` and `\rehead`:

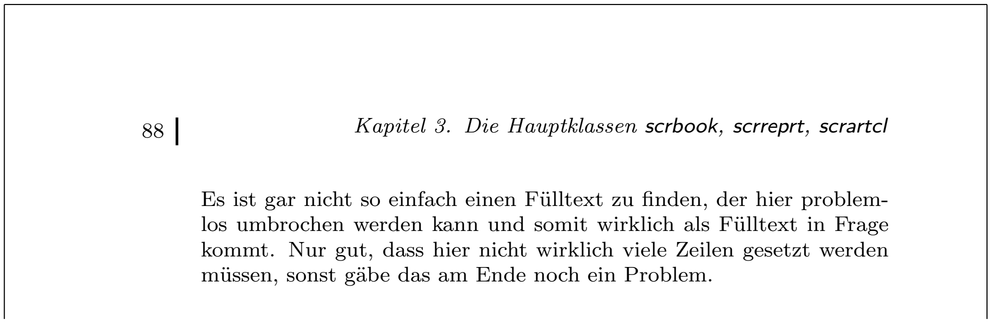
```
\lehead[\pagemark]{%
```



88 | *Kapitel 3. Die Hauptklassen scrbook, scrreprt, scrartcl*

Es ist gar nicht so einfach einen Fülltext zu finden, der hier problemlos umbrochen werden kann und somit wirklich als Fülltext in Frage kommt. Nur gut, dass hier nicht wirklich viele Zeilen gesetzt werden müssen, sonst gäbe das am Ende noch ein Problem.

Figure 2: Page head of figure 1 in combination with a standard layout



88 | *Kapitel 3. Die Hauptklassen scrbook, scrreprt, scrartcl*

Es ist gar nicht so einfach einen Fülltext zu finden, der hier problemlos umbrochen werden kann und somit wirklich als Fülltext in Frage kommt. Nur gut, dass hier nicht wirklich viele Zeilen gesetzt werden müssen, sonst gäbe das am Ende noch ein Problem.

Figure 3: Alternative alignment of the page number with respect to figure 2 in combination with a standard layout

```
  \makebox[\marginparwidth][l]{\pagemark}%
  \hspace{\marginparsep}\headmark
}
\rohead[\pagemark]{%
  \headmark\hspace{\marginparsep}%
  \makebox[\marginparwidth][r]{\pagemark}%
}
```

If you want to change only the width, height or depth of the rule beside the page number, you just have to change the `\pnumbar` command; neither the pagination command nor the page header definition needs to be changed.

Of course, you may also move the page numer to the page footer as is the default for the KOMA-Script classes. To do so, you don't need to change the command `\pagemark`, but only the usage of this command in the configuration of the page styles.

5 Concluding remark

In this article you've been told how to use scrpage2 to alter the layout of the page header and footer. You've seen behind the curtain of making design decisions. An implementation of the desired design has been shown, one that has been used for a real book project.

Alternatives for different margin settings have been explicated and suggestions for implementing some of them have been given. Finally, some further possibilities have been offered for changing design or implementation.

References

Kohm, Markus, and J.-U. Morawski. *KOMA-Script*. Lehmanns Media, Berlin, 4th edition, 2012.

◇ Markus Kohm
Freiherr-von-Drais-Straße 66
68535 Edingen-Neckarhausen
Germany
`komascript (at) gmx dot info`
`http://www.komascript.de`

CrafTEX: Applying TEX, MetaPost, and friends in crafts

Mari Voipio

Everything started because of my job as a documentation manager in high-tech industry: when the word processor gave up on our big fat instruction manual and the purpose-built software wasn't within budget, we ended up with ConTEXt. The transition period wasn't easy, but in time I learned to appreciate this software that does *not* make assumptions about what I'm attempting to do.

Thus, when I suddenly found myself with a textile craft book to be translated and prepared for printing, I thought of ConTEXt. Life happened and the booklet is still sitting on my desk waiting for its turn, but in the meantime I have learned many other things about TEX-based systems and started to exploit their potential in crafts.

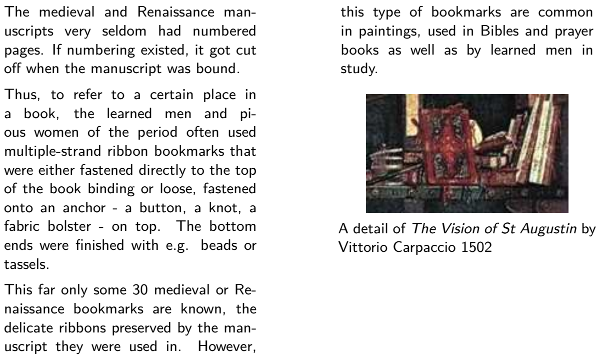
The experience has been mind-blowing! I come up with new implementations almost weekly, although I don't usually try things out until I have a real need for them. I am not a programmer, but I realize that a computer is especially useful in reducing the tedious repetitiveness of the planning stages. Nothing can ever prepare a crafter to what happens when you play with real yarn and real paper and glue, but the "what if that's lighter blue" and "I guess this is the wrong font here" process can be significantly sped up by computer simulation.

I don't feel complete until I've shared my knowledge with others. I don't get many face-to-face opportunities to do that, so I decided to go online: `http://www.lucet.fi`. I haven't had the energy to fight with WordPress about the printing styles, so instead I'm planning to do printer-friendly PDF instructions with ConTEXt and MetaPost.

Besides enhancing my creativity, I also use ConTEXt to deal with the boring but necessary parts of having my own little craft business, e.g. for creating price lists and business cards. This migration is still very much in process, but eventually everything will be done with ConTEXt and possibly MetaPost, with as few different style definitions as possible.

1 Plain vanilla typesetting

It may seem obvious to many readers that TEX can be used to typeset craft-related documents just as well as anything else. However, to me it has been a new world where vector graphics can be inserted as is and the layout is mainly limited by my own skill, not by what far-away programmers thought I'd need the software for. While shedding 20 years worth of WYSIWYG workarounds and reverse engineering has not been easy, I now find ConTEXt much more versatile than anything I've had before, doubling as a word processor and tool for desktop publishing.



The medieval and Renaissance manuscripts very seldom had numbered pages. If numbering existed, it got cut off when the manuscript was bound.

Thus, to refer to a certain place in a book, the learned men and pious women of the period often used multiple-strand ribbon bookmarks that were either fastened directly to the top of the book binding or loose, fastened onto an anchor - a button, a knot, a fabric bolster - on top. The bottom ends were finished with e.g. beads or tassels.

This far only some 30 medieval or Renaissance bookmarks are known, the delicate ribbons preserved by the manuscript they were used in. However, this type of bookmarks are common in paintings, used in Bibles and prayer books as well as by learned men in study.

A detail of *The Vision of St Augustin* by Vittorio Carpaccio 1502

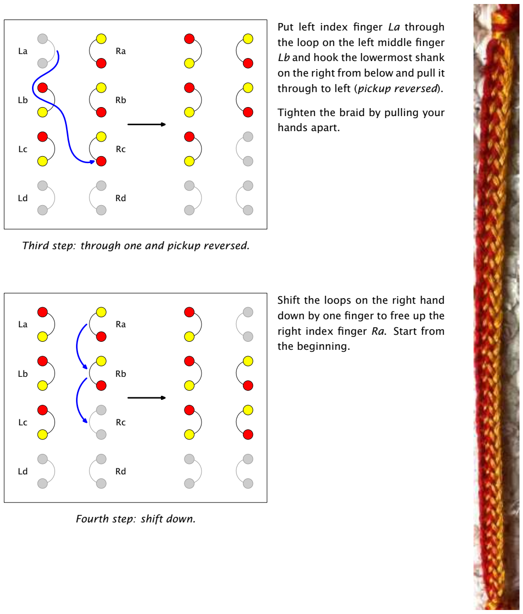
Figure 1: Documentation card for a historical bookmark.

Thus far I've used TEX, mainly ConTEXt, to typeset three kinds of documents: documentation on pieces I've created, instructions on how to do something, and little odds and ends necessary to my tiny craft business.

Documentation is an integrated part of my crafting. I feel that telling the story behind an item adds to its value. A hobby of mine is historical re-enactment, and there it is important to "justify" the existence of an item by explaining the historical context, usually with literary and visual references. Even a small item can illustrate history and bring it close, e.g. a bead-decorated multi-strand bookmark illuminates an era when page numbers were a rarity and books were stored flat rather than upright.

Instructions are all about sharing the knowledge one has gained. Instructions on paper can seldom fully replace hands-on training, but I always try to include both written instructions and graphics as different people prefer different versions. Nowadays I try to complement the instruction sheets with online videos, but I'd rather not count on the moving picture; living with slightly dodgy Internet connections has taught me to value downloadable instruction files!

The business end of my typesetting is the newest addition to my TEXing and also the one that saves me the most time: judicious use of layers and imposition suddenly makes perfect tags and labels much easier to achieve than it ever was with word processors. As I attend a variety of craft-related events, I used to have several different price lists, one "medieval", one "old-fashioned" and one "modern" (with man-made materials only appearing in the third one). Now I can fairly easily maintain just one price list from which



Put left index finger *La* through the loop on the left middle finger *Lb* and hook the lowermost shank on the right from below and pull it through to left (*pickup reversed*).

Tighten the braid by pulling your hands apart.

Third step: through one and pickup reversed.

Shift the loops on the right hand down by one finger to free up the right index finger *Ra*. Start from the beginning.

Fourth step: shift down.

Figure 2: An instruction page for a broad lace.

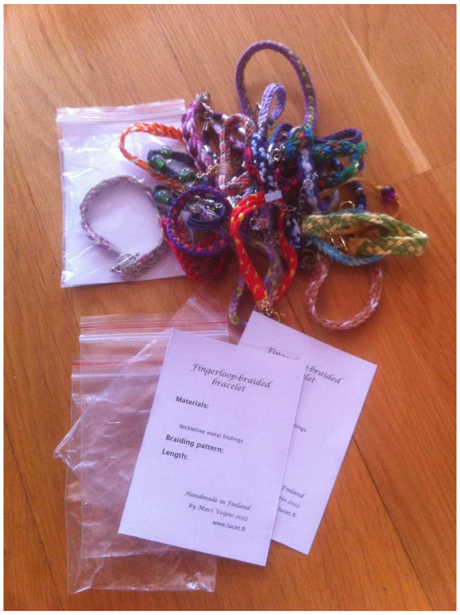


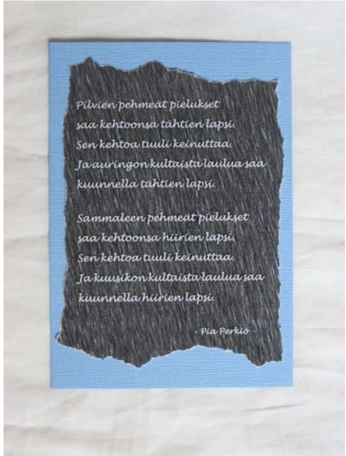
Figure 3: An example business order.

I pick whatever is needed for a given event, without dealing with (e.g.) tedious mail merge functions.

2 Scrapbooking and cardcraft

My handwriting was never great to start with, and I'm badly out of practice since switching to computers over 20 years ago. Thus, when I have a reason to make a greeting card, I resort to typesetting everything on computer, so I only need to sign the card after printing. I've managed to create these texts in a word processor or vector drawing program — depending on what was available — but ConTEXt has made the process much easier to control. This especially applies when the card is of unusual size, e.g. because I've messed up with a punched border and cut it off to save the rest of the card; changing the ConTEXt code to reflect this is much faster than fiddling with text boxes in a word processor file.

Sometimes it can be hard to come up with a suitable picture for a card. I wanted to express my condolences and support to a friend whose long-awaited baby was stillborn, but the available cards were either too somber and dark or otherwise inappropriate. In the end I resorted to the "poetry for all occasions" section in my bookshelf, and found a beautiful Finnish poem on a child resting on soft green moss, gently rocked by the wind, a poem that wasn't too "heavy" to remember this little one. When I had found the text, the rest of the card grew out of it: looking for and finding a (free) rainy background, trimming the text by tearing the edges, applying it with glue dots on blue card.



Pilvien pehmeät pielukset
saa kehtoonsa tähtien lapsi.
Sen kehtoa tuuli keinuttaa.
Ja auringon kultaista laulua saa
kuunnella tähtien lapsi.

Sammaleen pehmeät pielukset
saa kehtoonsa hiirien lapsi.
Sen kehtoa tuuli keinuttaa.
Ja kuusikon kultaista laulua saa
kuunnella hiirien lapsi.

- Pia Perkiö -

Figure 4: Condolences card.

I like using printed texts because I can get a new one if I mess it up in the assembly stage — but this card turned out to be one of those rare specimens where everything went right on the first go. As this was the first time I had tried using background texture, the learning curve was steep and bumpy as always, but definitely worth the effort. Not only was I content with how this card turned out (figure 4), I now have a template that can be tweaked to become any type of card with a background graphic and a text on top of it. Another poem, another background, a different edge treatment, e.g. a lace edge punch, and the card will look totally different with very little effort.

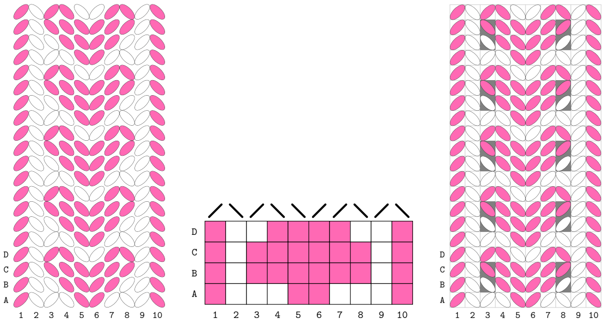


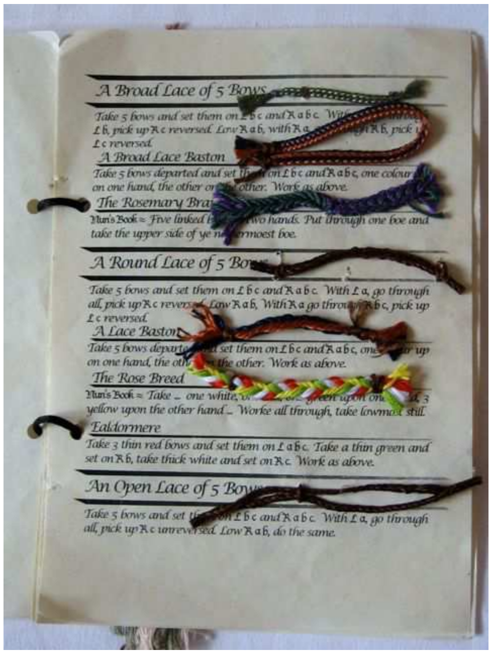
Figure 5: Tablet weaving of a heart braid pattern: simulation, threading chart, and turning sequence.

3 Tablet weaving

Tablet weaving or card weaving is an ancient technique for weaving patterned ribbons. It is a complicated technique due to the structure of the braid: the most common weaving tablet is a square with a hole for thread at each corner, and each card can be threaded either from the left or right, although all threads in a given card need to come from the same direction. During the weaving process the tablets are turned a quarter round away or toward the weaver to get a new shed (space for the weft), and one can either turn the whole pack together or split the tablets and turn them to different directions. All this makes for a fairly complex structure — and then there are additional techniques to further manipulate warp and weft threads.

When I originally learned tablet weaving, I was taught to plan my braids on squared paper. However, I soon found out that it was very easy to forget one of the variables, and those mistakes tended to be costly: it could take me three hours to thread a simple warp and start weaving, only to find out that the pattern wasn't at all what I wanted. I'm not the first one who thought about using computer as shortcut, and for several years I used open source software that had its features and limitations, but worked — only in Windows. When I changed over to Macs I had to come up with a different solution. For a while I used Inkscape to draw pattern simulations, but the oblique "stitch" turned out to be a bit hard to place accurately and in general I got tired of all the repetitive clicking and the work involved every time I wanted to change something. This was enough incentive to try to implement MetaPost instead.

For the simplest tablet weaving patterns I just make a simulation graphic for myself and follow it when threading the tablets. However, a threading chart makes the process of creating a warp a bit faster, especially if thread counts are included. For patterns where tablets turn separately, a turning chart is also needed. In mine I mark the deviating tablets with a darker background, as in the heart ribbon that was inspired by Valentine's Day (figure 5). I chose to weave this in evenly and tightly spun pearl cotton to ensure clean lines, but I was still surprised by how well it turned out compared with the simulation.



A Broad Lace of 5 Bows

Take 5 bows and set them on L b c and R a b c. With [illegible] L b, pick up R c reversed. Low R a b, with R a, [illegible] R b, pick [illegible] L c reversed.

A Broad Lace Baston

Take 5 bows departed and set [illegible] on L b c and R a b c, one colour [illegible] on one hand, the other o[illegible] other. Work as above.

The Rosemary Bra[illegible]

Nun's Book = Five linked [illegible] two hands. Put through one boe and take the upper side of ye n[illegible]rmoest boe.

A Round Lace of 5 Bo[illegible]

Take 5 bows and set them on L b c and R a b c. With L a, go through all, pick up R c revers[illegible] Low R a b, With R a go throu[illegible] R b c, pick up L c reversed.

A Lace Baston

Take 5 bows departe[illegible] set them on L b c and R a b c, one [illegible] up on one hand, the oth[illegible] the other. Work as above.

The Rose Breed

Nun's Book = Take – one white, [illegible] green upon on[illegible], 3 yellow upon the other hand – Worke all through, take lowmo[illegible] still.

Ealdormere

Take 3 thin red bows and set them on L a b c. Take a thin green and set on R b, take thick white and set on R c. Work as above.

An Open Lace of 5 Bo[illegible]

Take 5 bows and set t[illegible] on L b c and R a b c. With L a, go through all, pick up R c unreve[illegible]sed. Low R a b, do the same.

Figure 6: A page from my Helsinge Book.

4 Historical (re)creation

Another of my favorite braiding techniques is fingerloop braiding, where loops of yarn are held on the fingers of both hands (usually not the thumbs) and manipulated by switching them from finger to finger through or past each other. This technique is surprisingly well-documented in 15th and 17th century English manuscripts that have been transcribed by several researchers. By now I remember the simplest braiding sequences, but need cheat sheets for the more complicated ones. Because modern books are frowned on at some of the events I go to, I decided to create a Renaissance-style fingerloop braiding "recipe book" to show demonstration spectators and to jog my memory.

Thus, my "Helsinge Book of Braids and Laces" (named in period style after the place where it was created) is typeset in ConTEXt with the Lucida OpenType calligraphy and blackletter fonts, and printed on sturdy off-white parchment paper (figure 6). Sample braids are attached with a few sewing stitches through the paper. I chose this approach for two reasons: I wanted to make the book easy and cheap to replace if it gets rained on (for example); and I wanted to be able to add new braiding recipes anywhere in the book so I can keep related braids together instead of having to add them to the end of the book in the order I find or create them. Currently there are only six A5 pages of recipes in my printed copy and many samples are missing, but the project is constantly growing, spurred by the positive interest that my little booklet has received wherever I've shown it.

5 Other ideas

One idea leads to another, and the possibilities seem endless. In tablet weaving I need charts for double-faced weave and brocade, and a lot of automation to omit the most tedious stages of pattern planning. I could also use weaving drafts and especially simulation graphics for rigid heddle, inkle and backstrap weaving and even my small "real" weaving loom. I am also learning Japanese braiding techniques and recently found a book with beautiful patterns, but instruction charts that just don't parse for me, so I have to rework them using a system I can understand.

In other textile crafts I'm itching to try my hand at quilting, as MetaPost will be handy in playing with shapes and color. I should also be able to create cutting plans and even cutting patterns for many medieval and Renaissance items of clothing, another task I'm used to doing on gridded paper. As there are several common fabric widths and shrinkage depends on the fiber, the preliminary cutting plans often have to be remade after the fabric is purchased and washed; this should be at least less smudgy with MetaPost, if not less work.

I also want to learn to create interactive PDFs from my instruction sheets. When I've gotten more practice in integrating MetaPost in ConTEXt documents, I hope to be able to add all types of fancy frames around my texts and pictures as well. The combination of MetaPost and ConTEXt also allows for fitting text into shapes and clipping text and graphics to shapes.

Besides paper and fibre, I also play with beads. When I string a necklace, it typically has a pattern where two colors and/or shapes and/or sizes of beads are combined. I think this could be simulated to some extent by placing objects on an oval path of appropriate diameter or length. There are so many factors to be taken into account that MetaPost will never be able to substitute a real tactile beading board and a real strung necklace, but a quick simulation should show whether something will be pleasing to my eye and if so, what amounts of, say, 6 mm and 4 mm beads I need for the project. Again, playing with color is faster virtually; in real life, it takes a while to pick out all 50 red beads and put 50 blue in the vacated slots, while on the computer it is a matter of a quick round of find and replace — or replace all, if one is feeling courageous.

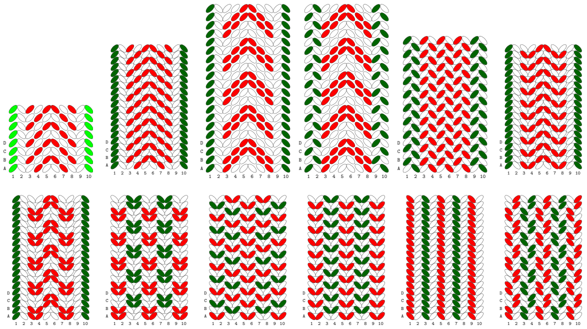


Figure 7: Testing a design theme and variations.

6 Why TEX and friends?

Recently my games with MetaPost and (Con)TEX(t) have attracted quite a bit of curiosity both within the ConTEXt community and among the braiders, so I often have to explain why I've made this choice. The main aim of the whole exercise is to shorten or entirely avoid the most repetitive and tedious stages of crafting to free up time and energy for creativity. If changing a color scheme on a braid takes three minutes instead of 30, I can try out a lot more color schemes in the same amount of time. When I go into that mode, I don't want technical issues to hold up the flow of new ideas. (See a design theme and variations in figure 7.)

I'm sure there is existing commercial or open source software for everything I use, and usually learning to use just that software would be faster and easier. The flip side of the coin is that dedicated software is (at best) good at doing what it was intended to do, but it can often be expensive, or restricted to just one operating system, or good at the basics but hopeless at an advanced level. As I'm interested in a wide variety of crafts, I need something that is as versatile as possible and the ConTEXt suite in combination with MetaPost and TEX/ConTEXt code ticks all the boxes for me. I hope to write more articles in the future going into more of the TEXnical and MetaPost-ish details.

◇ Mari Voipio
`mari dot voipio (at) lucet dot fi`
`http://www.lucet.fi`

MayaPS: Maya hieroglyphics with (LA)TEX

Bruno Delprat and Stepan Orevkov

Abstract

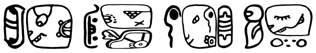
We present a system for hieroglyphical composition of ancient Maya texts, to be used for their palaeography, the production of dictionaries, epigraphical articles and textbooks. It is designed on the base of TEX and PostScript using the Dvips interface, and includes a set of Maya fonts.

