TUGboat

Volume 37, Number 2 / 2016
TUG 2016 (Toronto) Conference Proceedings

TEX Users Group

TUGboat (ISSN 0896-3207) is published by the TEX Users Group. Web: `http://tug.org/TUGboat`.

Individual membership

2016 dues for individual members are as follows:

- Regular members: $105.
- Special rate: $75.

The special rate is available to students, seniors, and citizens of countries with modest economies, as detailed on our web site. Also, anyone joining or renewing before March 31 receives a $20 discount.

- Regular members (early bird): $85.
- Special rate (early bird): $55.

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 detailed membership information, visit the TUG web site.

Subscriptions

TUGboat subscriptions (non-voting) are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate for 2016 is $110.

Institutional membership

Institutional membership is primarily a means of showing continuing interest in and support for TEX and the TEX Users Group. It also provides a discounted membership rate, site-wide electronic access, 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 list of trademarks which commonly appear in *TUGboat* should not be considered complete.

TEX is a trademark of American Mathematical Society.
METAFONT is a trademark of Addison-Wesley Inc.
PostScript is a trademark of Adobe Systems, Inc.

[printing date: September 2016]

Printed in U.S.A.

Board of Directors

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

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`

2016 Conference Proceedings

TeX Users Group
Thirty-seventh Annual Meeting
Toronto, Canada
July 25–27, 2016

TUGBOAT

COMMUNICATIONS OF THE TeX USERS GROUP

TUGboat Editor	Barbara Beeton
Proceedings Editor	Karl Berry

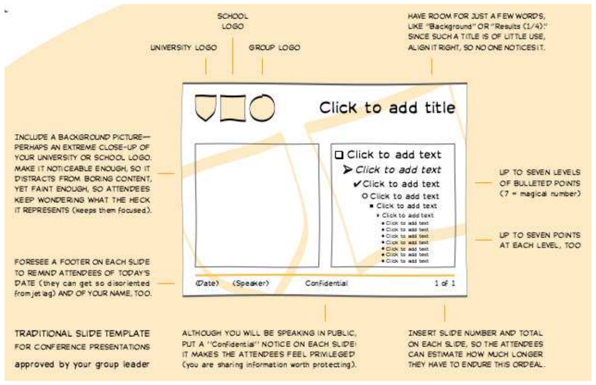
Volume 37, Number 2 • 2016
Portland • Oregon • U.S.A.

Passport to the TEX canvas

Pavneet Arora

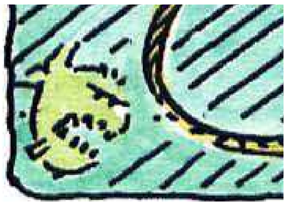
Welcome.

Whenever I begin a talk, in my head I hear Jean-luc Doumount's admonishment on what not to do when presenting. (From his article on presentations, "Traditions, templates, and group leaders".)


SCHOOL LOGO
UNIVERSITY LOGO
GROUP LOGO
HAVE ROOM FOR JUST A FEW WORDS, LIKE "Background" OR "Results (1/4)." SINCE SUCH A TITLE IS OF LITTLE USE, ALIGN IT RIGHT, SO NO ONE NOTICES IT.
Click to add title
INCLUDE A BACKGROUND PICTURE— PERHAPS AN EXTREME CLOSE-UP OF YOUR UNIVERSITY OR SCHOOL LOGO. MAKE IT NOTICEABLE ENOUGH, SO IT DISTRACTS FROM BORING CONTENT, YET FAINT ENOUGH, SO ATTENDEES KEEP WONDERING WHAT THE HECK IT REPRESENTS (keeps them focused).
Click to add text
Click to add text
Click to add text
Click to add text
Click to add text
Click to add text
UP TO SEVEN LEVELS OF BULLETED POINTS (7 = magical number)
UP TO SEVEN POINTS AT EACH LEVEL, TOO
FORESEE A FOOTER ON EACH SLIDE TO REMIND ATTENDEES OF TODAY'S DATE (they can get so disoriented from jet lag) AND OF YOUR NAME, TOO.
(Date) (Speaker) Confidential 1 of 1
TRADITIONAL SLIDE TEMPLATE FOR CONFERENCE PRESENTATIONS approved by your group leader
ALTHOUGH YOU WILL BE SPEAKING IN PUBLIC, PUT A "Confidential" NOTICE ON EACH SLIDE: IT MAKES THE ATTENDEES FEEL PRIVILEGED (you are sharing information worth protecting).
INSERT SLIDE NUMBER AND TOTAL ON EACH SLIDE, SO THE ATTENDEES CAN ESTIMATE HOW MUCH LONGER THEY HAVE TO ENDURE THIS ORDEAL.