The ancient Mayan writing system is very particular: base writing signs attach to each other from all four sides (left, right, top, bottom), and are also rotated and rescaled. This cannot be produced with TEX's usual tools.

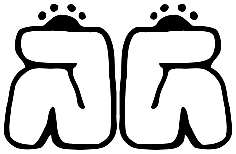
For example, we can type:

```
\maya{li.AM2 u.TUN/CHU uj.UJ.ki death.KIMI/la}
```

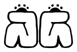
to obtain

 (Dresden codex).

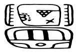
1 Introduction



The present package MayaPS is designed for editing the palaeography of ancient Maya hieroglyphical texts using TEX or LATEX *and* Dvips. The PhD dissertation [1] and a previously published Spanish language symposium communication [2] are typeset using it. MayaPS is available from `http://picard.ups-tlse.fr/~orevkov`.

To get the above Maya word  *xib* (male), we typed:

```
\mayaSize{2cm}\maya{422.422}
```

As another example, to get  *katun* (calendar cycle of 20 years), we input:

```
\maya{(023.153.023):220} \emph{katun} (calen...
```

2 Structure of the ancient Maya script

2.1 General principles

The ancient Maya logo-syllabic writing has been in use in Central America's Southern Mexico, Guatemala, Belize, Honduras and Costa Rica for more than 1300 years, from the 3rd century AD to the mid-16th century, when Spaniards forbade its use and burned Maya books on religious and political grounds.

This impressive civilization left rich inscriptions on monuments, ceramics and divinatory almanacs. They constitute nevertheless a small volume of available texts: three surviving manuscripts (the Dresden, Madrid and Paris codices) and about a thousand short inscriptions. Maya texts are now largely deciphered, with a variable degree of reliability.

The writing system comprises more than 500 base signs called glyphs. Since the end of the 19th century, Western scholars have set up catalogues of Maya hieroglyphics with different encoding numbers, the most popular being the Thompson Catalogue [11].

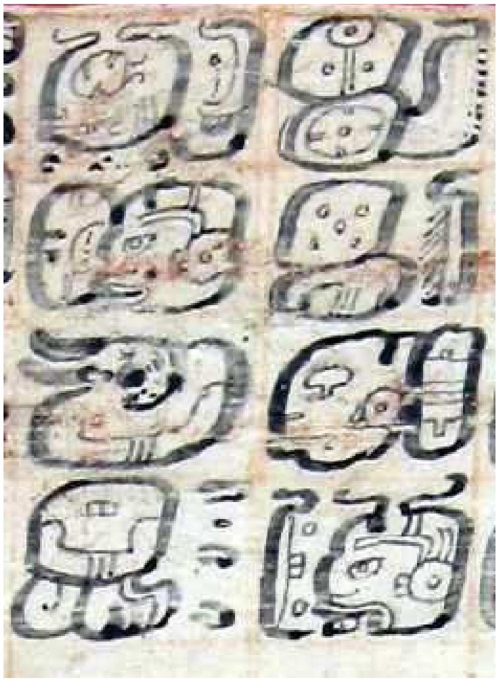
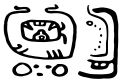
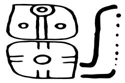
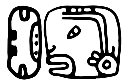
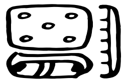
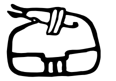
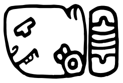
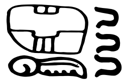
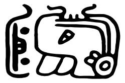


Figure 1: Dresden codex page 30b(2) & text palaeography with translation below

			
400/010.030	+176/204.031	117.260	133/111.023
tsel-ah	*lakin*	*chac-xib*	*kabil*
Was standing	East	red man	sweet
			
423/515	530.112	515/504.013	026.401
cehel-uah;	*Chac*	*hanal*	*u-bool*
deer tamal	god Chac	meal	its tribute

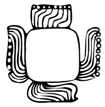
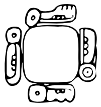
Ancient Maya words are composed by attaching together primitive (non-decomposable) glyphs, in a way rather similar to how Chinese characters are composed. Composed primitive glyphs are rescaled so that they harmoniously fill a rectangle of a fixed size (called a *cartouche* by Mayanists). The cartouches are placed in a regular way on a page. Maya manuscript texts are organized in blocks of 2 to 16 cartouches which constitute as many sentences, often followed by associated numbers, dates and eventually a picture. According to the number of cartouche spaces available on the almanac page to write a short or long sentence, the scribe would squeeze in or spread out writing signs among the cartouches to avoid empty boxes and obtain a nice-looking page layout.

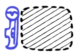
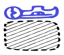
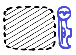
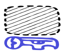
2.2 Glyph types and orientations

In the ancient Maya writing system, there are two types of primitive glyphs called *central elements* and *affixes*. Usually the shape of central elements is closer to square whereas affixes are narrower.

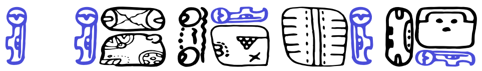
Central elements always appear in the same orientation but affixes turn so that they stick to other glyphs by their long side, following a general orientation rule.

Complete glyphic cartouches are made up of one to five basic signs or glyphs. Thompson [11] has shown that affixes, like *ni*, present rotation patterns and symmetries, around a central element, such as *KIN*, whose orientation is fixed.

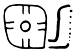
T116  *ni* T87  *te*

Through analysis of the Maya codices, we have determined that affix patterns follow a definite rule. For example, *te* (tree) is an affix and it usually attaches to a central element like this:    . So, there are five standard orientations for each affix: when it is single and when it attaches from the left, from the right, etc.

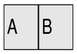
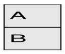
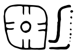
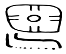
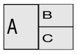
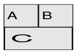
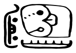
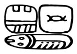
In the Dresden Codex we find the following corresponding cartouche compositions:



Affixes are generally syllabic value signs which can combine together or with a central element to write a Maya word. Central elements are generally of a logographic nature, corresponding to a morpheme or a word, and read globally; for example *KIN* (sun, day).

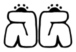
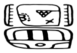
A complete glyphic cartouche often corresponds to a lexical entry with preceding and following grammatical affixes as in  *KIN-ni* (sun, day), but it can also in some cases correspond to two words if they are short, or more rarely otherwise to a part of an expression spelled over two cartouches.

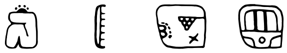
2.3 Glyph composition into cartouches

For the composition of glyphs the following standard notation is used in the historical and linguistic literature on ancient Maya: if $\boxed{\text{A}}$ and $\boxed{\text{B}}$ are two glyphs (primitive or not), then `A.B` and `A:B` encode the glyphs  and ; for example: 204.031  and 204:031 . To control the order of composition, one can use parentheses in the same way as in mathematical formulas. For example, both `A.B:C` and `A.(B:C)` stand for  but `(A.B):C` stands for ; thus: 026.172/023  and (154.123)/306  (in glyph codes, '`/`' means the same as '`:`').

3 Description of MayaPS

3.1 Main features

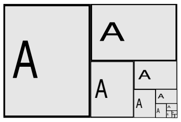
The text cartouches  and  that we used above are composed of these primitive glyphs:

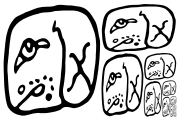


MayaPS does not consider any grammatical function or linguistic meaning of primitive glyphs. They are just graphical elements which are the elementary bricks of Maya typesetting, like letters in European languages.

As should be clear already, each primitive glyph is referred to by its code (more specifically called the *glyph code*). The glyph codes for the above are `422`, `023`, `153`, and `220`. In general, a glyph code is any sequence of digits `0`. . . `9` and letters `a`. . . `z`, `A`. . . `Z`. The encoding system is rather flexible. For example, after the command `\mayaDefine{A9z}{422}` you can type `maya{A9z}` to get .

Any formula of this kind is allowed by MayaPS, even something like this:

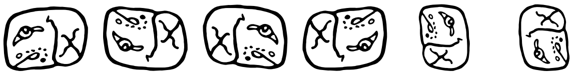




The picture on the right hand side (a fantasy writing cartouche, as there is no such glyph in ancient Maya writing) is printed by the command

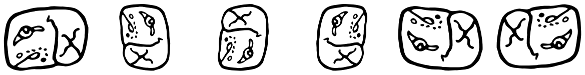
```
\maya{322.322:(322.322:(322.322:(322.322:(322.
322:(322.322)))))}
```

The type of each glyph (affix or central) and the five default orientations for each affix are written in the font file. Orientation can be changed with *modifiers* `-|+*?` whose meaning is:

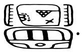
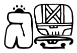
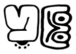


322 -322 |322 +322 *322 ?322

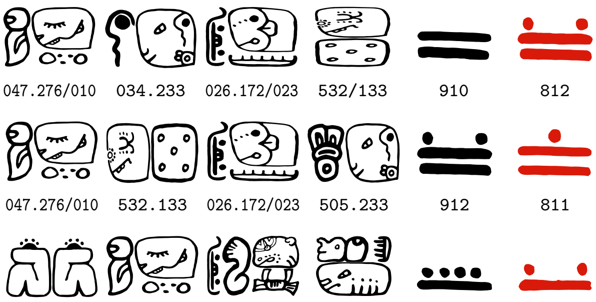
The modifiers can be composed together yielding:



322 *322 -*322 +-*322 ?+-*322 |?+-*322

Let us discuss again the glyph . We see that the primitive glyph 023 occurs here in two different ways: and . Moreover, in it occurs twice horizontally. MayaPS automatically chooses the orientation of each primitive glyph of the affix type according to "orientation rules" formulated by the first author after a careful analysis of ancient manuscripts. Of course, these rules have exceptions, but it is easy to handle them. For example, if you type `\maya{422.222/024}`, you obtain  (the default orientation), but if you type `\maya{-422.410}`, you obtain .

A more representative example is the palaeography of page 22c of the Dresden Codex, due to the first author:



047.276/010 034.233 026.172/023 532/133 910 812

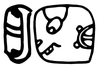
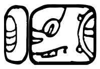
047.276/010 532.133 026.172/023 505.233 912 811

422.422 047.276/010 026.076.453/072 510/303 909 807

To obtain this, we typed:

```
\mayaC{  % \mayaC = glyphs with input codes
047.276/010 034.233 026.172/023 532/133 910 812
047.276/010 532.133 026.172/023 505.233 912 811
422.422 047.276/010 026.076.453/072 510/303 909
                                             807}
```

MayaPS permits the support of multiple fonts. In this paper we use mostly the font `codex` created using the tools mentioned in §4.1, but another style glyph set `gates` has been implemented based on the same glyph codes, for example:

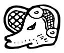
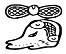
`codex` font:  `gates` font: 

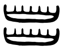
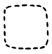
111.274 111.274

MayaPS provides a tool to add or replace glyphs in existing fonts and a tool for making new fonts.

3.2 Substitutions (ligatures)

A list of substitutions is associated with each Maya font. As we mentioned in the introduction, a new substitution $s_1 \to s_2$ can be defined by the command `\mayaDefine{`s_1`}{`s_2`}` where s_1 and s_2 are any strings (chains of characters). Substitutions are applied to the arguments of glyph drawing macros. They are applied non-recursively. Some substitutions can be predefined in a Maya font. Three types of substitutions are predefined in the font '`codex`':

(1) *Ligatures.* One or several affixes or central elements can be melded inside a central element or, more frequently, inside a head figurative element instead of being simply attached to it, forming a *ligature* as a single bound form. For example, when you type `\maya{070:349}`, you obtain  353 rather than  070:349 because of the predefined substitution (ligature) `070:349` → `353`. Here we typed `\maya{070:(349)}` to print the non-ligatured form. The ligature was not applied because '`070:349`' is not a substring of '`070:(349)`'.

As another frequent ligature example you have: 373 *Cacau* D7c (2), that decomposes into simpler glyphs: `369<023/023>`. The operator `<...>` indicates that both affixes 023  are placed in the centre of 369 .

Within Maya texts, both forms — melded into a ligature (single glyph code), and separately drawn (2–3 glyph codes) — are equivalent and may constitute orthographical variants, e.g.: and . Our catalogue includes around 100 ligature glyphs.

(2) *Thompson codes.* The basic glyph codes in the font '`codex`' are based on glyph numbers of

the Evreinov catalogue [3], from which font drawings were derived. However, many specialists are more familiar with glyph codes in Thompson's catalogue [11]. Due to predefined substitutions `T1`→`026` , `T2`→`410` etc., those codes can be used also for text input.

(3) *Phonetic values.* Reconstituted phonetic values in the Maya language can also be used to ease text input. Phonetic values of affixes are conventionally written in lowercase letters: `a`→`050` , `aj`→`044` , `ak`→`506` etc.; for central elements uppercase letters are used: `AT`→`200` , `BA`→`213` , `BA2`→`212`

As with Thompson codes, predefined substitution tables permit the use of multiple character input methods, analogous to Chinese character computer input with either *pinyin* (PRC's official romanization), *cangjie* (decomposition into graphic keys) or *dianbaoma* (Chinese telegraph codes).

4 Maya fonts

4.1 Font creation mechanism

A MayaPS font is an ASCII text file with the extension `.mpf`. Its structure is rather flexible. It is described in detail in [8]. It has several sections of PostScript code (a header and glyph definitions) [6] separated by lines starting with `%@`. TeX macros use these marks to select needed sections for including them into '`mayaps.tmp`' (see §7.1). Substitution rules (see §4.2) have the form `%L@` s_1 s_2.

There is a tool (involving a special vectorizer '`cotrace`') for creating MayaPS fonts out of monochrome bitmaps. The fonts supplied with MayaPS are made with it.

When MayaPS fonts created with this tool are used, they generate Type 1 fonts [5] in the resulting `ps` file. As Type 1 Maya fonts are used, the resulting `pdf` document after conversion is considerably smaller than the intermediate `ps` file. Only definitions for those font signs used in the text are included, and just once.

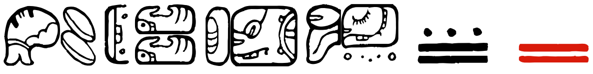
The `\mayaAddGlyph` macro allows for inclusion of a new glyph from an `eps` file but we do not recommend using it too frequently because it rapidly increases the resulting `ps` and, especially, `pdf` files.

4.2 Currently available fonts

To date, three extensive Maya script fonts and a partial Olmec script font have been produced by the first author.

The font '`codex`' was designed primarily from drawings of the Evreinov glyph catalogue [3] and is the one used up to now in this article.

The font '`gates`' is derived from the lead cast font designed by William Gates in the 1930s for his book [4], and has been implemented based on the same glyph codes as for font '`codex`'. For example, the beginning of the above quote from the Dresden Codex printed in the font '`gates`' looks like:

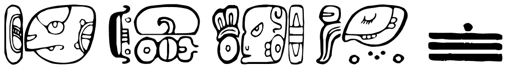


`451.452` `026.314/(314)` `111.274` `047.276/010` `913` `810`

(we typed `\gates` and then, just copied the same codes used above for a text in the '`codex`' style).

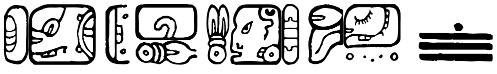
The font '`thompson`' was made based on the Thompson catalogue [11] drawings, where the original Txxx codes from the catalogue are implemented, showing that glyph codes in different fonts can be independent.

A correspondence ligature table, included in the font, supports glyph input using Evreinov-derived codes in a text displayed in the '`thompson`' font, as shown:

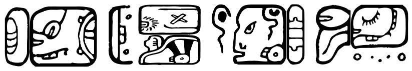


`T24.T1047a` `T1.T19:T59a` `T58a.T1026.T103` `T15.T736a:T140` `XVIn`

Similar correspondence ligature tables are included in the '`codex`' and '`gates`' fonts for glyph input using Thompson codes and phonetic reconstituted values, for example:

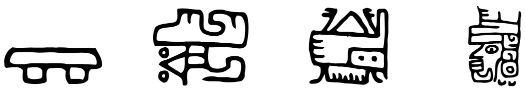


`T24.T1047a` `T1.T19:T59` `T58.T1026.T103` `T15.T736a:T140` `XVIn`
`111.274` `026.144:056` `505.233.112` `047.276:010` `916`



`li.AM2` `u.TUN/CHU` `uj.UJ.ki` `death.KIMI/la`

The font '`olmeca`' is derived directly from drawings of *La Mojarra* stella [7] of the 2^{nd} century AD and from the *Tuxtla* tablet. It represents a partial set of Olmec glyphs, which were composed in vertical texts without rotation of affixes.



5 Principles we tried to follow

5.1 Length optimization

Suppose you already have a ps file (produced by TeX/Dvips [9]) in an alphabetic language and you include ancient Maya glyphs into it. Then MayaPS adds to the ps output only:

- MayaPS header (7 Kb);
- definitions of primitive glyphs (0.5–3 Kb per glyph for 'codex');
- about 60 bytes for each occurrence of each composed glyph.

MayaPS includes the definitions of only those primitive glyphs which are effectively used in the text. Each definition is included only once even if the primitive glyph is used many times. This property holds after conversion from ps to pdf, because Type 1 fonts are used for primitive glyphs (as usual, the size of pdf generally lies between the sizes of ps.gz and ps).

5.2 Simplicity of installation and no need of support

To use MayaPS, it is enough to copy a few files into any directory (folder) 'visible' by TeX, for example, the directory where the tex file is. In particular, no extra font in the usual sense is needed (a typical beginner's problem is how to make TeX 'see' a new font).

The algorithm to draw a composed glyph is implemented in the PostScript language [6], and features of Dvips [9] are used for calling it from a TeX file via a \special macro (this is why MayaPS does not work with pdfTeX). So, TeX, Dvips, and PostScript are needed. Nothing else is used in MayaPS.

The only exception is the tool for creating new MayaPS fonts (mpf files) where C programs are used, but the font file format is described and it is easy to make an mpf file out of a Type 1 font (detailed instructions are given in reference [8]).

6 A few words about the implementation

6.1 Interaction between TeX and PostScript

A glyph code (example: 111.+176/111 for) is passed to the ps output by the Dvips command:

```
\special{"M(111.+176/111) w h d E}
```

where $w \times h$ is the cartouche size and d is the font descriptor (an integer number). Dvips literally includes the argument of \special{" } into the ps file and the task of drawing the composed glyph is delegated to a PostScript interpreter. The glyph drawing subroutine E is defined in the header included to the ps file by the Dvips command:

```
\special\{header:mayaps.tmp}
```

(see [9]; §5 for more detail).

Before issuing the command \special{"M...E}, all primitive glyph names are extracted from the glyph code and checked to see if their definitions are already included into the header mayaps.tmp. The token list \output is extended so that at the end of each page the definitions of all newly-appeared glyphs are copied from mpf files to mayaps.tmp.

6.2 Substitution mechanism

In earlier versions of MayaPS, the substitution mechanism was implemented by creating for each substitution $s_1 \to s_2$ a macro whose name (control sequence) contains s_1 and whose expansion is s_2.

Then, for each substring of each composed glyph, the corresponding macro was checked for existence by this command:

```
\ifx\csname ... \endcsname\relax
```

However, this command leaves the tested control sequence in TeX's memory forever. As a result, TeX's usual capacity (60000 control sequences) was exceeded when the thesis [1] exceeded 300 pages.

The new substitution mechanism creates the tree of initial subwords of left hand sides of all substitutions. Now the number of control sequences used does not exceed the size of this tree.

Acknowledgments. The idea to use the PostScript language rather than TeX for drawing composed glyphs belongs to Ilya Zakharevich. The TeX part of MayaPS is inspired by epsf.tex (by Tom Rokicki) and even some code is taken from there. Our glyph numbering system is adapted from the Evreinov catalogue [3], as are most codex font drawings. Glyph drawings for the gates font are taken without modification from William Gates' [4] Dresden Codex palaeography.

Another attempt to adapt (LA)TeX for ancient Native-American languages that concerned Olmec writing was done in [10], using a very different approach from ours.

References

[1] B. Delprat. *Le codex de Dresde: Paléographie et traduction comparée d'un almanach maya du 15e siècle.* Thèse de doctorat, Institut National des Langues et Civilisations Orientales, Paris, n.p. (thèse en cours).

[2] B. Delprat and S. Orevkov. mayaTeX – un sistema de composición tipográfica de textos

jeroglíficos mayas para la computadora. In *XXI Simposio de investigaciones arqueológicas en Guatemala*, Guatemala de la Asunción, 23–27 July 2007.

[3] E. V. Evreinov, Yu. G. Kosarev, and V. A. Ustinov. *Primenenie elektronikh vychislitel′nykh mashin v issledovanii pis′mennosti drevhikh maiya [The Use of Electronic Computing Machines Applied to Research on Ancient Maya Writing].* Akademia nauk SSSR [Academy of Sciences of the USSR], Novosibirsk, 1969. 4 vols.

[4] William E. Gates. The Dresden Codex reproduced from the tracings of the original colorings and finished by hand. In *Maya Society Publication*, number 2. The Maya Society at the Johns Hopkins University, Baltimore, 1932.

[5] Adobe Systems Inc. *Adobe Type 1 Font Format*. File `T1Format.pdf` available on `http://www.adobe.com`.

[6] Adobe Systems Inc. *PostScript Language Reference Manual*. Files `plrm.pdf` and `plrm2.pdf` available on `http://www.adobe.com`.

[7] Martha J. Macri and Laura M. Stark. *A Sign Catalog of the La Mojarra Script*. Pre-Columbian Art Research Institute, San Francisco, 1993.

[8] Stepan Orevkov. *MayaPS: Typing Maya with TEX/LATEX. Reference manual*. available on `http://picard.ups-tlse.fr/~orevkov`.

[9] T. Rokicki. *Dvips: A DVI-to-PostScript Translator*. File `dvips.pdf` included in most TEX distributions, available on `http://www.ctan.org`.

[10] A. Syropoulos. Typesetting Native American languages. *Journal of Electronic Publishing*, 8(1), 2002. `http://www.press.umich.edu/jep`.

[11] J. E. S. Thompson. *A Catalogue of Maya Hieroglyphs*. Univ. Oklahoma Press, 1962.

◇ Bruno Delprat
INALCO & SeDyL-CNRS, 7, rue Guy Môquet
94801 Villejuif cedex
France
`brunodelprat (at) club-internet dot fr`
`http://celia.cnrs.fr/Fr/Labo/Delprat.htm`

◇ Stepan Orevkov
Institut de mathématiques de Toulouse, Université Paul Sabatier
31062 Toulouse
France
`orevkov (at) math dot ups-tlse dot fr`
`http://www.math.univ-toulouse.fr/~orevkov/mayaps.html`

Experiences with Arabic font development

Sherif S. Mansour, Hossam A. H. Fahmy

Abstract

This is a report of our experiences attempting to use a new font, AlQalam, for the Arabic script within TEX. Then we want to make use of the new features introduced in LuaTEX to build our context analysis and line breaking engines to achieve a complete functional font package. We describe the challenges of producing high-quality Arabic fonts in general and what AlQalam has introduced to meet Arabic script requirements. We also describe the problems we faced trying to figure out how to use a new right-to-left font within TEX, what approaches we used to debug the font and some debugging results. This remains work in progress.

1 Arabic script and Naskh style

The Arabic alphabet is used to write many languages in many places around the world. Also, Arabic is of great importance to Muslims (about a quarter of the world's population), as it is the liturgical language of Islam.

The most distinctive features of the Arabic alphabet are that it includes 28 letters and is written from right to left in cursive style, i.e., many or all letters in a word are connected.

Arabic has six major writing styles: Kufi, Thuluth, Naskh, Riq'aa, Deewani, and Ta'liq. Naskh style is the most commonly used for printing, in both traditional texts such as the Muslim Holy Book (the Qur'an) as well as contemporary publications.

2 Challenges in producing high-quality Arabic fonts

The main challenge of producing high-quality Arabic fonts is that Arabic calligraphy is an art. The rules to follow when composing Arabic texts have great flexibility in choosing different variations of letter forms and constructing complex ligatures. These variations and ligatures add an aesthetic touch to the script and also justify the text as needed.

Every Arabic letter may have four basic forms depending on its location in the word: initial, medial, final, and isolated. Fig. 1 shows the different forms of the "Baa" letter as an example. Every letter form may have different variations to be used depending on the preceding or succeeding letter. Fig. 2 (taken from [8]) for example shows the different variants of the forms of "Baa". The first shape (starting from the right side of the first line) is the isolated "Baa" form variant. The second, third and sixth shapes are initial form variants of "Baa" when rising letters like "Alef" precede it, letters like "Jeem" precede it and letters like "Seen" precede it, respectively. The fourth and fifth shapes are medial form variants of "Baa" when letters like "Haa" and letters like "Raa" precede it respectively. The seventh shape is the general medial form variant of "Baa" and the last shape is a final form variant of "Baa".

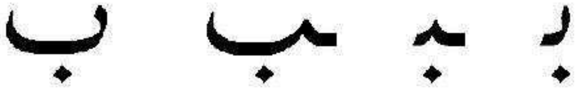


Figure 1: From right to left: initial, medial, final, and isolated forms of "Baa"

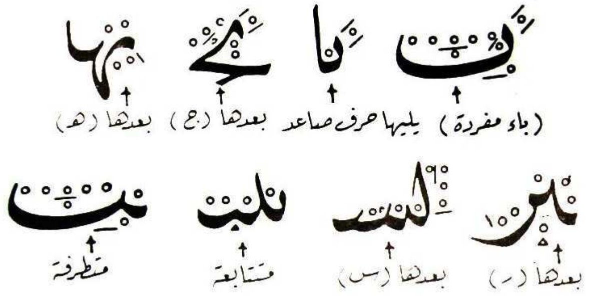


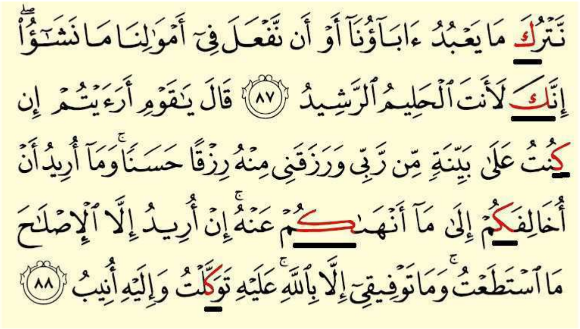
Figure 2: Letter "Baa" form variations

Fig. 3 shows a part of Mushaf Al-Madinah (The Holy Qur'an — Madinah print) [1] as a rich example of the usage of different combinations of variations with different elongations. This printing, like most printings of the Qur'an, was hand-written by a calligrapher and is not typeset. Computer typesetting (and to a large extent also traditional mechanical typesetting) are well behind what a calligrapher can produce for such complex texts.

The underlined shapes represent the same Arabic letter, which is "Kaf". Notice the different forms used depending on the location of the letter in the word and notice that some forms have different variations depending on the elongation requirements for line breaking, text justification and preceding or succeeding letters.

3 The font AlQalam and its features

Several trials were performed to produce high-quality Arabic fonts. One of them was a project named AlQalam ("The Pen" in Arabic), started in 2005 under the co-author's supervision [3]. AlQalam's target was to simulate an Arab calligrapher's pen for Naskh style (as used to typeset the Qur'an printing, for example). AlQalam then might be used in typesetting any traditional texts as well as any generic publi-



نَتۡرُكَ مَا يَعۡبُدُ ءَابَآؤُنَآ أَوۡ أَن نَّفۡعَلَ فِيٓ أَمۡوَٰلِنَا مَا نَشَٰٓؤُاْۖ
إِنَّكَ لَأَنتَ ٱلۡحَلِيمُ ٱلرَّشِيدُ ٨٧ قَالَ يَٰقَوۡمِ أَرَءَيۡتُمۡ إِن
كُنتُ عَلَىٰ بَيِّنَةٖ مِّن رَّبِّي وَرَزَقَنِي مِنۡهُ رِزۡقًا حَسَنٗاۚ وَمَآ أُرِيدُ أَنۡ
أُخَالِفَكُمۡ إِلَىٰ مَآ أَنۡهَىٰكُمۡ عَنۡهُۚ إِنۡ أُرِيدُ إِلَّا ٱلۡإِصۡلَٰحَ
مَا ٱسۡتَطَعۡتُۚ وَمَا تَوۡفِيقِيٓ إِلَّا بِٱللَّهِۚ عَلَيۡهِ تَوَكَّلۡتُ وَإِلَيۡهِ أُنِيبُ ٨٨

Figure 3: An example from surat Hud: forms of Kaf

cations (including scientific ones) in the languages using the Arabic script.

At first, AlQalam grew out of modifications to ArabTEX [4]. Modifications were mainly intended to respond to the specific needs of typesetting the Qur'an such as adding pause signs, some additional diacritics (marks used as phonetic guides) and the abilities to stack them on top of each other, scale them and correctly position them on the word. Also, some modifications to the pen used were made to improve the shape of some letters and symbols.

In 2008, a new font project for AlQalam was started [7]. That font is meta-designed such that each character is described by a number of parameters to allow the creation of many variants that connect with the surrounding characters correctly. Those variants may be different in shape and in their amount of elongation. Starting from this period many modifications were made and new features added.

AlQalam's font features up till now:

1. All font shapes are meta-designed using METAFONT to enable greater flexibility while joining glyphs together and provide smoother letter extensions.
2. It contains the generic four different forms of Arabic letters (initial, medial, final, and isolated).
3. It also contains different parameterized shapes for letter forms (the main source of the form variants is Mushaf Al-Madina).
4. It is based on the concept of primitives (reusable glyphs). For example, Fig. 4 shows the Waw head primitive (the small circle). This Waw head is reused in combination with the body (skeleton) of letter "Baa" to produce the letter "Faa". Also, the "Waw" head can be reused in combination with the body of the letter "Noon" to produce the letter "Qaf".

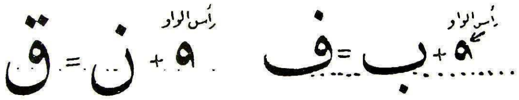


Figure 4: The "Waw" head primitive

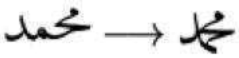


Figure 5: Vertical placement of glyphs

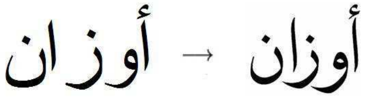


Figure 6: Kerning

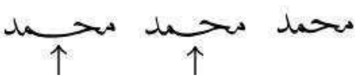


Figure 7: Joining glyphs with smooth, dynamic kashidas

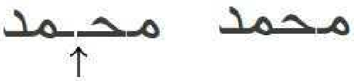


Figure 8: Static fixed length kashidas

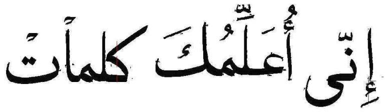


Figure 9: Parameterized diacritics

5. The font supports vertical placement of glyphs: Various ligatures have been added to the font to support complex vertical placement combinations as shown in Fig. 5.
6. Kerning: Borders of letter boxes have been adjusted to support kerning as shown in Fig. 6.
7. Joining glyphs with kashidas: the kashida is the most widely used glyph to join letters. AlQalam implements the kashida as a dynamic smooth glyph, as shown in Fig. 7 [2]. This is preferable to the current standard fonts that implement the kashida as a static fixed length glyph, as shown in Fig. 8.
8. Parameterized diacritics: A complete set of parameterized diacritics that can be elongated according to the width of the associated letter is available, as shown in Fig. 9.
9. Mathematical symbols: AlQalam is one of three fonts that have a complete set of Arabic math symbols at the time of writing. (The other

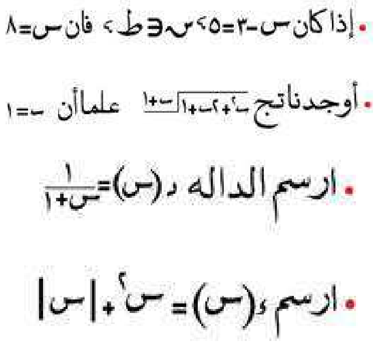

• إذا كان س-٣=٥، س∋ط، فان س=٨
• أوجد ناتج ... علما أن ب=١
• ارسم الداله د(س)=١/(س+١)
• ارسم د(س)=س٢+|س|

Figure 10: Set of Arabic mathematical equations typeset with AlQalam

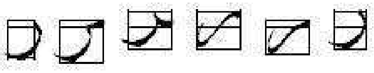


Figure 11: Character boxes approach

two fonts are RyDArab [5] and Mathfont [6].) Examples for Arabic equations generated using AlQalam are shown in Fig. 10.

4 Calling the different shapes of a letter

Testing of individual shapes was done along with the development of the font letter forms, form variants, mathematical symbols, etc. Once the basic development was finished, testing of the letters' positions against the baseline and against each other when included in the same word was needed. In addition, there was a need to perform checks for missing forms, to assure smooth joins between letters and correct kerning.

This drove us to call the METAFONT letters under TEX to debug the font. To do that, two approaches were possible:

1. Define the character's borders through boxes, similar to the Latin script, and define a character for each shape and with fixed elongation as shown in Fig. 11. This approach, by definition, must have a finite number of characters in the font at the end. Such a finite, i.e. limited, number of characters means that we must use only a limited set of values for the parameters of each character. So, while debugging we used for example only the "Noon" form shown in Fig. 12 despite the font's capability to generate elongated shapes of this form as shown in Fig. 13.

2. Using LuaTEX's embedded Lua and METAPOST engines to call the correct shape with the suitable elongation value. The main advantage of this approach is that we will be able to benefit from all the font features. But lacking experience in this approach made us think that we should postpone using it to the next phases, as more time will be needed to debug it.

Figure 12: Isolated form of the letter "Noon" without elongation

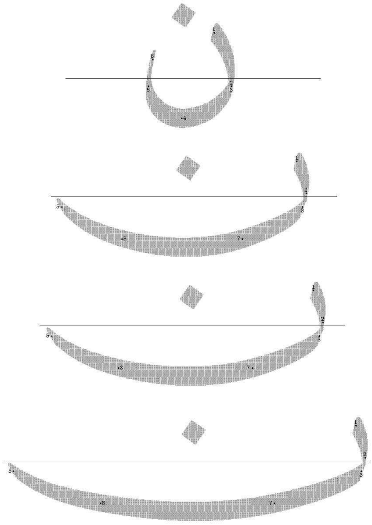


Figure 13: Isolated form of the letter "Noon" with different elongation values

Hence we started with the first approach to complete the current debugging phase. Meanwhile we will learn about the possibilities of calling METAPOST from within LuaTEX in order to generate dynamic fonts in the future.

5 Problems found during font debugging

As expected, some bugs appeared when using the font under TEX. If we take a look at Fig. 14 we can easily notice the bugs, from right to left as follows. The first and fifth cross signs mark a kerning bug with the letter "Waw" as it appears too far from the succeeding letter. The second and fourth signs mark bad joins of letter "Ain". Also the letter shape itself needs some modifications as marked by the third sign. More bad joins appear between letters "Qaf" and "Sad" and letters "Lam" and "Dal" as marked by the sixth and ninth signs. The seventh sign marks a missing letter that should be added to the font package. The eighth sign marks a mistake in specifying the border of letter "Alef" that caused it to be too close to the succeeding letter.

We have worked on fixing these bugs and many

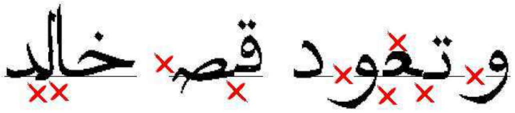


Figure 14: Font state at an early debugging phase

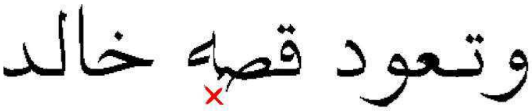


Figure 15: Font state after fixing bugs

others that appeared in turn, to get to a state with the font that made us ready to move to the next step (despite having very few joining bugs as shown in Fig. 15). This is implementing the multi-layer context analysis algorithm to make using the font more user friendly, by automatically choosing the suitable letter forms and form variants instead of the user having to do so manually.

6 Work in progress and future work

First, adding context analysis via LuaTEX's embedded Lua and METAPOST engines: This will enable calling the suitable METAFONT form with the suitable elongation parameters according to context. Multi-layer context analysis is expected, as a basic layer would be needed to choose the suitable form of the letter according to its preceding or succeeding letter, and a second layer would construct ligatures, correctly update the positions of the diacritics and associated symbols and also control the elongation parameters according to the text justification requirements. We are currently working on implementing the first layer.

Second, implementing a new line breaking algorithm with a new penalty system that supports usage of elongation, ligatures, form variants in addition to spacings when taking the breaking decisions [2]. A dialogue between METAPOST and TEX about word widths and line width requirements is expected. This will be performed through the usage of LuaTEX's callbacks feature to override the main algorithm.

References

[1] *The Holy Qur'an.* King Fahd Complex for Printing the Holy Qur'an, Madinah, KSA, 1986.

[2] Mohamed Jamal Eddine Benatia, Mohamed Elyaakoubi, and Azzeddine Lazrek. Arabic text justification. *TUGboat*, 27(2):137–146, January 2007.

[3] Hossam A. H. Fahmy. AlQalam for typesetting traditional Arabic texts. *TUGboat*, 27(2):159–166, January 2007.

[4] Klaus Lagally. ArabTEX — Typesetting Arabic with vowels and ligatures. In Jiří Zlatuška, editor, *EuroTEX '92: Proceedings of the 7th European TEX Conference, Prague, Czechoslovakia, September 14–18, 1992*, Proceedings of the European TEX Conference, pages 153–172, Brno, Czechoslovakia, September 1992. Masarykova Universita.

[5] Azzeddine Lazrek. RyDArab — Typesetting Arabic mathematical expressions. *TUGboat*, 25(2):141–149, 2004.

[6] Mathfont. `http://mhafiz.deyaa.org/mathfont.html`.

[7] Ameer M. Sherif and Hossam A. H. Fahmy. Meta-designing parameterized Arabic fonts for AlQalam. *TUGboat*, 29(3):435–443, January 2008.

[8] Ahmad Sabry Zayed. *Ahdath Al-turuq Leta'leem Al-khotot Al-'arabiya [New methods for learning Arabic calligraphy].* Maktabat ibn-Sina, Cairo, Egypt, 1990.

◇ Sherif S. Mansour, Hossam A. H. Fahmy
Electronics and Communications Dept.
Faculty of Engineering, Cairo University
Egypt
`sherif.s.mansour (at) gmail dot com`
`hfahmy (at) alumni dot stanford dot edu`

The fonts we choose

Boris Veytsman

One of the most important choices a book designer makes is the selection of the font for the body text. This decision defines the general "look and feel" of the future book and influences all other decisions.

How should a designer select the font? There is considerable lore about the suitability of certain fonts for certain kinds of books. Often the choice is based on "physiological" considerations: it used to be commonly held that serifed fonts are better for continuous reading, or that fonts with high contrast are suitable for textbooks, etc. However, recent studies (Morris, Aquilante, Yager, and Bigelow, 2002; Legge and Bigelow, 2011; Akhmadeeva, Tukhvatullin, and Veytsman, 2012) show that human brains are very good in accommodating to the differences in font sizes and shapes, and the ease of reading is more or less the same across the wide variety of fonts as long as the font is reasonable.[1] Thus the choice of body font might be less determined by the physiology of reading than used to be thought.

Does this mean that the choice is unimportant? In my opinion, not at all. Indeed, there are no "physiological reasons" to choose a formal suit over sweatpants (actually sweatpants might be more comfortable). Nevertheless a person coming to a fine dinner in sweatpants (or, for that matter, to a gym in white tie) is wrong: the costume sends a wrong *message.*

What kind of message does a font send? Eva Brumberger (2003a, 2003b) made a series of interesting studies on this subject. She asked participants to look at several typefaces[2] and estimate on a scale from 1 to 7 the applicability of such characteristics as "Cheap", "Cold", "Confident", "Sloppy", etc.[3] She found that the fonts have stable "personae": for example Black Chancery is "elegant", Arial is "direct" and Comic Sans is "friendly".

Even more interesting was another experiment: Brumberger gave the participants texts typeset with different typefaces and asked them to comment about the appropriateness of the chosen font for the given text and score the texts against the same characteristics. The participants were "clear and consistent" about the proper or improper choice of the body font. Did the properties of the typefaces color the readers' impression about the *text?* The answer is complicated. There was no statistically significant dependence of the reported text properties on the font, with one important exception. Namely, the perceived "seriousness" of the text strongly ($p < 0.004$) depended on the typeface chosen, with Times New Roman giving the text the strongest aura of seriousness and Counselor Script, a calligraphy font, having the strongest opposite effect. The effect depended on the texts themselves as well as on the gender and demography of the participants.

These results mean that a font does send a message to a reader, and on at least one scale (seriousness) it influences the message of the text.

Recently Errol Morris (2012a, 2012b) published a two-part series in a *New York Times* blog following an unusual experiment designed by Benjamin Berman. Morris asked his readers to tell whether they agree or disagree with a certain paragraph about the danger of Earth colliding with a kilometer-sized asteroid. Unbeknownst to them, the paragraph was presented to each viewer in one of five different fonts, chosen randomly: Baskerville, Computer Modern, Georgia, Helvetica, Comic Sans and Trebuchet. The answers were recorded with the font chosen.

While this experiment, as Morris himself readily recognizes, lacks the controlled environment of a true scientific study (the participants are self-selected, we do not know their demographics, etc.), the sheer number of answers (45 thousand!) makes the result very interesting.

Berman classified the positive responses with five points for "strongly agree", three points for "moderately agree" and one point for "slightly agree", and similarly for the negative responses. In Figure 1 we show the difference between the agreement and disagreement levels as defined by Berman's scores. This quantity can be interpreted as the overall measure of the readers' attitude toward the statement.

As seen from the figure, the reader's attitude depends on the font chosen. The most persuasive typeface turned out to be Baskerville, with Computer Modern being a close second, while the least persuasive one was, as expected, Comic Sans. Due to the large number of participants the confidence in the result is high ($p < 0.0068$). This result makes the CERN decision to announce the discovery of the Higgs boson using, of all fonts, Comic Sans, even more mysterious (Morris, 2012a).

One can argue that "trustworthiness" of the text is directly related to its perceived "seriousness", so

[1] Some typefaces and font sizes *are* difficult to read, but those were never intended for continuous reading.

[2] Adler, Arial, Bauhaus MD BT, Black Chancery, Casablanca Antique Italics, Comic Sans MS, Counselor Script, Courier New, Garamond, Harrington, Lucida Sans Italic, Lydian BT, Square 721 BT, Times New Roman, Van Dijk.

[3] The full list: Cheap, Cold, Confident, Dignified, Elegant, Feminine, Formal, Friendly, Inviting, Loud, Masculine, Playful, Pretentious, Professional, Relaxed, Scholarly, Serious, Sloppy, Straightforward, Warm.

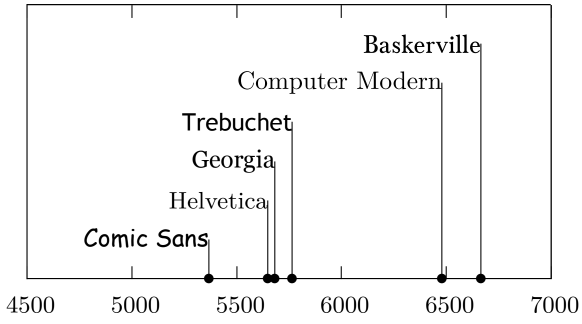


Figure 1: Level of trust in a statement presented in the given typeface (from the data in Morris, 2012a).

the results of experiments by Brumberger and Morris corroborate each other.

Another important conclusion is that fonts affect the reader's attitude towards the text. This is noteworthy for students who want better grades for their homework, and for their professors who want positive reviews of their scientific papers and research proposals. Errol Morris mentions a blog entry by Phil Renaud who noted a marked difference between his grades depending on the font of the essay. Since Computer Modern, the default typeface of TEX, scored high in this test, TEX users should probably rejoice: we made a good choice of the typesetting system and might expect to reap some benefits from it. By the way, neither study looked into the effects of such typographic features as good justification, hyphenation, line and page breaking. It is not too far fetched to suggest that these niceties might also add a point or two to the final grade or to the probability that a proposal gets a good review.

What causes this effect of a typeface on the reader's trust? I can only offer my own guess. Since this paper is typeset in Computer Modern, I hope you can believe it.

I do not think there is an inherent property of a typeface to be "trustworthy". Rather, our attitude towards it is determined by our background and previous experiences. The same is true for other cultural artifacts, such as clothes. Today we consider tuxedo to be formal dress (Errol Morris compares the formality of Baskerville to that of a tuxedo). However, it was originally (in the first half of the 19th century) casual dress, as opposed to the formal tailcoat. A smoking jacket was intended for smoking cigars in a relaxed manner, as different from a strict and scripted dinner. Thus the messages of a tuxedo then and now are completely different. Therefore there is nothing "inherently formal" in a tuxedo. We perceive it as formal today because we are accustomed for it to be worn on formal occasions. Its message is conditioned by our experiences.

I think the same is true for typography. We trust a text set in a certain typeface because we have read other trustworthy texts typeset in it. Baskerville has been used for good books for many years, which might explain its effect on the reader. The trust in Computer Modern might be caused by the fact that many well-educated readers of *New York Times* have read mathematics textbooks typeset in TEX and this typeface — and mathematics usually does not lie.

If this is true, then we as TEX users not only benefit from our software, but also have a certain responsibility towards the community. People give us a little bit of extra trust because other authors, who wrote in TEX with Computer Modern in the past, did a good job. I think we owe it to them to continue this tradition.

The TEXbook (Knuth, 1994) ends with the famous exhortation, "GO FORTH now and create *masterpieces of the publishing art!*" It seems that we ought to add to it the qualifier, which DEK probably considered self-evident, "And let the contents of these masterpieces be honest and true!"

Acknowledgement I am grateful to Barbara Beeton who brought the blog entry by Errol Morris to my attention and suggested that I write this essay.