And so I will resist the temptation to fall into the more obvious traps, although along the way I will no doubt transgress. Presumably we all know which city we are in. And we know the reason for our being here:

Even so, some of us may yet be unfamiliar with one critical aspect of this city: the pronunciation of its name. When Duane Bibby and I were discussing concepts for the conference logo, I put forth a small request: that the logo should have a small Easter Egg which would be a reminder of just this point. Any takers on what that Easter Egg might be?



That's right. Toronto is pronounced like piranha. So when you're at the next border, and the officer asks where you are heading to, simply keep this mental image in mind, and you will come off sounding like a true local.

The inspiration for the canvas and the painting style comes from The Group of Seven, an influential group of Canadian impressionist painters — AJ Casson and Lawren Harris being two of my favourites. Those on the Georgian Bay excursion this coming Friday will have a chance to see many Group of Seven works in person at the McMichael Art Collection.

1 Canvases

1.1 Collaboration

This year at Microsoft's Build conference, Satya Nadella used an interesting turn of phrase: he spoke of "conversational canvases". And let me just say that in my wildest dreams I never, ever, ever imagined that I would open any presentation of mine by quoting the CEO of Microsoft. I hope this is more of an indicator of what Microsoft is about these days than an indicator about me.

So let me say what a delight it is to have Kevin Larson from Microsoft's Advanced Typography Group join us. When I went about trying to contact the group, I landed up on a website that looked as if it was a screen scrape of a Windows 97 screen. The old Tektronix 4010s that I used to work on had a crisper image, and for that matter could be used as a crisper given the dosage coming off the panel!

It was a happy day indeed when I heard back from Microsoft, and then had an email exchange with Kevin. Kevin will be speaking this afternoon.

But back to canvases: what is this canvas that Satya was talking about? The buzz nowadays is about IoT (Internet of Things, or not only network enabled but aware devices), intelligent bots enabled by cloud based machine learning, augmented reality, and virtual reality.

When I thought about it some more, I began to suspect that what he was really referring to had more to do with a specific type of human activity, or mode of "interaction". That the canvas across these interactions was the common element. That is, in this "mobile first, cloud first" vision, the device has become the canvas.

When over-the-air TV ruled this continent's airwaves, if one were to switch from CBS News — with its exquisite branding with the very beautiful Didot typeface under the guidance of Lou Dorfsman — to ABC's Wide World of Sports did we refer to it as different viewing canvases? Or was it simply different content utilizing the same canvas, namely the television?

However you might consider the distinction between canvas and content, the question remains: "What sort of activities does this canvas enable and encourage?"

There are three prominent ones with conversational canvases: (1) collaboration, (2) cooperation, and sometimes (3) confrontation.

So these new world canvases would be Facebook, Twitter, Skype, WhatsApp, Slack, GroupMe, OwnCloud, email or any of the other bewildering range of choices available. They are all marked by a cacophony of voices, each clamouring to interject the next riposte and make a mark.

Here the canvas can also be seen as a viewport for the race to acquire the largest number of users as quickly as possible. We attribute virtue to the canvas not by its content but rather by its viability. And a declining or even steady user base is taken as a mark of irrelevance.

In the immortal words of perhaps the greatest of the modern worldly philosophers:

"If you ain't first, you're last!"
— Ricky Bobby, *Talladega Nights*

And since we are so dependent on — at least the first two activities — collaboration and cooperation, it is natural to put so much emphasis on these conversational canvases. Plus, who amongst us can resist the siren call of shiny new technology, and what it might represent of our needs and aspirations — our innate desire to be acknowledged for being at the vanguard of the future?

Indeed it is reflected in the very names given to the gatherings where these are showcased, or as company mottos: Build (Microsoft), Invent (HP), Maker Fair (Raspberry Pi).

It is an expression of our creative impulse into something tangible, whether it be a woodworking project, or a new tablet PC.

But if this is the main attraction of the big top with its "barkers and coloured balloons", to quote Neil Young, what is happening behind the tent?