References

Akhmadeeva, Leyla, I. Tukhvatullin, and B. Veytsman. "Do serifs help in comprehension of printed text? An experiment with Cyrillic readers". *Vision Research* **65**, 21–24, 2012.

Brumberger, Eva R. "The Rhetoric of Typography: The Persona of Typeface and Text". *Technical Communication* **50**(2), 206–223, 2003a.

Brumberger, Eva R. "The Rhetoric of Typography: The Awareness and Impact of Typeface Appropriateness". *Technical Communication* **50**(2), 224–231, 2003b.

Knuth, Donald Ervin. *The TEXbook*. Computers & Typesetting A. Addison-Wesley Publishing Company, Reading, MA, 1994. Illustrations by Duane Bibby.

Legge, Gordon E., and C. A. Bigelow. "Does Print Size Matter for Reading? A Review of Findings from Vision Science and Typography". *J. Vision* **11(5)**(8), 1–22, 2011.

Morris, Errol. "Hear, All Ye People; Hearken, O Earth (Part One)". New York Times Opinionator, 2012a. http://opinionator.blogs.nytimes.com/2012/08/08/hear-all-ye-people-hearken-o-earth.

Morris, Errol. "Hear, All Ye People; Hearken, O Earth (Part Two)". New York Times Opinionator, 2012b. http://opinionator.blogs.nytimes.com/2012/08/08/hear-all-ye-people-hearken-o-earth-part-2.

Morris, R. A., K. Aquilante, D. Yager, and C. Bigelow. "Serifs Slow RSVP Reading At Very Small Sizes But Don't Matter At Larger Sizes". In *SID 2002, San Jose, CA: Digest of Technical Papers*, pages 244–247. The Society for Information Display, 2002.

Using TeX Gyre Pagella OpenType Math

Herbert Voß

Abstract

With XeLaTeX and/or LuaLaTeX one can use OpenType, TrueType, and/or Type 1 fonts in his documents. The isolated world of (LA)TeX fonts is now history. However, the number of available mathematical fonts which corresponds to the possible text fonts is still very small. This brief note is an example of using the TeX Gyre Pagella OpenType font, including math.

1 Introduction

With the 2012 release of TeX Live another free OpenType math font has become available: Pagella (Palatino) Math from the TeX Gyre project (`http://www.gust.org.pl/projects/e-foundry/tex-gyre`), coordinated by GUST, the Polish TeX user group. This is an important step forward toward completing this longstanding work.

The following example shows an arbitrary composition of text and mathematical characters (from Stephen Hartke's document at `http://ctan.org/pkg/free-math-font-survey`). Both text and math are taken from the Pagella OpenType font.

Theorem 1 (Residue Theorem). Let f be analytic in the region G except for the isolated singularities $a_1, a_2, \ldots, a_m$. If γ is a closed rectifiable curve in G which does not pass through any of the points a_k and if $\gamma \approx 0$ in G then

$$\frac{1}{2\pi i}\int_\gamma f = \sum_{k=1}^{m} n(\gamma; a_k)\operatorname{Res}(f; a_k).$$

AΛΔ∇BCDΣEFΓGHIJKLMNOΘΩ℧PΦΠΞQRST UVWXYΥΨZ 1234567890
aαbβc∂dδeϵεfζξgγhℏℏıiȷjkκlℓλmnηθϑoσςφϕ℘ρϱ q r stτπuμνυvwωϖxχyψz ∞ ∝ ∅∅dð

2 Loading the fonts

Using OpenType and TrueType fonts with XeTeX or LuaTeX requires a bit of setup, usually done automatically at installation. In general, one can have the font files saved in his local or main TeX tree, the local or main system font directory. For Windows there is only one main directory, e. g. `c:\\windows\fonts`. On GNU/Linux, one can also have font files saved in his home directory.

While XeTeX uses the system `fontconfig` programs to find a font file, LuaTeX uses its own font handling which creates a font list `otfl-names.lua`. The name is a little bit misleading because it also lists the TrueType fonts (with extension `.ttf`).

3 XeLaTeX

XeTeX uses the config file `fonts.conf` from the system `fontconfig`. This is in general not available on Windows; thus, TeX Live and MiKTeX provide a `fonts.conf` for use there. Listing all available fonts with corresponding directories can be done by running the program `fc-cache` in a terminal:

```
fc-cache -v > xetex-font-search.log
```

(The TeX Live manual has more details about font configuration.)

4 LuaLaTeX

As already mentioned, LuaTeX uses its own file to find where the font files are saved. If a font defined in a document isn't found in that list, LuaTeX creates a new list to be sure that newly saved font files are also found. If one uses several new font files it is simpler to run the program `mkluatexfontdb` by hand before running the TeX document.

5 Using the fonts

There is in general no difference between XeLaTeX and LuaLaTeX when it comes to defining the fonts in a document:

```
\usepackage{fontspec}
\usepackage{unicode-math}
[...]
\defaultfontfeatures{Ligatures=TeX}
\setmainfont[
  BoldFont=texgyrepagella-bold.otf,
  ItalicFont=texgyrepagella-italic.otf,
  BoldItalicFont=texgyrepagella-bolditalic.otf]
 {texgyrepagella-regular.otf}
\setmathfont{texgyrepagella-math.otf}
\setsansfont [...]
```

With such font definitions we get PDF output with embedded fonts as shown here (truncated):

```
$ pdffonts beispiel.pdf
name                                        type
------------------------------------        -----------
GOKAFU+TeXGyrePagella-Regular-Identity-H    CID Type OC
OCZARV+TeXGyrePagella-Bold-Identity-H       CID Type OC
PHLUTA+TeXGyrePagella-Italic-Identity-H     CID Type OC
XEYJEO+TeXGyrePagella-BoldItalic-Identity-H CID Type OC
...
```

⋄ Herbert Voß
`herbert (at) dante dot de`

Editor's note: First published in *Die TeXnische Komödie* 3/2012, pp. 71–72; translation by the author.

OpenType math font development: Progress and challenges[*]

Ulrik Vieth

Abstract

A main reason for the development of the LuaTEX and XƎTEX engines has been to provide support for Unicode and OpenType font technology, which implies support for Unicode math and OpenType math as well. One important ingredient is the development of full-featured OpenType math fonts, which are needed to replace traditional math fonts. In this paper, we review recent progress in OpenType math font development as well as the many challenges faced by font developers of OpenType math fonts.

1 Introduction

In this paper, we will discuss technical details of OpenType math font development, so we will assume that readers have some familiarity with the basic concepts of Unicode math and OpenType math font technology.

When we speak about Unicode math, we refer to an effort that was undertaken between 1998 and 2002 by a group of scientific and technical publishers to get math symbols and alphabets accepted into the Unicode standard. As a result of this activity, hundreds of math symbols as well as dozens of math alphabets have been added to Unicode, and have become part of the official standard ever since Unicode 3.2 was released in 2002 [1, 2].

From a technical point of view, Unicode math is nothing special, just a convenient term for a subset of Unicode that is relevant for typesetting math.

When we speak about OpenType math, we refer to an extension of the OpenType font format [3] that was developed by Microsoft when they introduced support for math typesetting in Office 2007 [4, 5]. As a result of this, a new optional MATH table has been added to the OpenType font format, which is used to store all the additional information needed for proper typesetting of math, such as font metric parameters controlling the spacing of math as well as additional lookup mechanisms of glyph variants [6, 7].

From a technical point of view, OpenType math does represent an extension of the OpenType font format, but it uses a well-defined extension mechanism, so the optional MATH table will only be seen by typesetting engines which happen to know about math, while other engines will safely ignore it.

Finally, it is helpful to understand how Unicode math, OpenType math, fonts and typesetting engines work together.

Unicode math, by itself, only defines the encoding of mathematical input. It does not define any semantics of how a math formula is arranged or spaced. That is a matter left to the font technology (OpenType) and the typesetting engine (LuaTEX or XƎTEX or MS Office).

In Unicode, each math symbol is usually represented only once, regardless of how many sizes may be needed for proper typesetting. Letters of math alphabets are the exceptions: since a font change in math usually also conveys a different meaning, each variation of a letter has a separate slot.

OpenType, as a font technology, provides the glyphs and metric information for mathematical output. OpenType math fonts are encoded based on Unicode, but they can extend beyond the scope of Unicode by taking advantage of the private use area.

Where Unicode math defines only a single slot for each symbol, OpenType math provides lookup mechanisms for multiple sizes of glyph variants or glyph substitutions for constructed symbols.

Where Unicode math does not define any provisions for the semantics of math, OpenType math provides a table to store the font metric information controlling the spacing, but leaves interpretation of these parameters to the typesetting engine.

In the end, it all depends on having an OpenType math-capable typesetting system to take advantage of the information in OpenType math fonts and to properly arrange math formulas.

When OpenType math was first introduced, MS Office 2007 was the only available OpenType math engine, but both XƎTEX and LuaTEX have since implemented OpenType math capabilities [8, 9].

While XƎTEX provides only a partial implementation, LuaTEX aims to provide a full-featured OpenType math engine. (As of 2012, work on improving math typesetting in XƎTEX has been ongoing, so hopefully both engines will eventually be able to produce the same quality of math typesetting.)

2 Progress in OpenType math fonts

When OpenType math was introduced, only a single math font was available: Cambria Math, which was developed by Tiro Typeworks on behalf of Microsoft and bundled with MS Office 2007. In some sense, the situation was reminiscent of the early days of TEX, when Computer Modern was the only available math font in METAFONT format.

[*] First submitted for publication in the Proceedings of the 5th ConTEXt Meeting 2011 (to appear). Updated and revised for BachoTEX 2012. Updated and revised again for *TUGboat*. Reprinted with permission.

In recent years, several more OpenType math fonts have been added, so by the time of writing the first revision of this paper (September 2011) we had at least 4 math fonts available as released versions:

- Cambria Math, by Tiro Typeworks on behalf of Microsoft [11],
- Asana Math, by Apostolos Syropoulos [12],
- XITS Math, by Khaled Hosny, derived from the STIX fonts [13, 14],
- Latin Modern Math, by the GUST e-foundry [15, 16] (released in June 2011).

In the meantime, several additional choices of math fonts that were under development have been completed and released as well:

- Minion Math, by Johannes Küster [17],
- Lucida Math, by Khaled Hosny on behalf of TUG, designed by Bigelow & Holmes [18, 19] (released in March 2012),
- TEX Gyre Pagella Math, by the GUST e-foundry [20] (released in June 2012),
- TEX Gyre Termes Math, by the GUST e-foundry [20] (released in October 2012).

Finally, two more math font projects are under development or have been announced:

- Neo Euler, by Khaled Hosny on behalf of DANTE, designed by Hermann Zapf [21, 22],
- Maxwell Math, by Tiro Typeworks [23].

Given all these recent and ongoing font projects, we now have OpenType math font support for a number of popular text typefaces, such as Latin Modern, Times, Palatino, Lucida Bright, and Minion, although some of these are non-free, but subject to the usual industry proprietary licensing.

In some sense, the situation is now reminiscent of the early days of PostScript font support for TEX, when choices of math fonts were still very few, but several popular typefaces were already supported.

An interesting note is that these fonts have been developed by relatively few teams and individuals: Tiro Typeworks (Cambria, Maxwell), the GUST e-foundry (Latin Modern, TEX Gyre), and Khaled Hosny (XITS (with STIX), Neo Euler, Lucida), in addition to the solo efforts by Johannes Küster (Minion Math), and Apostolos Syropoulos (Asana Math).

We may conclude that OpenType math font development remains a very challenging task, that has been mastered by only a few.

3 References for OpenType math

Before we consider the challenges faced by font developers of OpenType math fonts, it may be worthwhile to consider the question: What is the basis for OpenType math font development?

First, there is a specification of the OpenType MATH table, developed by the Microsoft typography group. The specification is officially considered experimental and is available only on request, so it has remained unpublished for years, despite the fact that it has been widely adopted as a *de facto* standard by typesetting engines and font tools.

Next, there is a reference implementation of an OpenType math font, Cambria Math, developed by Tiro Typeworks on behalf of Microsoft. This font is widely available, and can be viewed with font editors such as FontForge, making it easily possible to study how it is constructed and what it contains.

Finally, there is a reference implementation of an OpenType math typesetting engine, namely MS Office, developed by the Microsoft Office group. Unlike the specification, which is at least somewhat open, the reference implementation is completely closed and off-limits, so it is impossible to see how the specification is actually implemented.

Given this scenario, developers of OpenType math fonts or engines face the problem of determining what defines the reference behavior and what may be needed to make their fonts behave correctly with different typesetting engines.

First, the OpenType math specification may not be perfect, leaving some gray areas open to questions or interpretations. For example, there is hardly any description when to apply italic correction.

Next, the reference OpenType math font may not be perfect either. For example, it may have some incorrect parameter settings, which may confuse some engines when interpreting the parameter values literally.

Finally, the reference OpenType math engine may not be perfect either. For example, it may have interpreted the specification in certain ways, or it may have included some workarounds to patch certain problems in the reference font.

In LuaTEX, the implementation of the math engine has followed the specification as much as possible, but in case of doubt, it has chosen to follow the reference implementation. OpenType math fonts developed and tested with LuaTEX should work with MS Office equally well, although they may not work quite as well with earlier versions of XƎTEX.

4 OpenType math development challenges

Development of OpenType math fonts is challenging for many reasons. Besides the inherent complexity, the size of the project is also a factor. Typical examples of OpenType math fonts may include between

1500 and 2500 symbols or letters, not counting size variants or optical design sizes. Besides technical issues and design issues, achieving some level of completeness is already a challenge by itself.

4.1 Completeness of math symbols

Unicode math defines thousands of math symbols in all. However, developers of OpenType math fonts can choose what to implement and will typically implement only a subset of the most important symbols, accepting some level of incompleteness.

Most OpenType math fonts include a common subset, comparable to what is in traditional TeX math fonts based on 7-bit or 8-bit encodings, but very few OpenType math fonts will provide the complete set of math symbols defined in Unicode.

At the moment, XITS Math is the most complete of all available OpenType math fonts, because it is based on the STIX fonts [24], which have taken nearly a decade to design and review all the glyphs.

At the other end of the spectrum, Neo Euler is the least complete of all OpenType math fonts, which is understandable given that Euler always had to rely on borrowing symbols from other fonts.

All the other available OpenType math fonts rank somewhere in between these extremes, with each of Asana Math, Lucida Math, and Minion Math providing some ranges of additional symbols that go beyond the scope of Cambria Math. By comparison, Latin Modern Math is still far less complete.[1]

One important factor to consider when converting TeX math fonts to OpenType format is that a number of macros need to be replicated by designed symbols. This will include symbols such as triple dots, double and triple integrals, negated or stacked symbols, arrows with hooks or tails, long arrows, over- and underbraces.

In the end, how much incompleteness can be tolerated depends on actual usage. If you are using only basic math, the symbol coverage of any available font will suffice, but if you need some special notations, it may be worthwhile to check to see which fonts provide the required symbols.

Probably the best reference of Unicode math symbols for various OpenType math fonts can be found in the documentation of the `unicode-math` package [25].

4.2 Completeness of math alphabets

Unicode math defines more than a dozen shapes of math alphabets:

- 4 shapes of a serif alphabet (regular, italic, bold, bold italic), each including Latin and Greek,
- 4 shapes of sans-serif (regular, italic, bold, bold italic), some including Latin and Greek,
- 2 shapes of Script/Calligraphic (regular, bold), each including upper- and lowercase,
- 2 shapes of Fraktur/Blackletter (regular, bold), each including upper- and lowercase,
- 1 shape of open face or Blackboard bold (regular), also including upper- and lowercase.

Once again, developers of OpenType math fonts can choose what to implement and will typically implement a common subset of the most important alphabets, but will not necessarily provide all the shapes.

Except for Neo Euler, which has only an upright shape by design, most fonts include at least 4 shapes of the main serif alphabet, but the completeness of other math alphabets varies considerably.

Some fonts may not include any sans-serif at all, some may include only an incomplete range of sans-serif (only Latin, no Greek), some may be missing bold Script and Fraktur, and some may be missing lowercase Script or lowercase and numerals in Blackboard Bold.

Besides missing some alphabets, some fonts may also provide some additional alphabets, such as an alternative italics, or a different style of Script or Calligraphic. Typically, these alphabets will have to be accessed via OpenType features using numbered stylistic sets.

As mentioned above, how much incompleteness is tolerable depends on your usage. If you are typesetting physics, you may well be interested in having a bold sans-serif alphabet for typesetting tensors, but you may need them only a few times in a series of books. In such cases, you may ask if you really need Greek for tensors, or if you can do with Latin only. And if you do need Greek for tensors, you may ask if you really need lowercase Greek, or if you can do with uppercase Greek only.

Depending on your requirements, your choices of math fonts providing the required alphabets may be limited, or you may be able to avoid the limitations. Finally, you may also consider substituting another font for certain math alphabets.

Taking advantage of stylistic sets or range substitutions depends on support by macro packages, but such features are already provided (for LaTeX) by the `unicode-math` and `fontspec` packages (on top of `luaotfload`) [26, 27, 28, 29].

4.3 Choosing typefaces for math alphabets

Unicode math combines a number of different shapes of math alphabets into a single font, including a

[1] However, the updates of Latin Modern Math and TeX Gyre Math in October 2012 have made them much more complete than their original releases.

matching serif and sans-serif typeface, a Script or Calligraphic, a Fraktur or Blackletter, a Blackboard bold, and a typewriter design (which we will ignore).

In the case of comprehensive font families, such as Latin Modern or Lucida, the choice of matching typeface designs will be obvious, as there is already a set of serif, sans-serif, and other font shapes that have been designed to be used together.

In other cases, however, choosing a set of matching typeface designs leads to a non-trivial design question: What is the proper choice of a sans-serif to be combined with a given serif font?

For a Times-like serif font (as in XITS Math), Arial or Helvetica may be an obvious choice of a sans-serif font (although this is debatable), but what should be used with Palatino or Euler? Should the sans-serif be another Hermann Zapf design (such as Optima)? What should be used with Minion? Should the sans-serif be another Adobe design (such as Myriad)? Or should it be something entirely different?

Should the sans-serif be a matching design with similar characteristics or should it be a contrasting design? How much similarity is needed to achieve consistency? How much contrast is needed to make individual letters clearly distinguishable?

Answers to such fundamental design questions may not be clear or obvious, but at some point font designers or developers will have to make a choice, or choose not to make a choice.

In some cases, such as for Minion Math or Neo Euler, decisions have been deliberately left open, leaving the fonts incomplete without any sans-serif alphabets. In other cases, such as for Asana Math (derived from `pxfonts` and `cbgreek`), decisions seem to have been taken based on what was available or which sans-serif fonts offered a suitable set of Greek besides Latin.

Besides the choice of sans-serif, similar design decisions may arise for the choice of Script, Calligraphic, Fraktur, or Blackboard Bold designs.

For Script, Calligraphic, or Fraktur, there seems to be considerable agreement among different font designers regarding the typical look of these shapes. Several different fonts seem to be very similar in the overall style, although each is still different in its individual design, as was discussed in a separate article by Michael Sharpe [30].

For Blackboard Bold, however, some very different approaches have been favored by different designers. In some cases, such as Cambria Math or Minion Math, the Blackboard Bold design is derived from an open face version of the main serif font. In other cases, such as XITS Math, Lucida Math, and Latin Modern Math, the Blackboard Bold is a very different style (typically sans-serif), which may be unrelated to the main sans-serif font.

4.4 Choices of Script (Calligraphic)

Design choices of Script alphabets fall into several groups, being fairly similar within a group. One group favors a very embellished style of Script:

XITS Math	𝒜ℬ𝒞𝒳𝒴𝒵 𝒶𝒷𝒸𝓍𝓎𝓏
Asana Math	𝒜ℬ𝒞𝒳𝒴𝒵 𝒶𝒷𝒸𝓍𝓎𝓏
Lucida Math	𝒜ℬ𝒞𝒳𝒴𝒵 𝒶𝒷𝒸𝓍𝓎𝓏
TG Termes	𝒜ℬ𝒞𝒳𝒴𝒵 𝒶𝒷𝒸𝓍𝓎𝓏

Another group favors a restrained style of Script:

Cambria Math	𝒜ℬ𝒞𝒳𝒴𝒵 𝒶𝒷𝒸𝓍𝓎𝓏
LM Math	𝒜ℬ𝒞𝒳𝒴𝒵
Neo Euler	𝒜ℬ𝒞𝒳𝒴𝒵
TG Pagella	𝒜ℬ𝒞𝒳𝒴𝒵 𝒶𝒷𝒸𝓍𝓎𝓏

Finally, several fonts also provide a Calligraphic as an alternative to Script (usually for uppercase only):

XITS Math	𝒜ℬ𝒞𝒳𝒴𝒵 (StylisticSet=1)
Lucida Math	𝒜ℬ𝒞𝒳𝒴𝒵 (StylisticSet=4)

It is noteworthy that Latin Modern Math currently does not provide the traditional Calligraphic style from Computer Modern as an alternative set, but that might be added in the future.

4.5 Choices of Fraktur (Blackletter)

Design choices of Fraktur alphabets are also similar among different fonts:

XITS Math	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷
Asana Math	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷
Cambria Math	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷
TG Termes	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷
TG Pagella	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷
LM Math	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷
Neo Euler	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷

In this case, LM Math, TG Pagella, and Neo Euler are all based on the same design of Euler Fraktur, whereas TG Termes is based on another source.

The only exception is Lucida Math, which features a completely different style of Blackletter:

Lucida Math	𝔄𝔅ℭ𝔛𝔜ℨ 𝔞𝔟𝔠𝔵𝔶𝔷

This may be seen as a example that not every Blackletter font is equally well suited for use in math.

4.6 Choices of Blackboard Bold

Design choices of Blackboard Bold alphabets again fall into multiple groups. One group favors a serif design which is derived from the main serif font:

Cambria Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚
Asana Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚
Minion Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝟘𝟙𝟚
TG Termes	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚
TG Pagella	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚

Another group favor a sans-serif design which may be unrelated to the main sans-serif font:

XITS Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚
Lucida Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ
LM Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚

Finally, the designs of individual letters can vary significantly among different math fonts, and are an additional consideration in font choice. For example, some users may have fairly strong preferences regarding such details as to whether the stem or the diagonal of the letter 'N' is double-struck.

4.7 Design issues of math alphabets

Besides the high-level design questions regarding the choice of matching typefaces for math alphabets to be assembled in an OpenType math font, there also some low-level design questions regarding the glyph shapes of individual typefaces.

In particular, we may ask: How should an upright Greek look, and how should a bold sans-serif Greek look compared to a bold serif Greek?

Unicode defines a number of math alphabets, many of which are supposed to come with a complete set of Latin and Greek for upper- and lowercase. This applies to all 4 shapes of the main serif typeface and to 2 out of 4 shapes of sans-serif.

4.8 Design of upright Greek alphabets

Unlike Unicode math, traditional TEX math fonts did not provide a complete set of Greek in all shapes.

Whereas uppercase Greek, just like uppercase Latin, came in several different shapes, including serif and sans-serif versions, lowercase Greek was only available in italics and bold italics.

As it turns out, creating an upright version of lowercase Greek by removing the slant while reusing the same designs of the italic version (as for Latin Modern Math) does not guarantee good results.

Comparing the results to other designs clearly shows that some letters in the unslanted Greek appear unbalanced, in particular for γ, ν, π, ϵ.

LM Math (upright = unslanted)

αβγδεζηθικλμνξοπρςστυϕξψωϵϑφϱϖ
αβγδεζηθικλμνξοπρςστυϕξψωϵϑφϱϖ

Cambria Math (upright = designed)

αβγδεζηθικλμνξοπρςστυϕξψωϵϑϰφϱϖ
αβγδεζηθικλμνξοπρςστυϕξψωϵϑϰφϱϖ

Obviously, font projects aiming for good results will need to take glyph design of individual math alphabets seriously, at which point a skilled font designer may be needed in addition to a font developer working on the technical aspects of font assembly.

4.9 Design of sans-serif Greek alphabets

Besides the design of upright Greek letter shapes, the design of sans-serif Greek alphabets may pose another challenge to font developers.

Strictly speaking, lowercase Greek letter shapes do not have serifs at all, so whether a Greek typeface design matches the look of a serif or sans-serif largely depends on matching the typical proportions and the typical stroke thickness of such fonts.

Usually, a sans-serif design exhibits a uniform stroke thickness, whereas a serif design exhibits contrast between thin or thick strokes, but the amount of contrast may vary between different fonts.

For the purposes of typesetting physics, letters from serif and sans-serif alphabets may be used next to each other to distinguish between different types of entities (vectors or tensors) by subtle differences in font shape (serif or sans-serif).

If the serif font exhibits a high contrast (as in the case of XITS Math), it is easy to tell apart from a sans-serif font, but if the serif font has a fairly uniform thickness itself (as in the case of Lucida Math), it becomes difficult to tell which one is which.

Lucida Math

αβγδεζηθικλμνξοπρςστυϕξψωϵϑϰφϱϖ
αβγδεζηθικλμνξοπρςστυϕξψωϵϑϰφϱϖ

XITS Math

αβγδεζηθικλμνξοπρςστυϕξψωϵϑϰφϱϖ
αβγδεζηθικλμνξοπρςστυϕξψωϵϑϰφϱϖ

Depending on the characteristics of the font, design of a clearly distinct sans-serif Greek may depend on more factors than just stroke thickness and may also require further adjustments to glyph shapes.

4.10 Technical issues regarding font metrics

Finally, besides achieving completeness and finding solutions to various design issues, there remain some technical issues to consider.

Most notably, there is an important difference in how glyph metrics are stored in OpenType math fonts as opposed to traditional TEX math fonts, and how those glyph metrics are interpreted in OpenType math engines following the reference behavior, such as LuaTEX (as opposed to XƎTEX).

In traditional TEX fonts, the actual glyph width used to be represented by the combination of the nominal width and the italic correction, but in Open-

Type fonts, the italic correction is disregarded, and only the nominal width is taken into account.

When converting traditional TEX math fonts to OpenType, it becomes necessary to adjust the glyph metrics to match the interpretation in OpenType math engines to ensure proper rendering in LuaTEX, while sacrificing the rendering in current versions of XETEX. (As of 2012, work on improving math typesetting in XETEX has been ongoing, so hopefully both engines will eventually adopt the same interpretation of glyph metrics.)

In recent developments, several font projects besides Cambria Math have adopted the OpenType interpretation of glyph metrics, such as Lucida Math and XITS Math, while others such as Latin Modern remain to be revised. Hopefully, other font projects will eventually follow to ensure consistent behavior when switching between different OpenType math fonts or different typesetting engines.

5 Conclusion

In this paper, we have reviewed recent progress in OpenType math font development as well as the many challenges faced by font developers of OpenType math fonts, including completeness of math symbols and math alphabets, design issues, and technical issues regarding the glyph metrics.

While significant progress has been made in recent years, resulting in recent or upcoming releases of several important OpenType math font projects, math font development remains a challenging task and more work remains to be done on developing new fonts or improving existing fonts.

References

[1] Barbara Beeton: Unicode and math, a combination whose time has come: Finally! *TUGboat*, 21(3), 174–185, 2000. `http://tug.org/TUGboat/tb21-3/tb68beet.pdf`

[2] Barbara Beeton, Asmus Freytag, Murray Sargent: Unicode Support for Mathematics. Unicode Technical Report UTR#25. 2001. `http://www.unicode.org/reports/tr25/`

[3] Microsoft Typography: OpenType specification, version 1.6, 2009. `http://www.microsoft.com/typography/otspec/`

[4] Murray Sargent: High-quality editing and display of mathematical text in Office 2007. `http://blogs.msdn.com/murrays/archive/2006/09/13/752206.aspx`

[5] John Hudson, Ross Mills: Mathematical Typesetting: Mathematical and scientific typesetting solutions from Microsoft. Promotional Booklet, Microsoft, 2006. `http://www.tiro.com/projects/`

[6] Ulrik Vieth: Do we need a Cork math font encoding? *TUGboat*, 29(3), 426–434, 2008. Reprinted in *MAPS*, 38, 3–11, 2009. `http://tug.org/TUGboat/tb29-3/tb93vieth.pdf` `http://www.ntg.nl/maps/38/02.pdf`

[7] Ulrik Vieth: OpenType Math Illuminated. *TUGboat*, 30(1), 22-31, 2009. Reprinted in *MAPS*, 38, 12–21, 2009. `http://tug.org/TUGboat/tb30-1/tb94vieth.pdf` `http://www.ntg.nl/maps/38/03.pdf`

[8] Jonathan Kew: XETEX Live. *TUGboat*, 29(1), 151–156, 2008. `http://tug.org/TUGboat/tb29-1/tb91kew.pdf`

[9] Taco Hoekwater: Math in LuaTEX 0.40. *MAPS*, 38, 22–31, 2009. `http://www.ntg.nl/maps/38/04.pdf`

[10] Hans Hagen: Unicode Math in ConTEXt. *MAPS*, 38, 32–46, 2009. `http://www.ntg.nl/maps/38/05.pdf`

[11] Tiro Typeworks: Projects — Cambria Math. `http://tiro.com/projects.html`

[12] Apostolos Syropoulos: Asana Math Font. `http://www.ctan.org/pkg/asana-math`

[13] Khaled Hosny: XITS Fonts. `http://www.ctan.org/pkg/xits` `http://github.com/khaledhosny/xits-math`

[14] STIX Consortium: STIX Fonts. `http://www.ctan.org/pkg/stix` `http://www.stixfonts.org/`

[15] GUST: e-foundry. `http://www.gust.org.pl/projects/e-foundry`

[16] GUST: The `lm-math` font package. `http://www.ctan.org/pkg/lm-math`

[17] Johannes Küster: Minion Math 1.020. `http://typoma.de/en/fonts.html`

[18] Ulrik Vieth, Mojca Miklavec: Another incarnation of Lucida: Towards Lucida OpenType. *TUGboat*, 32(2), 169–176, 2011. `http://tug.org/TUGboat/tb32-2/tb101vieth.pdf`

[19] Karl Berry: Lucida OpenType fonts available from TUG. *TUGboat*, 33(1), 11, 2012.

http://tug.org/TUGboat/tb33-1/tb103lucida.pdf

[20] GUST: The tex-gyre-math font package. http://www.ctan.org/pkg/tex-gyre-math

[21] Khaled Hosny: Neo Euler Font. http://github.com/khaledhosny/euler-otf

[22] Hans Hagen, Taco Hoekwater, Volker RW Schaa: Reshaping Euler: A collaboration with Hermann Zapf. *TUGboat*, 29(3), 283–287, 2008. http://tug.org/TUGboat/tb29-2/tb92hagen-euler.pdf

[23] Tiro Typeworks: Fonts — Maxwell Math. http://tiro.com/fonts.html

[24] Barbara Beeton: The STIX Project: From Unicode to fonts. *TUGboat*, 28(3), 299–304, 2007. http://tug.org/TUGboat/tb28-3/tb90beet.pdf

[25] Will Robertson: Symbols defined by unicode-math. http://mirror.ctan.org/macros/latex/contrib/unicode-math/unimath-symbols.pdf

[26] Will Robertson: Unicode mathematics in LaTeX: Advantages and challenges. *TUGboat*, 31(2), 211–220, 2010. http://tug.org/TUGboat/tb31-2/tb98robertson.pdf

[27] Will Robertson: The unicode-math package. http://www.ctan.org/pkg/unicode-math http://github.com/wspr/unicode-math

[28] Will Robertson: The fontspec package. http://www.ctan.org/pkg/fontspec http://github.com/wspr/fontspec

[29] Khaled Hosny et al.: The luaotfload package. http://www.ctan.org/pkg/luaotfload http://github.com/khaledhosny/luaotfload

[30] Michael Sharpe: Math alphabets and the mathalpha package. *TUGboat*, 32(2), 164–168, 2011. http://tug.org/TUGboat/tb32-2/tb101sharpe.pdf

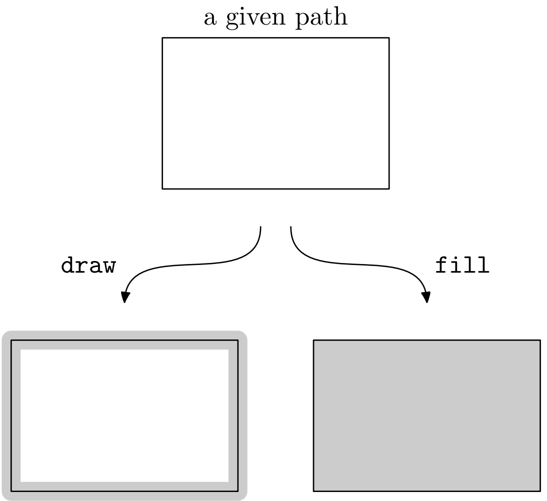
◇ Ulrik Vieth
Stuttgart, Germany
ulrik dot vieth (at) arcor dot de

From drawn to filled paths

Linus Romer

Abstract

While drawing given paths by pens is one ability of METAFONT, it is filled paths, the so-called *outlines*, which play the key role in workaday typefaces such as *Computer Modern*. The following article shows in short the relationship between drawn and filled paths.

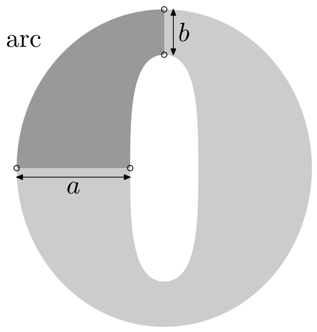


path is used as a route for a pen

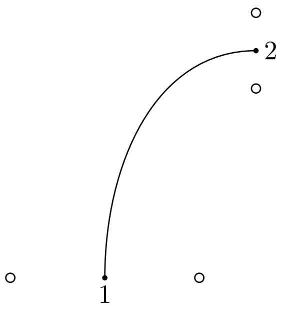
path is used as an outline of a shape

1 Construction of the letter "o"

For the sake of simplicity, this article concentrates on one exemplary shape: The letter "o". It is quite natural to divide this construction problem into four analogous parts, one of them highlighted below. Let us call such a part an *arc*. We will only look at the highlighted arc that connects the left side with the top of the "o".



We assume that we already know the beginning and the end of the arc as well as the widths a and b. The shape of the arc is then essentially determined by the shape of the path in the middle of the arc:



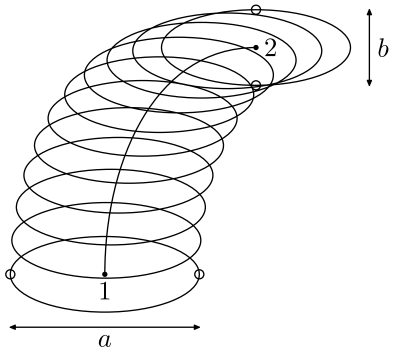
The shape of this path could be verbalized as follows: "Leave point 1 in upwards direction and make a nice curve to point 2, such that it would leave there traveling rightwards". The translation to METAFONT is much less redundant:

```
z1{up} ... z2{right};
```

With this midpath one can imagine already roughly the final shape of the arc. There are now several approaches that lead to the desired arc. The next sections present you six techniques that are available in the current METAFONT84 or were part of the old METAFONT78. Obsolete commands and methods of METAFONT78 are marked with a dagger (†).

2 Drawing with fixed elliptical pens

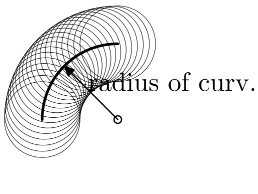
This primitive approach comes from the idea that one lets a pen glide over the given midpath. For every point on the path, the same shape is drawn on the surface (therefore "fixed"). In most cases one uses elliptical shaped pens.



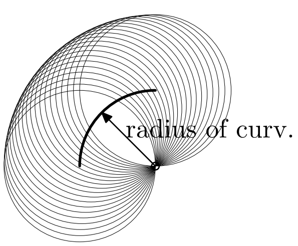
For the shown arc, the width and the height of the ellipse must correspond to a and b respectively.

```
pickup pencircle xscaled a yscaled b;
draw z1{up} ... z2{right};
```

The figure reveals already a big problem that can occur in this method: The final height of the arc at point 2 can become larger than b. However, if the diameter of the ellipse is smaller than the radius of curvature, one can use this method confidently. Put more simply, this means that only "big" pens lead to critical cases.

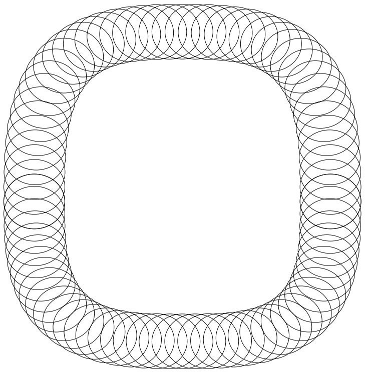


uncritical case



critical case

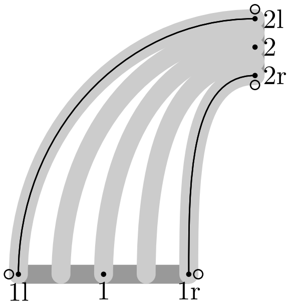
Donald E. Knuth used a fixed elliptic pen for all letters of the METAFONT logo:



As you can see, the used pen is nearly circular (the ellipses have flattening factor of only 0.1).

3 Double drawing with circular pens†

In METAFONT78 there was no direct way to fill contours. Instead, Knuth defined in [1] `ddraw`†, a predecessor of the `filldraw` command (see section 5). With `ddraw`† (for *double drawing*) one could simulate the filling–in of certain contours by stroking inside the contour several times.



To get our arc filled with `ddraw`†, we do not look at the midpath but rather apply its description on the points to the left and right and get the two outlining side paths.

```
z1l{up} ... z2l{right};
z1r{up} ... z2r{right};
```

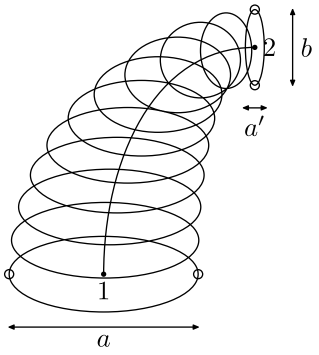
These side paths are now drawn with a normal circular pen (called `cpen`† in [1]). Of course, one has to shift the starting and ending points of these paths by the amount of the pen radius to get the correct borders.

The exact filling–in then works with a sufficiently large number of interpolated paths. The arc shown above clearly does *not* contain sufficiently many interpolated paths. METAFONT78 made sure that this could not happen in most cases. Nevertheless, if it happened for complicated paths one could increase the so-called *safety factor*†, which allowed more overlapping curves but also consumed more computational power.

4 Drawing with variable width pens

For more convenient coding, Knuth defined in [1] a high–level command based on `ddraw`†. With this mechanism, METAFONT78 could vary the pen width while drawing along a path.

```
† hpen;
† draw |a|1{0,1} ... |a'|2{1,0};
```

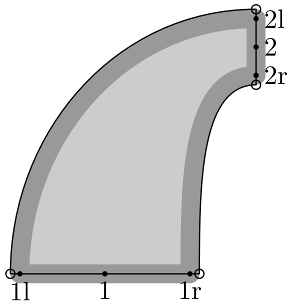


This kind of generalized drawing was only allowed for special pens. For our arc I have used an elliptical pen with fixed height *b* but variable horizontal width (called `hpen`†). Every drawing command with `hpen`† or its vertical counterpart `vpen`† became automatically translated to a `ddraw`† command, as detailed in [1].

The old *Computer Modern* base file at [3] indicates that Knuth gave up this approach around 1982: The obsolete `darc`† macro for drawing two connected arcs is still based on drawing with variable width pens, whereas the replacement `arc` macro directly uses `ddraw`†.

5 Filling and drawing the outlines

The whole `ddraw`† construction was somewhat cumbersome and inefficient, but there was no proper filling command available at that time in METAFONT78. The page description language *PostScript*, which supports the filling of contours, entered the community not before 1984. Knuth implemented a method to fill contours for the upcoming METAFONT84 and described the method in [4] and its implementation in [5]. Since then `ddraw`† has been replaced by `filldraw` (filling and drawing at the same time).



As with `ddraw`†, the lines which outline our arc are shifted a bit, filled as well as drawn.

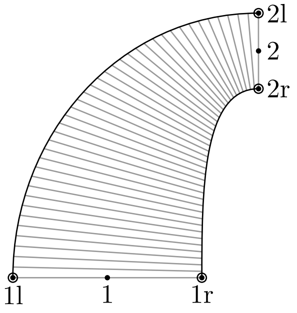
```
filldraw z1l{up} ... z2l{right}
 -- z2r{left} ... z1r{down} -- cycle;
```

The string `--` stands for straight paths and `cycle` for returning to the starting point.

6 Penstroking

For me, `penstroke` is *the* connecting link between the pen world and the outline world.

Double drawing or filldrawing with circular pens does not much depend on the size of the circular pens as long as they are much smaller than the radius of curvature. One can also make these circular pens infinitely small which alters the outlining contours to infinitely thin curves. The idea behind `penstroke` is the following: Corresponding points on the left and right part of the path are connected like a helix by straight and infinitely thin lines to fill the area in between. These lines may also be understood as pens with finite width and zero height, so-called *razor pens*.



The advantage of such penstroking over `ddraw`† and `filldraw` is that you do not have to worry about problematic pen offsets. In addition, vertices are now sharp and no longer rounded by circular pens.

Internally, `penstroke` is translated to a filling command. You can tell METAFONT to

```
penstroke z1e{up} ... z2e{right};
```

and according to [4] METAFONT automatically does

```
fill z1l{up} ... z2l{right}
 -- z2r{left} ... z1r{down} -- cycle;
```

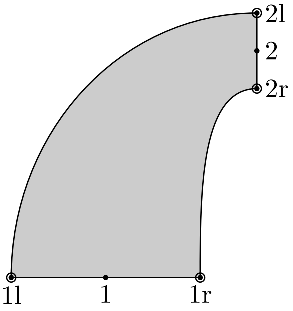
So you can *think* in pen terms and *do* outline filling at the same time, which is a big relief for programming.

7 Filling the outlines

Of course, you can directly use the more complex `fill` command:

```
fill z1l{up} ... z2l{right}
 -- z2r{left} ... z1r{down} -- cycle;
```

This makes sense when you want to design outlines that can hardly be understood as penstrokes (e.g. serifs).



8 Close relatives

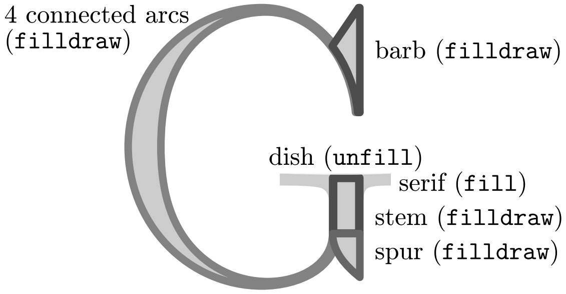
The methods that we have just seen in sections 3 through 7 are very similar to each other. I will try to point out their relationships in the following table:

	outlining pen = 0	outlining pen > 0
high–level command	`penstroke`	`draw` with variable width pens†
low–level command	`fill`	`filldraw` (`ddraw`†)

9 *Computer Modern* and pens

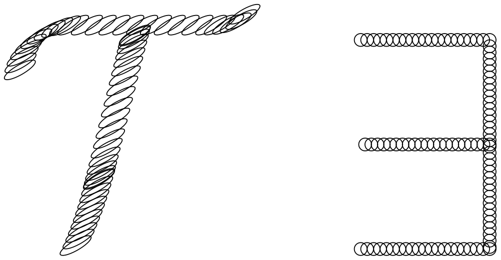
Computer Modern is the family of typefaces used by default by TEX and was first written using METAFONT78. The source was published in [2]. Knuth thereafter completely rewrote *Computer Modern* for METAFONT84. So the statement "*Computer Modern* is based only on pens" is surely true for the old sources, as METAFONT78 did not have any true filling of contours. For the current sources I claim: "*Computer Modern Roman* is a typeface based mainly on outlines. Pens are additionally used to soften vertices."

Let us look at the letter "G" of *Computer Modern Roman* to understand this claim better:



I have marked the several parts with their names and the way they are "painted" in the output. You can see that there is no standalone `draw` command used here. Note that `unfill` is just a special case of `fill` with white instead of black colour. The singular purpose of the `filldraw` command here is to make vertices rounded. In theory, this could also be solved by a more complex filling command.

Most parts of the huge *Computer Modern* family are based mainly on outlines. There are, however, also some parts which use pens exclusively:



The calligraphic uppercase letters (like the "$\mathcal{T}$" here) are for instance created from paths that are traced by a rotated elliptic pen. Mathematical symbols (like the "$\exists$" here) often make use of circular pens.

10 Conclusion

No matter whether you prefer drawn or filled paths, METAFONT can handle both. In any case the significance of METAFONT lies not in the power of pens but in its ability to parametrize fonts in large generality.

References

[1] Donald E. Knuth. *TeX and METAFONT: New directions in typesetting.* American Mathematical Society, 1979.

[2] Donald E. Knuth. *The Computer Modern family of typefaces.* Stanford University, 1980.

[3] Donald E. Knuth. cmbase.mf. `http://www.saildart.org/CMBASE.MF[MF,SYS]1`, 1982.

[4] Donald E. Knuth. *The METAFONTbook.* Addison–Wesley, 1986.

[5] Donald E. Knuth. *METAFONT: The Program.* Addison–Wesley, 1986.

◇ Linus Romer
Hirzlistrasse 3
Uznach, 8730
Switzerland
`linus.romer (at) gmx dot ch`

Glisterings

Peter Wilson

Catching fire, taking hold
All that glisters leaves you cold
No-one is near, no-one will hear
Your changeling song take shape
In Shadowtime.

Shadowtime, Siouxsie and the Banshees

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

Twixt the optimist and pessimist
The difference is droll:
The optimist sees the doughnut
But the pessimist sees the hole.

Optimist and Pessimist,
McLandburgh Wilson

1 Cutout windows

While winnowing my shelves and piles of books, journals, magazines, paper, etc., in anticipation of a move from the US to the UK I came across a *TUGboat* article by Alan Hoenig [2] in which he provides TEX code for creating an open window in the middle of a paragraph. An example of a paragraph with a cutout is in Figure 1. This was produced by:

```
\input{cutsty.tex}
\window{2}{0.4\textwidth}{0.25\textwidth}{5}
    This paragraph is set within the ...
...in a minipage in a \TUB\ \texttt{figure*}).
\endwindow
```

I tried out the code as given but found that it needed a tweak here and there to improve the spacing. Here is my version of Alan's code for rectangular cutouts, which can be used in both TEXed and LATEXed documents.[1] Most of my changes to the code are changes of name and argument specification to make it acceptable to both TEX and LATEX.

```
% cutsty.tex  Based on Alan Hoenig,
%  ‘TeX Does Windows --- The Conclusion’,
%  TUGboat 8:2, pp.211-215, 1987
```

First some counts, lengths, and boxes are needed (I have used `cut` as the start of each of these names to try and avoid clashes with other code):

```
\newcount\cutlines \newcount\cuttoplines
\newdimen\cutlftside \newdimen\cutrtside
\newtoks\cuta
\newcount\cutn
\newbox\cutrawtext \newbox\cutholder
\newbox\cutwindow \newbox\cutfinaltext
\newbox\cutaslice \newbox\cutbslice
\newdimen\cuttopheight
\newdimen\cutilgvs % glue or shift
```

The main user commands are `\window` and the accompanying `\endwindow`. The first of these takes four arguments as:

`\window{`⟨*top-lines*⟩`}{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*cut-lines*⟩`}`

where ⟨*top-lines*⟩ is the number of lines before the window cutout, ⟨*left*⟩ is the width of the text at the left of the window and ⟨*right*⟩ the width of the text at the right, and ⟨*cut-lines*⟩ is the number of lines used for the window (i.e., the height of the window). The macro gets a `\parshape` for the forthcoming text, gets and applies any vertical shift, opens a box for the text and then applies the `\parshape`.

```
\def\window#1#2#3#4{%
  \cuttoplines=#1\relax
  \cutlines=#4\relax
  \cutlftside=#2\relax
  \cutrtside=#3\relax
  \cuta={}%
  % calculate the \parshape spec
  \parshapespec
  % reset the these arguments
  \cuttoplines=#1\relax
  \cutlines=#4\relax
  % calculate and apply any vertical shift
  \cutshift \vskip-\cutilgvs
  % start a box for collecting the text
  \setbox\cutrawtext=\vbox\bgroup
  \parshape=\cutn \the\cuta}
```

The text, in the form of a single paragraph with a constant `\baselineskip` is put between the two `\...window` commands; in the case of LATEX you can, but don't need to, use a `window` environment instead.

The general scheme is to use a specially shaped paragraph which effectively splits the text into three sets of lines; those before the cutout; those that will form the cutout; and the rest. The lines forming the cutout are short while the others are full length. An example is shown in Figure 2. The final output is assembled from the top set of lines, the cutout lines combined in pairs, and the remainder. The final form of a paragraph with a cutout is shown in Figure 3.

```
\def\endwindow{%
  \egroup     % end \box\cutrawtex
  \parshape=0 % reset parshape
  \computeilg % find ILG using current font
  \setbox\cutfinaltext=
    \vsplit\cutrawtext
      to\cuttoplines\baselineskip
```

[1] Alan also gave code for creating arbitrary shaped holes.

This paragraph is set within the `window` environment. There are limitations on the `window` arguments and text. There must be at least one line of text above the window and if the number of lines specified for the opening exceeds the available lines then the text after the `window` environment will be moved down by an amount corresponding to the excess. A window will not extend into a second paragraph. The environment is effectively a box and will not break across a page boundary. There should be enough space at the left and right of the window for a few words on each side (don't try to make either of these zero in an attempt to have a window opening to the margin). There is usually not enough width to put a significant window into a column on a two-column page (this has been set in a minipage in a *TUGboat* `figure*`).

Figure 1: A generated window

If you have to have a cutout in a narrow column keep the words short. Use one or two or maybe one or more extra letters so that they may fit into the available area without too much odd spacing. If the words are hyphenatable this will help a lot as then a long one may be cut into two short bits.

Figure 2: Split window lines

```
\cuttopheight=\cutlines\baselineskip
\cuttopheight=2\cuttopheight
\setbox\cutholder=
  \vsplit\cutrawtext
    to\cuttopheight
% \cutholder contains the narrowed text
%  for window sides. Slice up \cutholder
%  into \cutwindow
\decompose{\cutholder}{\cutwindow}
\setbox\cutfinaltext=\vbox{%
  \unvbox\cutfinaltext\vskip\cutilgvs
    \unvbox\cutwindow%
    \vskip-\cutilgvs\unvbox\cutrawtext}%
\box\cutfinaltext}
```

A `\parshape` is used to specify quite general paragraph shapes [3, Ch. 14] or [1, Ch. 18]. Its $2n+1$ parameters specify the indentation and length of the first n lines in the following paragraph which must start immediately (no empty line after the parameters). The first parameter is n followed by n pairs of indentation and line length values. In general:
`\parshape` $n\ i_1\ l_1\ i_2\ l_2\ \dots\ i_n\ l_n$
If there are more than n lines then the specification for the last line ($i_n\ l_n$) is used for the rest of the lines in the paragraph.

If you have to have a cutout in a narrow column keep the words short. Use one or two or maybe one or more extra letters so that they may fit into the available area without too much odd spacing. If the words are hyphenatable this will help a lot as then a long one may be cut into two short bits.

Figure 3: Assembled window lines

`\parshapespec` calculates the `\parshape` parameters to generate a paragraph with ⟨*top-lines*⟩ full lines followed by ⟨*cut-lines*⟩ of length ⟨*left*⟩ alternating with ⟨*cut-lines*⟩ of length ⟨*right*⟩.

```
\def\parshapespec{%
  \cutn=\cutlines \multiply \cutn by 2
    \advance\cutn by \cuttoplines
    \advance\cutn by 1\relax
  \loop
    \cuta=\expandafter{\the\cuta 0pt \hsize}
    \advance\cuttoplines -1\relax
    \ifnum\cuttoplines>0\repeat
  \loop
    \cuta=\expandafter{\the\cuta
          0pt \cutlftside 0pt \cutrtside}%
    \advance\cutlines -1\relax
  \ifnum\cutlines>0\repeat
  \cuta=\expandafter{\the\cuta 0pt \hsize}}
```

An example paragraph at this stage of the process is in Figure 2.

The `\decompose{`⟨*narrow*⟩`}{`⟨*split*⟩`}` command takes a box ⟨*narrow*⟩ and for each pair of lines puts the first at the left and the second at the right of the box `{`⟨*split*⟩`}`. That is, it converts pairs of lines into single lines with text at the left and the right with a space between.

```
\def\decompose#1#2{%
  % loop over the windowed lines
  \loop\advance\cutlines -1
   % get a pair of lines
  \setbox\cutaslice=\vsplit#1 to\baselineskip
  \setbox\cutbslice=\vsplit#1 to\baselineskip
  % split into the two sides
  \prune{\cutaslice}{\cutlftside}
  \prune{\cutbslice}{\cutrtside}%
  % assemble into one line
  \setbox#2=\vbox{\unvbox#2\hbox
  to\hsize{\box\cutaslice\hfil\box\cutbslice}}%
  \ifnum\cutlines>0\repeat}
```

For the example in Figure 2 the `\decompose` macro converts the 6 narrow lines into the 3 cutout lines shown in Figure 3.

`\prune{`⟨*vbox*⟩`}{`⟨*width*⟩`}` is used to prune the glue that TEX puts at the end of a short `\parshape` line. It takes a `\vbox` containing a single `\hbox`, `\unvbox`es it, cancels the `\lastskip` and puts it in a box of ⟨*width*⟩ wide; a `\strut` is needed to keep the spacing consistent.

```
\def\prune#1#2{%
  \unvbox#1
  \setbox#1=\lastbox % \box#1 is now an \hbox
  \setbox#1=\hbox to#2{\strut\unhbox#1\unskip}}
```

`\cutshift` calculates the amount that the windowed paragraph must be raised, which is half a `\baselineskip` for each windowed line. (This is my addition).

```
\def\cutshift{%
  \cutilgvs=\cutlines\baselineskip
  \cutilgvs=0.5\cutilgvs}
```

`\computeilg` computes the interline glue in the windowed paragraph. This is the last macro so finish the file with an `\endinput`.

```
\def\computeilg{%
  \cutilgvs=\baselineskip
  \setbox0=\hbox{(}
    \advance\cutilgvs-\ht0
    \advance\cutilgvs-\dp0}
\endinput
```

Artwork or text may be placed in the cutout. How to do that is a very different problem and one that I am not intending to address here, but zero-sized pictures and headers or footers come to mind [4]. Perhaps solutions will have been published by the time this article appears.

?

Since the preceding was first written, the cutwin package [5] has appeared which lets you create variously shaped cutouts and place things in the resulting window.

References

[1] Victor Eijkhout. *TEX by Topic, A TEXnician's Reference.* Addison-Wesley, 1991. ISBN 0-201-56882-9. Available at `http://www.eijkhout.net/tbt/`.

[2] Alan Hoenig. TEX does windows — the conclusion. *TUGboat*, 8(2):211–215, 1987.

[3] Donald E. Knuth. *The TEXbook.* Addison-Wesley, 1984. ISBN 0-201-13448-9.

[4] Peter Wilson. Glisterings: Ornaments. *TUGboat*, 32(2):202–205, 2011.

[5] Peter Wilson and Alan Hoenig. Making cutouts in paragraphs, 2010. Available on CTAN in `macros/latex/contrib/cutwin`.

◇ Peter Wilson
20 Sovereign Close
Kenilworth, CV8 1SQ
UK
herries dot press (at)
earthlink dot net

Stubborn leaders and juggling boxes: A slightly unusual table of contents

Boris Veytsman

Abstract

A macro for typesetting an unusual table of contents is introduced. The history of development of this macro is described in detail showing the refinement of requirements and refactoring of the code. The TeX boxes, their measuring and inspection are discussed.

1 Introduction

Many publishers that accept manuscripts in TeX or LaTeX put their styles on CTAN; see the (certainly incomplete) list at `http://www.ctan.org/keyword/publishers`. However, when a journal uses TeX internally, there are also macros not publicly released. These macros are used in typesetting covers, journal title pages, issue and volume tables of contents, indices etc. They are specific to the journal. Without major changes they can be used only by the people who want to create another one with exactly the same look and feel. For some reason this practice is frowned upon by the publishers. Still, sometimes the tricks in these macros might be of interest to TeXnicians. Thus extracting them and publishing separately seems to be worth doing.

I recently released the package (Veytsman, 2012) for typesetting articles for the journal *Res Philosophica*, `http://www.resphilosophica.org`. I also wrote a package to typeset the covers and technical information for the journal. In this article I describe a somewhat non-trivial macro used in the table of contents of the journal.

An important (maybe *the* most important) part of the work of a programmer is the understanding of the requirements. Thus I would like to tell the history of this macro: how it was designed and re-designed after studying the samples.

2 The problem and a naïve solution

The entries of the journal's table of contents were typeset like the ones in Figure 1: the article title, then the dots connecting the entry to the page number, following by the page number itself, and the authors on the next line in italics. The dots are called *leaders* in TeX. LaTeX has a nice macro `\dotfill` which makes infinitely stretchable leaders.[1] With this macro our TeX source looks like this:

```
% #1 is the title, #2 is the author(s)
% #3 is the page number
\def\settocentry#1#2#3{\par
  #1\space\dotfill\space#3\par
  \textit{#2}\par\medskip}
\parskip=0pt\parindent=0pt
\settocentry{Article title}{A. U. Thor}{5}
```

Article title 5
A. U. Thor

Figure 1: A simple TOC entry

Article title 5
A. U. Thor

A very very very very very very long article
title 9
A. N. Other

Figure 2: Failure of the naïve solution

The problem is solved!

Unfortunately, it is not so simple. What happens if the article title is very long? The result is shown in Figure 2. The leaders start from the second line of the title, separating it from the author, which just does not look right. This situation occurs not infrequently because the width of the journal pages is rather small (6 in.).

An examination of the samples showed that the designer wanted (1) the title and the authors to occupy no more than 60% of the text width, and (2) the leaders to *always* start from the first line of the entry, even if the title occupied two or more lines. Of course, there are some arguments against this design. One can object to the fact that the title is "broken" by the leaders. On the other hand, the eye looking for a title or authors moves in the vertical direction and probably disregards the intervening leaders, so this objection might not be too strong after all. Anyway, this was the design which I was given to implement.

3 A less naïve solution

A less naïve idea is the following. Let us typeset the title in a box of certain width, say `\entrywidth`, with the baseline at the first line. If it occupies one line, we throw away the box and use the solution from the previous section. However, if it occupies two or more lines (which we can check by measuring its depth), we align it with the leaders and page number.

To make our code simpler, let us split the main macro into two parts: setting the title and setting the authors. We also put the page into a global

[1] While most of our code works in plain TeX, we do use some LaTeXisms like `\dotfill`, `\itshape`, etc., which are easy to emulate in other formats.

Article title 5
A. U. Thor

A very very very very very 9
very long article title
A. N. Other

Figure 3: A less naïve solution

`\articlepage` to be used by the `\setarticletitle` macro — mostly to avoid cluttering the definitions:

```
\def\settocentry#1#2#3{%
  \gdef\articlepage{#3}%
  \setarticletitle{#1}\par
  \setarticleauthors{#2}\par\medskip}
```

To set the article title we put it in a `\vtop` box `\titlebox`. Since it is `\vtop`, the baseline of the box coincides with the baseline of the first line, and to check the presence of the second line we need to measure its *depth* rather than height. Also, we add `\strut` at the end of the box to keep the distance between the lines uniform:

```
\newdimen\tocboxwidth
\tocboxwidth=0.6\textwidth
\newbox\titlebox
\def\setarticletitle#1{\par
  \setbox\titlebox=\vtop{%
    \hsize=\tocboxwidth\relax
    #1\strut\par}%
  \ifdim\dp\titlebox<\baselineskip\relax
  % the box is one line long, forget it
    #1\space\dotfill\space\articlepage
  \else % The box has more than one line
    \vtop{\hsize=\textwidth\leavevmode
      \box\titlebox\space\dotfill
      \space\articlepage}%
  \fi}
```

We also set the authors in a box of the same width:

```
\def\setarticleauthors#1{%
  \vtop{\hsize=\tocboxwidth
    \strut\itshape#1\strut}}
```

Is our problem solved? Unfortunately not. First, titles are usually no more than two lines long. Therefore it would be better to typeset them ragged right. Second, often the article title is split at the logical break, for example, after a colon. Therefore we can expect an input like this:

```
\parskip=0pt\parindent=0pt\raggedright
\settocentry{Article title}{A. U. Thor}{5}
\settocentry{A very very very very
  longwinding article
  title}{A. N. Other}{9}
\settocentry{The state of the art:\\
  New developments and results}%
  {C. O. R. Respondent}{21}
```

Article title 5
A. U. Thor

A very very very very   9
longwinding article title
A. N. Other

The state of the art: 21
New developments and
results
C. O. R. Respondent

Figure 4: Failure of the less naïve solution

As seen in Figure 4, this input breaks our TOC. The leaders start at the end of the box, leaving an ugly gap. Note that if not for the leaders, our solution would be perfect: the page numbers are aligned with the first lines of the titles.

4 The current solution

Whenever I hit a wall in my TeX hacking, I try to find a clue in the book by Eijkhout (2007). This worked for me again. In §5.9.6 there I found a nice trick using `\lastbox`. This command deletes the last box from the vertical list and makes it available for inspection and retypesetting. If we apply this command iteratively, we "pick apart" the list line by line until we hit the top. Eijkhout used it to typeset the lines in a paragraph in a way that depended on their natural width. He put the paragraph in a box, and then inspected each line starting from the last one, re-typesetting if necessary (see also Eijkhout, 1990).

This trick provides the solution to our problem. Indeed, let us typeset the title in a box of width `\tocboxwidth`. Then let us pick this box apart, moving the lines into another box. When we hit the last line from the bottom (i.e. the first line from the top), we break this line and re-typeset it with the leaders and the page number.

There are two additional problems to be solved. First, a vertical list has not just boxes, but also glue and penalties. Eijkhout found that he needs to put `\unskip` and `\unpenalty` in his loop after getting the last box. LaTeX `\raggedright` adds additional skips to the lines, so after some meditation over the kernel code and trials I put there a longer incantation with one `\unpenalty` and three `\unskip`s. Second, how do we know that we have hit the top line? The only way to find it is on the *next* iteration of the loop, where `\lastbox` returns an empty box. Thus we need to postpone moving the line to the result until we inspect the next one. Therefore the final algorithm includes four boxes: `\trialbox` to initially typeset the title, `\resultbox` to put the result in, `\lastlinebox` for the line read last, and `\prevlinebox` for the previously read line. The algorithm is the following:

1. We typeset the title into a `\trialbox`.
2. On each iteration of the loop we take the last line of this box and put it in the box `\lastlinebox`.
3. If this box is empty, we have hit the top. Then we break `\prevlinebox`, add leaders and page number and put on the top of `\resultbox`.
4. Otherwise we put `\prevlinebox` in `\resultbox`, put `\lastlinebox` in `\prevlinebox` and repeat the loop.

A final note before we start going into the code: when we add vertical boxes, we need to check first whether they are empty, otherwise we introduce unwanted vertical space.

The `\setarticle` macro creates `\trialbox` and then typesets the `\resultbox`:

```
\newbox\trialbox
\newbox\resultbox
\newbox\lastlinebox
\newbox\prevlinebox
\def\setarticletitle#1{%
  \setbox\trialbox=\vtop\bgroup
    \hsize=\tocboxwidth\relax
  \strut#1\strut\par\getlastline\egroup
  \box\resultbox}
```

The heart of our algorithm is the `\getlastline` macro. It implements iteration in a very TEXish way: by recursively calling itself.

```
\def\getlastline{%
  % Reading the last line
  \global\setbox\lastlinebox=\lastbox
  \ifvoid\lastlinebox % We hit the top;
                      % construct the
                      % result and
                      % finish
    \global\setbox\resultbox=\vtop
      \bgroup\hsize=\textwidth
         \hbox to \textwidth
         {\unhbox\prevlinebox
           \unskip\unpenalty\space
           \dotfill\space\articlepage}%
         \ifvoid\resultbox\else
         \box\resultbox\fi
       \egroup
  \else % We did not hit the top yet
    \unskip\unpenalty\unskip\unskip
    \ifvoid\prevlinebox\else
       \global\setbox\resultbox=\vtop
       \bgroup
          \box\prevlinebox
           \ifvoid\resultbox\else
           \box\resultbox\fi
       \egroup
    \fi
    \global\setbox\prevlinebox
       \box\lastlinebox
    {\getlastline}% Recursion!
  \fi}
```

Article title 5
A. U. Thor

A very very very very 9
longwinding article title
A. N. Other

The state of the art: 21
New developments and
results
C. O. R. Respondent

Figure 5: The current solution

The results are shown in Figure 5. As one can see, now the leaders behave correctly.

5 Conclusion

Our article shows how TEX boxes provide a rich environment for a rather complex typesetting problem. The `\lastbox` command is a powerful tool which can be used for many non-trivial tasks.

References

Eijkhout, Victor. "Unusual Paragraph Shapes". *TUGboat* **11**, 51–53, 1990. `http://www.tug.org/TUGboat/tb11-1/tb27eijkhout.pdf`.

Eijkhout, Victor. *TEX by Topic.* Lulu, 2007. `http://eijkhout.net/texbytopic/texbytopic.html`.

Veytsman, Boris. *Typesetting Articles for* Res Philosophica, 2012. `http://mirror.ctan.org/macros/latex/contrib/resphilosophica`.

◇ Boris Veytsman
School of Systems Biology &
Computational Materials
Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
borisv (at) lk dot net
http://borisv.lk.net

The Treasure Chest

This is a list of selected new packages posted to CTAN (`http://ctan.org`) from August to October 2012, with descriptions based on the announcements and edited for brevity.

Entries are listed alphabetically within CTAN directories. A few entries which the editors subjectively believed to be of especially wide interest or otherwise notable are starred; of course, this is not intended to slight the other contributions.

CTAN remains as vibrant and important to the TEX community as it has ever been, but the *TUGboat* editors have received no feedback on this column in many years, and electronic notifications of CTAN updates are now widespread and (of course) much more timely. So, if you find this column useful, please drop us a line; otherwise, we will likely drop it in future issues.

◇ Karl Berry
`tugboat (at) tug dot org`
`http://tug.org/ctan.html`

biblio

`francais-bst` in `biblio/bibtex/contrib`
Two `natbib`-compatible styles for standard French bibliography typesetting.

`mkbib` in `biblio/bibtex/utils`
GTK+ BIBTEX bibliography creator.

fonts

`biolinum-type1` in `fonts`
(pdf)LATEX support for the Biolinum fonts.

`hacm` in `fonts`
The alphabet of the constructed language Arka.

*`lm-math` in `fonts`
Latin Modern OpenType math font.

`libertine-type1` in `fonts`
(pdf)LATEX support for the Linux Libertine fonts.

graphics

`hobby` in `graphics/pgf/contrib`
Hobby's algorithm for generation of Bezier curves, for PGF/Ti*k*Z.

`jmakepdfx` in `graphics/pgf/contrib`
Java interface to Ghostscript for conversion of PDF to PDF/X.

`setdeck` in `graphics/pgf/contrib`
Provides LATEX commands to typeset cards from the game set.

`smartdiagram` in `graphics/pgf/contrib`
Create smart diagrams from lists of items.

`svg` in `graphics`
Include `.svg` files made by Inkscape.

`tikzposter` in `graphics/pgf/contrib`
Making posters with Ti*k*Z.

info

`examples/Presentations_en` in `info`
Examples from the book *Presentations with LATEX* by Herbert Voß.

`latex-sciences-humaines` in `info`
French book about LATEX and humanities, 270 pp.

macros/generic

`catcodes` in `macros/generic`
Deal with category code switching.

`langcode` in `macros/generic`
Adjust language-dependent settings.

`plainpkg` in `macros/generic`
Support for making generic packages.

macros/latex/contrib

`abraces` in `macros/latex/contrib`
Generalized constructions similar to `\overbrace` and `\underbrace`.

`actuarialangle` in `macros/latex/contrib`
Typeset nice looking "angles" in present value of an annuity symbol.

`adhocfilelist` in `macros/latex/contrib`
Shell script to output a list of LATEX `\Provides...` contents, possibly filtered according to various criteria.

`bohr` in `macros/latex/contrib`
Create Bohr models of atoms.

`calcage` in `macros/latex/contrib`
Calculate age in years (based on `datenumber`).

`chkfloat` in `macros/latex/contrib`
Check for floats too far from their origin.

`counttexruns` in `macros/latex/contrib`
Count how often a LATEX document is compiled.

`designcon` in `macros/latex/contrib`
Tools for developing DesignCon conference papers using LATEX and LYX.

`eledform` in `macros/latex/contrib`
Formalize textual variants in critical editions.

*`eledmac` in `macros/latex/contrib`
Successor to `ledmac` and `ledpar`.

`exsheets` in `macros/latex/contrib`
Create exercises or questionnaires, with solutions.

`filedate` in `macros/latex/contrib`
Access and compare info and modification dates.

`fnumprint` in `macros/latex/contrib`
Typeset a number as a word or Arabic numerals.

* `imakeidx` in `macros/latex/contrib`
Package for producing multiple indexes.

`libertine` in `macros/latex/contrib`
Compatibility package for existing documents to use `biolinum-type1` and `libertine-type1`.

`loops` in `macros/latex/contrib`
Generic looping macros.

`ocgx` in `macros/latex/contrib`
Use PDF feature 'Optional Content Groups' (OCG) without JavaScript.

`physics` in `macros/latex/contrib`
Typesetting equations in vector calculus and linear algebra via Dirac notation.

`pkuthss` in `macros/latex/contrib`
Template for dissertations at Peking University.

`pxcjkcat` in `macros/latex/contrib`
Change CJK category settings in upLaTeX.

`resphilosophica` in `macros/latex/contrib`
Class for typesetting articles for the journal *Res Philosophica*.

`scrjrnl` in `macros/latex/contrib`
Class to typeset diaries or journals.

`ulthese` in `macros/latex/contrib`
LaTeX class to prepare theses for the Université Laval, Québec.

`uspatent` in `macros/latex/contrib`
LaTeX class and LyX layout files to support writing US patent applications.

`venndiagram` in `macros/latex/contrib`
Create Venn diagrams with Ti*k*Z.

macros/latex/contrib/biblatex-contrib

`biblatex-caspervector` in `m/l/c/biblatex-contrib`
`biblatex` support for Chinese LaTeX users.

`biblatex-publist` in `m/l/c/biblatex-contrib`
`biblatex` support for publication lists.

`biblatex-trad` in `m/l/c/biblatex-contrib`
`biblatex` support for traditional BibTeX styles: `plain`, `abbrv`, `unsrt`, `alpha`.

`oscola` in `m/l/c/biblatex-contrib`
`biblatex` support for the Oxford Standard for the Citation of Legal Authorities.

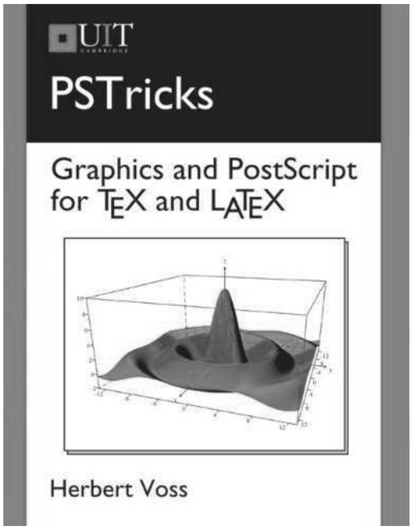
macros/plain

`plipsum` in `macros/plain/contrib`
'Lorem ipsum' paragraphs generator for plain TeX, with many f-ligatures.

Book review: *PSTricks: Graphics and PostScript for TeX and LaTeX*

Boris Veytsman

Herbert Voß, *PSTricks: Graphics and PostScript for TeX and LaTeX*. UIT Cambridge, 2011. 928 pp. Paperback, US$64.99. ISBN 9781906860134.`http://www.uit.co.uk/BK-PSTricks`.



Many years ago when I was a student at the Theoretical Physics department of Odessa University, USSR, I regularly both praised and cursed *Introduction to the Theory of Quantum Fields* by N. N. Bogolyubov and D. V. Shirkov. This was a large volume (over five hundred pages) covering many topics not described anywhere else. It was one of the most essential books for students at that time. On the other hand, it was plagued by typos, misspellings and outright errors. You could not just take a formula from this book and apply it for your work. You needed to find a pencil, a thick stack of blank sheets and follow the derivation, correcting authors' mistakes and getting the right answer after much toil and sweat. According to an urban legend, when the authors taught the course on Quantum Fields, every student was required to present a newly found error in the textbook to get a passing grade — a Soviet variation of the famous DEK's checks.

I recalled this experience when I've read the new book by Herbert Voß. The book describes PSTricks — an immensely capable system of graphics combining the power of TeX and PostScript. While many people nowadays use Ti*k*Z (an alternative system), PSTricks remains the strongest one, in my opinion. The beautiful examples throughout the book and on the PSTricks home page (`http://tug.org/PSTricks/main.cgi/`) demonstrate the vast possibilities of this system. Today PSTricks can be easily used to produce PDF files, which used to be one of the most important advantages of Ti*k*Z.

Unfortunately, the official manual of PSTricks is rather old, and documentation for many important parts of the system is dispersed among multiple guides for individual packages. Moreover, some of the documentation remains available only in French or German, which is frustrating for many users.

This book strives to be a comprehensive guide to the complex world of PSTricks. The author describes in detail the base system and spends some time on the contributed packages, in a style resembling the *LaTeX Companion* series of books. For some packages, such as `pst-geo`, this book provides English documentation for the first time ever (after the time spent with `pst-geo` I feel I almost can read French). This book will be useful for both novices and experts. Herbert Voß is the leading developer of PSTricks, and his knowledge of the system is unsurpassed. I am an old user of PSTricks (full disclosure: the book mentions a contributed package co-authored by me), but even for me some of the chapters were eye-openers, like the very thorough description of three-dimensional plotting, or the detailed discussion of `pscustom` objects.

PSTricks, despite its long history, is still being actively developed, so the information in the printed book quickly becomes obsolete (as mentioned above, Herbert himself contributes to the obsolescence of his manual). At the moment I am writing this review, the PSTricks web site announces several updates made just two days ago. Nevertheless a book like this is important: it serves as a compass in the ocean of documentation. In many cases even the fact that a package solving some problem exists can be pivotal. Thus buying this book is a good investment for anybody working with PSTricks.

Unfortunately the comprehensiveness and indispensability are not the only common features between *Graphics and PostScript for TeX and LaTeX* and *Introduction to the Theory of Quantum Fields*. Like the latter book, the former has too many typos for the user's comfort. Typos in a mathematical formula are rather nasty; typos in a program code can also be very bad. I started to list the typos for this review, but soon found out there were just too many of them in the book. There was an effort to crowd-source the proofreading of the manuscript: the volunteers from the `pstricks` mailing list got a chapter each to find the typos (another full disclosure: I worked on proofreading Appendix A), but this obviously did not work as well as a professional copy editor would. Even the book index has problems: on page 886 LaTeX packages are indexed under "Pakete".

In many cases the language of the book might be improved, and sometimes I could not figure out the meaning of the author's text.

The style of the book is uneven. Well written chapters coexist with somewhat haphazard texts. Sometimes the commands are used in the examples, but explained only several pages later, if ever. For example, the code on page 174 uses `\pst@@@vlabel` and `\pst@@@hlabel` macros which are not explained anywhere in the text. The book is illustrated by extensive examples. Sometimes the code for these examples is printed in the book; sometimes the reader is referred to the web site. The same can be said about the typographic style of the book: some chapters use marginal notes, some not. Some use "dangerous bend" symbols, some not. This might be expected in a collection of chapters by different authors, but a monograph should have some stylistic uniformity.

The book could be improved by a chapter introducing PostScript language for the beginners. The book does include a list of PostScript commands in Appendix B — a useful resource but by no means suitable as the first reading about the subject. The knowledge of PostScript is not necessary for the basic usage of PSTricks, but is helpful for the advanced tricks. Appendix D on PDF would probably be better as a chapter — with the extended discussion of transparency and other PDF-related features.

There are probably many other improvements that could be made to this book.

The urban legend mentioned in the beginning of this review says that at some point Bogolyubov and Shirkov collected the writeups of their students and used them for the corrected edition of their book. I do not know whether this is true. However, the fourth Russian edition on my shelf today has many fewer typos and mistakes than the early ones.

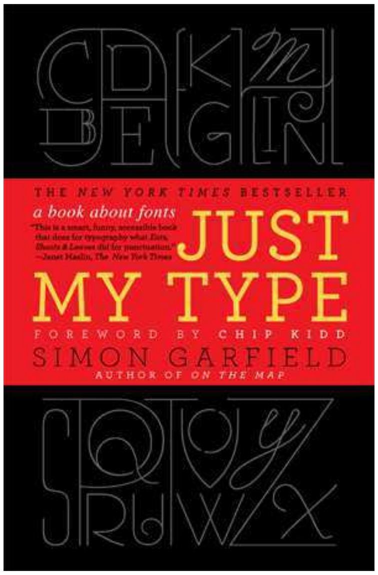
This book by Herbert Voß has had five editions in German. This is the first English one. I wish the author would make a new edition correcting the problems I've described. However, even now this is the most definitive English book on PSTricks — albeit very frustrating at times.

◇ Boris Veytsman
School of Systems Biology &
Computational Materials
Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
borisv (at) lk dot net
http://borisv.lk.net

Book review: *Just My Type: A book about fonts*

David Walden

Simon Garfield, *Just My Type: A book about fonts*. Gotham Books, 2011. 356 pp. Hardcover, US$27.50. ISBN 9781592406524. (Also in paperback.)


THE *NEW YORK TIMES* BESTSELLER
a book about fonts
"This is a smart, funny, accessible book that does for typography what *Eats, Shoots & Leaves* did for punctuation."
—Janet Maslin, *The New York Times*
JUST MY TYPE
FOREWORD BY CHIP KIDD
SIMON GARFIELD
AUTHOR OF *ON THE MAP*

I was excited when I heard about the book *Just My Type*. I have become increasingly interested in fonts as I have worked with TeX over the past 15 years, and in the past several years I have checked a couple of dozen books about type, type design, and type designers out of libraries and skimmed through them. Thus, when this book was published, I immediately bought a copy thinking it would be neat to read a book that written for a lay audience. As I initially dipped into the book, reading sections here and there, I enjoyed what I read. However, when I tried to read the book carefully cover to cover to do this review, my appreciation of the book wavered. It was hard going — too much information with too little logical connection over the book's approximately 350 pages.

The book is a collection of essays (an introduction and 22 chapters averaging a little under 14 pages each) with titles such as:

- Legibility vs Readability
- Baskerville is Dead (Long Live Baskerville)
- Can a Font be German or Jewish
- Pirates and Clones

Interleaved among the chapters are a dozen so-called "FontBreaks", each three or so pages long and named after a font or class of fonts, for example

- Gill Sans
- Mrs Eaves & Mr Eaves
- Moderns, Egyptians, and Fat Faces
- The Interrobang

However, any of the chapters or FontBreaks may have information about fonts, font design, font designers, use of fonts, theories of fonts, and so on. The FontBreaks are also essays, albeit shorter than the chapter essays; and there is really no thread that ties these 35 chapter and FontBreak essays together or orders them in some logical way.

Thus, I think the book will appeal most to people who don't have time or interest in pushing through a continuous coherent text and rather only seek to read one short type-related story at a time. It may be ideal for bedtime reading.

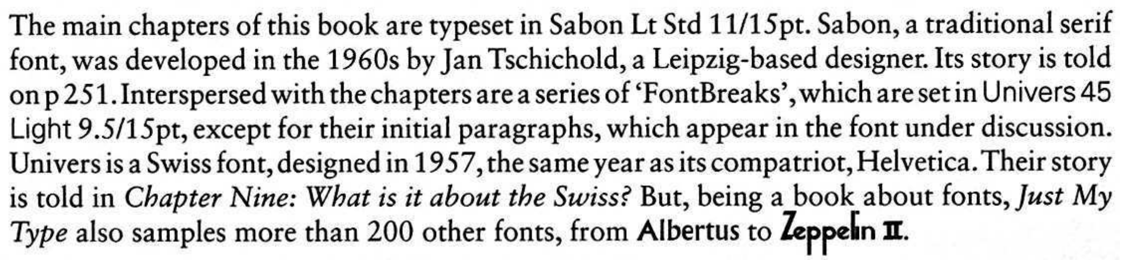
For me, with my modest but non-trivial background with and about fonts, some of the book's stories recapitulated what I already knew (e.g., about Helvetica from watching Gary Hustwit's documentary movie titled *Helvetica*). Some of the chapters added to what I already knew (e.g., more about the history of ampersands). Some of the chapters were on topics entirely new to me (e.g., the Type Archive founded and directed by Sue Shaw in South London — I want to visit this place). I also was tickled to read for the first time about the logo of the Beatles band. In a chapter titled "The Serif of Liverpool", we learn about the origin of this logo text with its capital B followed by small caps with a long descender on the T:

BEATLES

As is typical in all of the book's chapters, this chapter then goes on to other topics related to the initial theme of the essay — in this case, other famous logos in the rock world.

The book contains scads of specimens of different typefaces and fonts. The excerpt in figure 1 from the book's copyright page explains a bit about the fonts used. In that text, the words "Univers 45 Light", "Albertus", and "Zeppelin II" are set in the mentioned font. (The rest of the paragraph is in Sabon.) The mentioned samples of "more than 200 other fonts" throughout the book are mostly only a word or two long — the name of the mentioned font, as here.

This display of type specimens throughout the book is impressive as a compositing feat (imagine having to make so many fonts available to one's typesetting system for a single project). However, it lacks organization (and comparable samples of full alphabets) making it not so useful in terms of serving as a specimen book. The book does have an index (not always available in books written for lay readers), and this allows one to go from a font name to one or more book pages which may include a sample of the font.



The main chapters of this book are typeset in Sabon Lt Std 11/15pt. Sabon, a traditional serif font, was developed in the 1960s by Jan Tschichold, a Leipzig-based designer. Its story is told on p 251. Interspersed with the chapters are a series of 'FontBreaks', which are set in Univers 45 Light 9.5/15pt, except for their initial paragraphs, which appear in the font under discussion. Univers is a Swiss font, designed in 1957, the same year as its compatriot, Helvetica. Their story is told in *Chapter Nine: What is it about the Swiss?* But, being a book about fonts, *Just My Type* also samples more than 200 other fonts, from **Albertus** to **Zeppelin II**.

Figure 1: Fragment from copyright page of the book (magnified).

The book also includes a three page bibliography of relevant books and a two page list of Internet resources. It would have been nice for a reader interested in digging deeper into the content of any of the essays if the book cited the relevant bibliographic items via footnotes or end notes from the main text. However, this is one more book where the designer (or publisher's marketing department) apparently couldn't allow such research aids lest they damage the book's popularity. (I would think that a bibliography could include page references back to the main text that would to some extent provide the useful links without cluttering the main text with those unpopular superscript reference numbers.)

Two aspects of the book appear to be included to add graphical pizazz that I think fail to achieve a purpose in terms of providing usable information. There is a periodic-table layout of fonts inside the front and back covers of the book, taken from `squidspot.com/Periodic_Table_of_Typefaces.html`. However, the author (or book designer) didn't include enough information to make this chart useful to me. Then there is a foreword by Chip Kidd that is apparently supposed to show in a flashy way (2 pages of text, 14 pages of graphics, not much explanation) the use of a lot of different fonts. However, the foreword doesn't connect well with the rest of the book. Perhaps the foreword would have impressed me more if I had ever before heard of Chip Kidd (Wikipedia suggests he is a well-known graphic designer). Or it might have said more to me if I already had a well-developed eye for graphic design.

As mentioned above, the book would have appealed to me more if there was some logical thread that flowed through the book, although I can imagine that was hard for the author to figure out given the many disparate topics he touches upon. Overall the book is an impressive effort given (it seems) that the author is more of a journalist writing books on a variety of non-fiction topics (`simongarfield.com`) than a long-time expert in the world of fonts. I can imagine that going from an interest in the topic through the study to write an entire book on the topic would have been a pretty educational exercise, and the author has chosen a nice subset of what he learned as stories to include in the book.

Perhaps one thread of continuity could have been the author telling us more about his background with type and how he pursued the information in this book. How much did he already know, how did he learn more, and what mental map did he have in mind as he wrote the book? I know that the author is supposed to keep himself out of the story, but including some information about his journey in collecting the information for this book might provide an interesting side story that would draw the reader along through the book (and perhaps give the reader some greater global mental picture of the world of type).

In any case, the book is what it is. I will put it on my book shelf as an informal adjunct to the few other (more formal) books on type I own and the others I will continue to get from the library from time to time. (In fact, there is little overlap between *Just My Type* and the two books I already have on my shelf, Bringhurst's *The Elements of Typographic Style* and Felici's *The Complete Manual of Typography*.) No doubt I will dip into *Just My Type* again occasionally to recall an interesting anecdote or use it as a potential one-shot resource (using the book's index) when looking up something about fonts before consulting more specialized books.

My recommendation for others in the TEX world who may have about the same level of typeface expertise/non-expertise that I have is to borrow the book from your public library first to see how you feel about it. Then buy it for your own library of books on fonts and their history and use if you think it will be a useful addition, or buy it to give as a gift to friends and relatives who are naive on the subject of typefaces. It does contain a lot of fascinating information for the right reader.

◇ David Walden
`walden-family.com/texland`

Abstracts

***Les Cahiers GUTenberg* issue 56 (2011)**

Les Cahiers GUTenberg is the journal of GUTenberg, the French-language TeX user group (`www.gutenberg.eu.org`).

This issue is the proceedings of the fourth ConTeXt meeting and third TeXperience meeting, jointly published with NTG (as *MAPS* no. 43) and CSTUG (as *Zpravodaj* no. 2–4/2011).

Ján Kula and Pavel Stříž, Preface; p. 69

Arthur Reutenauer, Mobile TeX: Porter TeX pour l'iPad [Mobile TeX: Porting TeX to the iPad]; pp. 84–90

The article presents the achievement of Richard Koch, amongst other things the author of TeXShop and MacTeX developer, who has successfully compiled and used TeX on Apple's iPad.

Luigi Scarso, Jouer avec Flash depuis ConTeXt MkIV [Playing with Flash in ConTeXt MkIV]; pp. 91–101

The article presents one of the approaches to embedding Flash animations in a PDF file using ConTeXt MkIV and the Lua language running in the background.

Luigi Scarso, MicroTalk — `pdfsplit`; pp. 102–115

MicroTalk is a short and technical paper that shows some unusual, hopefully useful, ideas under the rubric "figure to code". The main topic is always typographic programming in ConTeXt and Lua. A bit of Lua code, the `\clip` macro and Leptonica extensions are the ingredients for this recipe to cook a `\pdfsplit` macro that tries to split a PDF into parts as the `\vsplit` does with `\vbox`es.

Ulrik Vieth, Expériences de typographie OpenType math avec LuaLaTeX et XeLaTeX [Experiences typesetting OpenType math with LuaLaTeX and XeLaTeX]; pp. 116–126

Compares OpenType math typesetting in two common TeX engines these days, LuaLaTeX and XeLaTeX. The differences in the outputs are presented as red-blue layering of the PDF files.

Taco Hoekwater and Hartmut Henkel, LuaTeX 0.60; pp. 127–133

[Published in *TUGboat* 31:2.]

Taco Hoekwater, LuaTeX 0.63 : référence [LuaTeX 0.63 short reference]; pp. 134–139

John Haltiwanger, Subtext: Une proposition de grammaire procédurale pour préformater les documents multisupports [Subtext: A proposed procedural grammar for a multi-output pre-format]; pp. 140–146

The article brings some thoughts and notes on typesetting for multi-output pre-format from the single source code.

Willi Egger, Redistribuer les pages [Arranging pages]; pp. 147–156

There is still much to be considered until we can hold a finished book in our hands, after the content is ready. In this article an overview on possible page arrangement schemes is presented. Although ConTeXt already has a considerable range of possibilities built-in, more arrangement schemes will be added in the near future, making ConTeXt even more versatile.

Libor Sarga, Guide TeX it: difficiles débuts de la composition pour des non-compositeurs [Guide TeX it: Uneasy beginnings of typesetters from the perspective of non-typesetters]; pp. 157–165

The article describes the process of typesetting a proceedings in TeX from the perspective of prospective typesetters along with challenges and obstacles encountered and solved during the work. Focused on the problems of generating a desired table of contents and captions of graphic objects, it further lists minor annoyances and tricks used to solve them. Also described is a field-proven electronic content management and synchronization system for different file versions utilized while working on the project in a decentralized fashion.

Jan Přichystal, Composition des tables et listes, et autres nouvelles fonctions de TeXonWeb [Typesetting of tables and lists and other new features in TeXonWeb]; pp. 166–169

This article describes new features in TeXonWeb. TeXonWeb is a web application which allows using the (LA)TeX typesetting system without needing its installation on a local computer. One of the most important characteristics of this application is to help beginners to start working with (LA)TeX. It offers them many tools, such as table and list wizards, to ease their first steps.

Timothy Eyre, ConTeXt pour les zines [ConTeXt for zines]; pp. 170–180

The article describes the design of the *New Escapologist* magazine, our motivations for using ConTeXt, some of the typographical features of the magazine and my experiences with using the ConTeXt Mark II macro package.

Hans Hagen, La technologie hybride de ConTEXt MkIV [ConTEXt MkIV hybrid technology]; pp. 182–300

The paper presents development, new features and tools of LuaTEX and ConTEXt MkIV.

Tomáš Hála, Les épreuves dans la pratique éditoriale et leur implémentation dans un système TEX [Marking proof-sheets in publishing practice and its implementation in the TEX system]; pp. 301–308

This paper deals with ways of marking proof-sheets in publishing practice. Four possible solutions are shown and discussed. Three of them are based on existing macros (page style `\headings`), or packages (`fancyhdr.sty`, `zwpagelayout.sty`); the fourth is original and specific, and contains a new style for LATEX — `thproof.sty`.

Karel Píška, Les fontes avec des tables OpenType complexes [Fonts with complex OpenType tables]; pp. 309–332

The paper presents development of complex OpenType fonts. Sample fonts cover Czech and Georgian handwriting with pervasive letter connections.

To begin, general principles of "advanced typography" are shown — complex metric data represented by OpenType tables (GSUB and GPOS) — and compared with the ligature and kerning tables in Metafont.

Then the history of OpenType font production is described — approaches, tools and techniques. Crucial problems, critical barriers, attempts and ways to reach successful solutions, are discussed and several tools for font creation, testing, debugging and conversions between various text and binary formats are demonstrated. Among these tools are, for example, AFDKO, VOLT, FontForge, TTX, and Font-TTF. Their features, advantages, disadvantages, and also cases of possible incompatibilities (or maybe errors) are illustrated. Finally, use of OpenType fonts in TEX world applications is presented: XƎTEX and LuaTeX (ConTEXt MkIV), the programs supporting reading and processing OpenType fonts directly.

[Received from Thierry Bouche.]

ArsTEXnica #14 (October 2012)

ArsTEXnica is the journal of GuIT, the Italian TEX user group (`http://www.guit.sssup.it/`).

Gianluca Pignalberi, Editoriale [From the editor]; p. 5

Jean-Michel Hufflen, Beyond BibTeX; pp. 7–14

This article is related to the production of bibliographies for documents typeset by means of LATEX or ConTEXt. We explore some recent ways to go beyond what is allowed by BibTEX. Then we explain why the programs `mlbiblatex` and `mlbibcontext` seem to us to be promising.

Ivan Valbusa, La forma del testo umanistico: la classe `suftesi` [Classical text shape: The `suftesi` class]; pp. 15–30

This article analyzes the main aspects that arise when typesetting a humanistic text, from the choice of font and layout to the style of titles and text elements (even making use of philosophical reflections). It also introduces the tools for typesetting provided by the `suftesi` class.

Claudio Beccari and Heinrich Fleck, Una classe per comporre un dizionario [A class for composition of a dictionary]; pp. 31–40

In this article we describe the history, the development, and the most relevant characteristics of classes `dictionarySCR` and `xdictionarySCR`, that allow to typeset, by pdfLATEX and XƎLATEX respectively, thematic dictionaries in a variety of disciplines.

Enrico Gregorio, LATEX3: un nuovo gioco per i maghi e per diventarlo [LATEX3: A new game for becoming a wizard]; pp. 41–47

We shall examine, by means of examples, some of the functions provided by the LATEX3 kernel. In particular, we shall focus on the usage of the sequence variable type and show something about the `l3keys` module.

Claudio Beccari, Un pacchetto per il controllo del codice fiscale italiano [A package to check the Italian personal fiscal code number]; pp. 48–52

The Italian Fiscal Code is a special string that encodes personal identity data; it may be issued only by the State Administrative Offices, but when a document is being written a piece of software should be available to check the correctness of the encoded string, in particular it must control the final check letter for coherence with the remaining data.

Roberto Giacomelli, Grafica ad oggetti con LuaTEX [Object-oriented graphics with LuaTEX]; pp. 53–71

In this paper we will make an attempt to measure the TEX user advantages and efficacy of an object oriented language, the programming paradigm recently added to the TEX world by the new typesetting engine LuaTEX. A case study was developed writing a new graphic object library in Lua, based on the PGF package, carried in a source TEX file with a small and simple package.

Luigi Scarso, MFLua: Intrumentation of METAFONT with Lua; pp. 72–81

[Published in *TUGboat* 32:2.]

Agostino De Marco, Gestione 'quasi automatica' dei valori numerici in esempi di calcolo ed esercizi svolti [Managing numbers in calculation examples and exercises]; pp. 82–94

In this paper I will show some techniques to manage numbers for facilitating the preparation of calculation examples or worked-out exercises. This is a well-known problem for authors of scientific textbooks. When they want to include a gallery of examples with calculations in their book it is of utmost importance to adopt a strategy designed to limit both typographic and conceptual errors. The techniques shown here handle numeric values which are produced outside the typesetting process with external software tools. Finally, similar techniques are mentioned that use LuaLATEX or the package `l3fp`.

Grazia Messineo and Salvatore Vassallo, Il pacchetto esami per la creazione di prove scritte [The package `esami` to generate written exams]; pp. 95–103

We present the package `esami` which extends some useful properties of the `exerquiz` and `probsoln` LATEX packages to produce databases of exercises in a collaborative way, and to produce exams.

Grazia Messineo and Salvatore Vassallo, Test online di matematica: il Progetto M.In.E.R.Va [Online math test: The M.In.E.R.Va project]; pp. 104–112

We present the packages `esami-online`, to create online exams with automatic corrections, and `minerva`, to create sets of exercises for high school students.

Matteo Fadini, Stemma codicum con LATEX [Stemma codicum with LATEX]; pp. 113–122

During preparation of a critical edition, it is necessary to draw the stemma codicum, i.e. the graphical representation of the lineage relationships of the textual transmission through the linkage of manuscripts and prints. There is no specific package designed to draw the stemmata. Nevertheless it is possible to adapt the packages usually employed for generative syntax trees in linguistic studies. This paper discusses two cases: `syntree`, a user-friendly package, and `xytree`, a more complex yet more powerful package. The analysis limits the description to the useful functions for drawing stemmata codicum.

Jerónimo Leal and Gianluca Pignalberi, Composizione di uno stemma codicum con TikZ [Stemma codicum typesetting with TikZ]; pp. 123–131

While LATEX has very good packages to typeset a more or less complex stemma codicum, TikZ allows one to achieve results at least as good, without requiring harder notational complexity.

Claudio Vincoletto, Alcuni stralci sul Computer Bodoni [Bits and pieces about Computer Bodoni]; pp. 131–137

The Computer Modern font source may be conveniently modified in order to obtain a new version of a Bodoni type nearer to the original one.

[Received from Gianluca Pignalberi.]

The PracTEX Journal 2012-1

The PracTEX Journal is an online publication of the TEX Users Group. Its web site is `http://tug.org/pracjourn`. All articles are available there.

Lance Carnes, Editorial — LATEX in the IT world

Editor's introduction to the issue.

Editors, Feedback from readers

Brian D. Beitzel, Formatting Sweave and LATEX documents in APA style

[Also published in *TUGboat* 33:1.]

Dimitrios Ververidis, Daniel Schneider and Joachim Köhler, The Vocal Tract LATEX package

`VocalTract.sty` is a package to visualize the vocal tract. A vocal tract is manipulated by a vector of articulation parameters according to S. Maeda's model. Animation can be achieved by providing a sequence of vectors over time, e.g. from Matlab. An embedded sequence of vectors in `VocalTract.sty` for certain German phonemes allows for a sequence of phonemes animation when no vector is available.

Han Lin Shang, Writing posters with `beamerposter` package in LaTeX

The `beamerposter` package in LaTeX is an excellent tool for the creation of posters. There are several options available using the `beamerposter` package when writing a poster in LaTeX. Here, I would like to present some of these options associated with the `beamerposter` package. I shall introduce the basics and some useful companion packages that make a poster look neat and nice.

Jim Hefferon, Seeing stars

MetaPost is a great tool for line art. We walk through creating a rate one-to-five graphic with it, highlighting some of its advantages over a mouse-driven graphics tool.

Luca Merciadri, TeX in the eBook era

There are many advantages to reading eBooks, and their usage is ever-increasing. There are those who prefer traditional printed books, but there is also a growing audience whose lifestyle and taste is suited to eBooks. We discuss some advantages of using eBooks, and creating them with LaTeX. We continue by explaining technical aspects that might improve and help you with your LaTeX eBook. We end with a short discussion about the PDF file format and its use with eBook documents.

Hongbin Ma, Easy-to-use Chinese MTeX Suite

Although there are many free and commercial TeX collections and distributions, Chinese TeXers still have many difficulties in TeXing Chinese documents. Some of the problems Chinese TeX users confront with TeX distributions are: the huge size, complex structure, nontrivial installation and configuration, lack of a full-featured Chinese editor, and lack of other needed add-ons. These difficulties may prevent many people from becoming TeXers or TeX experts. Motivated by these issues and many other practical demands, the author and several friends developed an easy-to-use and easy-to-learn Chinese MTeX Suite, which is a green, compact, free, convenient, pragmatic and powerful TeX distribution with many add-ons and unique features which are seldom available in other TeX distributions. The developers of MTeX have made continuous efforts to make this software more powerful and suitable for TeXers, programmers, teachers, and scientists. This article gives a brief introduction to the MTeX suite, including its motivation, main features, installation, usage instructions, kernel, default editor (SciTE LaTeX IDE), and other add-ons.

Yossi Gil, Bashful writing and active documents

Computerized typesetting still relies on metaphors drawn from the letterpress printing domain and is still concerned largely with the production of documents printed on paper. Active documents is an emerging technology by which the product of computerized typesetting is more than an aesthetically pleasing composition of letters, words and punctuation characters broken into lines and pages. An active document offers modes of interaction with its reader, while the document itself may change its content in response to events taking place in the external world. Bashful documents, the concept proposed by the `bashful` package, and discussed in this article, extend the user interaction offered by active documents to the time of the document creation. For example, the author of a textbook on computer programming may use `bashful` to automatically include in the text a transcript of a demonstration program, that is a precise replica of the program's execution at the time the document was authored. When writing a report on an important experiment, a scientist may employ `bashful` to automatically execute the experiment, whenever the report's text is run through LaTeX, and even include the experiment's results in the output document.

Rayans Carvalho and Francisco Reinaldo, Documenting ITIL processes with LaTeX

Many companies have evolved with the implementation of the Information Technology Infrastructure Library (ITIL), using the best practices and processes to achieve practical results. Good practice suggests what to do, but at the same time raises doubts about how to do it and which tools to use to get better work performance in ITIL. Noting these facts, this article presents a LaTeX-based processes and services documentation tool, as suggested by ITIL.

Lars Madsen, Avoid eqnarray!

[Published in *TUGboat* 33:1.]

Editors, Ask Nelly

Customizing lists?; Producing logos?

Editors, Distractions: guitar chords; font quizzes

Editors, Book review: *LaTeX and Friends* by Marc R.C. van Dongen

Letters

PUB and pre-TEX history

Don Knuth

I recently learned of an excellent web resource that strongly deserves to be better known: Some years ago, Larry Tesler put up

`http://www.nomodes.com/pub_manual.html`

which tells the story of PUB — which was my own introduction to computer typesetting. I did the first several errata collections for TAOCP with PUB, some time before I had learned that good typography was *also* possible, if pixels got small enough.

On that page he shows the original PUB manual on the left and gives extended annotations on the right ... at least at the beginning. He tells me that he would be glad to annotate further if encouraged by readers to do so.

Please tell *TUGboat* readers of this work at your earliest opportunity!

Cordially,

◇ Don Knuth
`http://www-cs-faculty.stanford.edu/~knuth`

Calendar

2012

Nov 9 — “Letterpress: something to say”, St. Bride Library, London, England. `stbride.org/events`

Nov 9 – 10 — 13. IBG-Jahrestagung in München, “Das E-Book. Herausforderung und Chance für die Buch- und Verlagswelt”, Internationalen Buchwissenschaftliche Gesellschaft, München, Germany. `www.buchwiss.de`

Nov 10 – May 19 — Exhibit: “Geletterd & geleerd [Literate & Learned]. Brill: 330 Years of Typography in the Service of Scholarship”, Museum Boerhaave, the Dutch National Museum for the History of Science and Medicine, Leiden, Netherlands. `www.museumboerhaave.nl/english/exhibitions/literate-learned`

Nov 13 — “Crafty types”, St. Bride Library, London, England. `stbride.org/events`

Dec 7 — “A Short History of Type”, lecture by Frank Romano, Museum of Printing, North Andover, Massachusetts. `www.museumofprinting.org`

Dec 14 — “A Short History of Printing”, lecture by Frank Romano, Museum of Printing, North Andover, Massachusetts. `www.museumofprinting.org`

2013

Jan 15 — Conference,“The Design of Understanding”, St Bride Library, London, England. `stbride.org/events`

Mar 7 – 9 — Typography Day 2013, Indian Institute of Technology. Guwahati, Assam, India. `www.typoday.in`

Mar 6 – 8 — DANTE Frühjahrstagung and 48th meeting, Justus-Liebig-Universität, Gießen, Germany. `www.dante.de/events/DANTE2013.html`

Mar 11 — *TUGboat* **34**:1, submission deadline (regular issue)

May 1 — **TUG election:** nominations due. `tug.org/election`

Jun 5 – 12 — The 5th International Conference on Typography and Visual Communication (ICTVC), “Against lethe ... ”, University of Nicosia, Cyprus. `www.ictvc.org`

Jun 10 – Aug 2 — Rare Book School, University of Virginia, Charlottesville, Virginia. Many one-week courses on type, bookmaking, printing, and related topics. `www.rarebookschool.org/schedule`

Jun 27 – 29 — Ladies of Letterpress Conference, Mt. Pleasant, Iowa. `www.letterpressconference.com`

Jul 8 — *TUGboat* **34**:2, submission deadline (regular issue)

Jul 16 – 19 — Digital Humanities 2013, Alliance of Digital Humanities Organizations, University of Nebraska–Lincoln. `dh2013.unl.edu`

Jul 18 – 21 — SHARP 2013, “Geographies of the Book”, Society for the History of Authorship, Reading & Publishing, University of Pennsylvania, Philadelphia. `www.library.upenn.edu/exhibits/lectures/SHARP2013`

Jul 21 – 25 — SIGGRAPH 2013, “Left Brain + Right Brain”, Anaheim, California. `s2013.siggraph.org`

Sep 10 – 13 — ACM Symposium on Document Engineering, Florence, Italy. `www.doceng2013.org`

Sep 26 – 27 — The Eleventh International Conference on the Book, Universität Regensburg Universitätsbibliothek, Regensburg, Germany `booksandpublishing.com/the-conference`

TUG 2013
Tokyo, Japan.

Oct 23 – 26 — The 34th annual meeting of the TeX Users Group. Presentations covering the TeX world. `tug.org/tug2013`

Status as of 1 November 2012

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568. e-mail: `office@tug.org`). For events sponsored by other organizations, please use the contact address provided.

A combined calendar for all user groups is online at `texcalendar.dante.de`.

Other calendars of typographic interest are linked from `tug.org/calendar.html`.

TEX Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at `http://tug.org/consultants.html`. If you'd like to be listed, please see that web page.

Aicart Martinez, Mercè
Tarragona 102 4^o 2^a
08015 Barcelona, Spain
+34 932267827
Email: `m.aicart (at) ono.com`
Web: `http://www.edilatex.com`
We provide, at reasonable low cost, LATEX or TEX page layout and typesetting services to authors or publishers world-wide. We have been in business since the beginning of 1990. For more information visit our web site.

Dangerous Curve
PO Box 532281
Los Angeles, CA 90053
+1 213-617-8483
Email: `typesetting (at) dangerouscurve.org`
Web: `http://dangerouscurve.org/tex.html`
We are your macro specialists for TEX or LATEX fine typography specs beyond those of the average LATEX macro package. If you use XƎTEX, we are your microtypography specialists. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical TEX and LATEX typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a TEX book.

Latchman, David
4113 Planz Road Apt. C
Bakersfield, CA 93309-5935
+1 518-951-8786
Email: `texnical.designs (at) gmail.com`
Web: `http://www.elance.com/s/dlatchman`
Proficient and experienced LATEX typesetter for books, monographs, journals and papers allowing your documents and books to look their possible best especially with regards to technical documents. Graphics/data rendered either using Ti*k*Z or Gnuplot. Portfolio available on request.

Moody, Trent
1981 Montecito Ave.
Mountain View, CA 94043
+1 650-283-7042
Email: `trent.moody (at) ymail.com`
Construction of technical documents with mathematical content from hand written (or partially formatted) sources. Delivered documents will be `.tex` and `.pdf` files produced with TEX or/and LATEX. Delivered documents can be publication ready manuscripts, macro libraries for modular document development, or mathematical libraries for document reuse.

I am an independent contractor with a PhD in mathematical physics from the University of California, Santa Cruz.

Peter, Steve
295 N Bridge St.
Somerville, NJ 08876
+1 732 306-6309
Email: `speter (at) mac.com`
Specializing in foreign language, multilingual, linguistic, and technical typesetting using most flavors of TEX, I have typeset books for Pragmatic Programmers, Oxford University Press, Routledge, and Kluwer, among others, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. In addition, I've helped publishers write, maintain, and streamline TEX-based publishing systems. I have an MA in Linguistics from Harvard University and live in the New York metro area.

Shanmugam, R.
No. 38/1 (New No. 65), Veerapandian Nagar, Ist St.
Choolaimedu, Chennai-600094, Tamilnadu, India
+91 9841061058
Email: `rshanmugam92 (at) yahoo.com`
As a Consultant, I provide consultation, training, and full service support to individuals, authors, typesetters, publishers, organizations, institutions, etc. I support leading BPO/KPO/ITES/Publishing companies in implementing latest technologies with high level automation in the field of Typesetting/Prepress, ePublishing, XML2PAGE, WEBTechnology, DataConversion, Digitization, Cross-media publishing, etc., with highly competitive prices.
I provide consultation in building business models & technology to develop your customer base and community, streamlining processes in getting ROI on our workflow, New business opportunities through improved workflow, Developing eMarketing/E-Business Strategy, etc. I have been in the field BPO/KPO/ITES, Typesetting, and ePublishing for 16 years, handled

Shanmugan, R. (cont'd)
various projects. I am a software consultant with Master's Degree. I have sound knowledge in TeX, LaTeX2ε, XMLTeX, Quark, InDesign, XML, MathML, DTD, XSLT, XSL-FO, Schema, ebooks, OeB, etc.

Sharma, Ganesh Kumar
A - 251 / 1, Opposite Primary School,
Shastri Nagar, Delhi 110052, India
+91 9810748682, 9013121611
Email: `ganeshsharma (at) yahoo.com`

I am a Master of Computer Applications (MCA) degree holder. I am well versed with MetaPost, HTML, MathML, Java, CSS, PHP, Unix shell scripting, C++, Ti*k*Z, Gnuplot and PostScript etc.

As a consultant and service provider, I am handling LaTeX and XeLaTeX composition to technical illustration, editorial services for: project management of conference proceedings; class/style files creation for LaTeX publications; a full management service for journals including correspondence with authors and issue make-up, including manuscript Preparation (pagination / composition, copy editing and proof reading), scanning and graphics designing, origination from handwritten manuscript or use of author-supplied code (TeX or word processor), and author support; the supply of HTML, PDF files (including hyperlinks and bookmarks) and other coding for electronic publication. I can typeset the books in Sanskrit and Hindi languages using LaTeX very well.

Currently, I am giving editorial services to many universities, reputed publishers and multinational companies, research groups etc.

Sievers, Martin
Im Treff 8, 54296 Trier, Germany
+49 651 4936567-0
Email: `info (at) schoenerpublizieren.com`
Web: `http://www.schoenerpublizieren.com`

As a mathematician with ten years of typesetting experience I offer TeX and LaTeX services and consulting for the whole academic sector (individuals, universities, publishers) and everybody looking for a high-quality output of his documents.

From setting up entire book projects to last-minute help, from creating individual templates, packages and citation styles (BibTeX, biblatex) to typesetting your math, tables or graphics — just contact me with information on your project.

Sofka, Michael
8 Providence St.
Albany, NY 12203
+1 518 331-3457
Email: `michael.sofka (at) gmail.com`

Skilled, personalized TeX and LaTeX consulting and programming services.

I offer over 25 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in TeX and LaTeX: Automated document conversion; Programming in Perl, C, C++ and other languages; Writing and customizing macro packages in TeX or LaTeX; Generating custom output in PDF, HTML and XML; Data format conversion; Databases.

If you have a specialized TeX or LaTeX need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss your project.

Veytsman, Boris
46871 Antioch Pl.
Sterling, VA 20164
+1 703 915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`

TeX and LaTeX consulting, training and seminars. Integration with databases, automated document preparation, custom LaTeX packages, conversions and much more. I have about seventeen years of experience in TeX and thirty years of experience in teaching & training. I have authored several packages on CTAN, published papers in TeX related journals, and conducted several workshops on TeX and related subjects.

Young, Lee A.
127 Kingfisher Lane
Mills River, NC 28759
+1 828 435-0525
Email: `leeayoung (at) morrisbb.net`
Web: `http://www.latexcopyeditor.net`
`http://www.editingscience.net`

Copyediting your `.tex` manuscript for readability and mathematical style by a Harvard Ph.D. Your `.tex` file won't compile? Send it to me for repair. Experience: edited hundreds of ESL journal articles, economics and physics textbooks, scholarly monographs, LaTeX manuscripts for the Physical Review; career as professional, published physicist.

2013 TEX Users Group election

Barbara Beeton
for the Elections Committee

The positions of TUG President and six members of the Board of Directors will be open as of the 2013 Annual Meeting, which will be held in October 2013 in Japan.

The directors whose terms will expire in 2013: Kaja Christiansen, Jonathan Fine, Steve Grathwohl, Jim Hefferon, Klaus Höppner, and David Walden.

Continuing directors, with terms ending in 2015: Barbara Beeton, Karl Berry, Susan DeMeritt, Michael Doob, Taco Hoekwater, Ross Moore, Cheryl Ponchin, Philip Taylor, and Boris Veytsman.

The election to choose the new President and Board members will be held in Spring of 2013. Nominations for these openings are now invited.

The Bylaws provide that "Any member may be nominated for election to the office of TUG President/ to the Board by submitting a nomination petition in accordance with the TUG Election Procedures. Election ... shall be by written mail ballot of the entire membership, carried out in accordance with those same Procedures." The term of President is two years.

The name of any member may be placed in nomination for election to one of the open offices by submission of a petition, signed by two other members in good standing, to the TUG office at least two weeks (14 days) prior to the mailing of ballots. (A candidate's membership dues for 2013 will be expected to be paid by the nomination deadline.) The term of a member of the TUG Board is four years.

A nomination form follows this announcement; forms may also be obtained from the TUG office, or via `http://tug.org/election`.

Along with a nomination form, each candidate must supply a passport-size photograph, a short biography, and a statement of intent to be included with the ballot; the biography and statement of intent together may not exceed 400 words. The deadline for receipt of nomination forms and ballot information at the TUG office is **1 May 2013**. Forms may be submitted by FAX, or scanned and submitted by e-mail to `office@tug.org`.

Ballots will be mailed to all members within 30 days after the close of nominations. Marked ballots must be returned no more than six (6) weeks following the mailing; the exact dates will be noted on the ballots.

Ballots will be counted by a disinterested party not affiliated with the TUG organization. The results of the election should be available by early June, and will be announced in a future issue of *TUGboat* as well as through various TEX-related electronic lists.

2013 TUG Election — Nomination Form

Only TUG members whose dues have been paid for 2013 will be eligible to participate in the election. The signatures of two (2) members in good standing at the time they sign the nomination form are required in addition to that of the nominee. **Type or print** names clearly, using the name by which you are known to TUG. Names that cannot be identified from the TUG membership records will not be accepted as valid.

The undersigned TUG members propose the nomination of:

Name of Nominee: ____________________

Signature: ____________________

Date: ____________________

for the position of (check one):

□ **TUG President**

□ **Member of the TUG Board of Directors**

for a term beginning with the 2013 Annual Meeting, **October 2013**

1. ____________________
(please print)

____________________ ____________
(signature) (date)

2. ____________________
(please print)

____________________ ____________
(signature) (date)

Return this nomination form to the TUG office (forms submitted by FAX or scanned and submitted by e-mail will be accepted). Nomination forms and all required supplementary material (photograph, biography and personal statement for inclusion on the ballot) must be received in the TUG office no later than **1 May 2013**.[1] It is the responsibility of the candidate to ensure that this deadline is met. Under no circumstances will incomplete applications be accepted.

□ nomination form

□ photograph

□ biography/personal statement

TEX Users Group **FAX:** +1 815 301-3568
Nominations for 2013 Election
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

[1] Supplementary material may be sent separately from the form, and supporting signatures need not all appear on the same form.

TUGBOAT Volume 33 (2012), No. 3

Introductory

Intermediate

Intermediate Plus

Advanced

Contents of other TeX journals

Reports and notices