In other words is this all there is?

1.2 Discovery

Behind this illusion of spectacle, behind the curtain that forms the backdrop of the glitz and glamour of the stage is another form of activity. And this activity sustains the show. It is the activity of Discovery.

This endeavour is lonelier; the path more difficult. The effort expended can span many years, even decades, with little in the form of assurance that a positive outcome awaits the supplicant. And even if you are mostly right, or even completely right, someone else might still beat you to the punch. Acknowledgement of great achievement in this activity is rarely resounding.

Working here can often feel like pushing on a string. Think of the connection that takes you from Menaechmus' discovery of the mathematics behind conic sections, to Kepler's work on planetary motion, to NASA's Apollo missions. This is the 47th anniversary of Apollo 11, and one of the most compelling things on Twitter is not the two political conventions! It is the real time broadcast of the mission communications for that Apollo mission.

Discovery is the driving force behind research, both pure and applied. It spans the fields of science, technology, engineering and mathematics, or STEM for short.

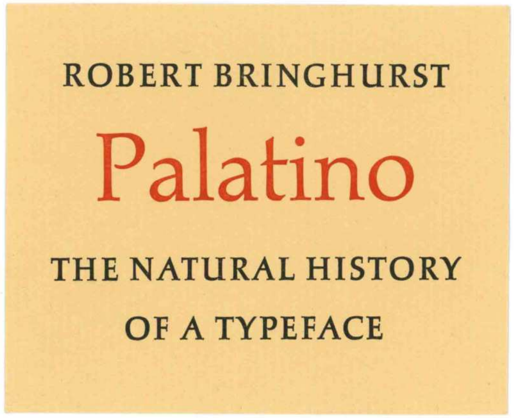
In this vast domain there too is a common canvas. Here, it is the *Page* that holds court. Even if we don't yet know for how long, the primacy of the *Page* is, for now, unquestioned in this dominion.

And the *Page* is the ideal canvas for using TeX. For TeX has an unequalled ability to express complex ideas using the specific notations and syntactic constraints used in all of these fields. The rarest of cuneiform marks, along with their most pedantic rules of layout can all be handled with aplomb using TeX. Even after all of these years, it is able to adapt to the natural language of the domain rather than force the language to conform to its limitations. (Cartoon originally from `somethingofthatilk.com`.)

LATEX
YOUR PAPER MAKES NO GODDAMN SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.

This canvas, in contrast to the conversational canvases — perhaps only because those are still evolving rapidly — has a natural order. We may still be in the process of abstracting overarching rules of representation on those electronic canvases — and here I am thinking of Twitter Bootstrap as a significant milestone along the way — but that is something for the future.

Thanks to another one of our special guests, Robert Bringhurst, we are able to talk about proportion and spacing to arrive at a much deeper understanding of what constitutes beauty on this canvas. Just think of the implications of this. We can speak of beauty with a common appreciation. What a marvellous thing to share! Robert Bringhurst will be speaking tomorrow.


ROBERT BRINGHURST

Palatino

THE NATURAL HISTORY OF A TYPEFACE

But is even Discovery all there is? The more appropriate question might be, whether the activity of Discovery qualifies as Aristotle's "Unmoved Mover", so to speak?

1.3 Contemplation

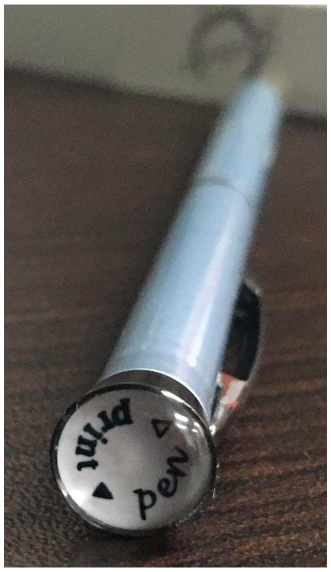
There is an even more isolating endeavour than *Discovery*. And it is *Contemplation*, which in turn feeds its nearest neighbours *Intuition*, *Imagination*, and *Interpretation*.

This is the most solitary of endeavours, and the practitioner must traverse a vastly more barren landscape as they climb the slope and try to pierce "The Cloud of Unknowing". The artist must confront their own self-doubt when faced with that blank page or surface.

> "The writer who loses his self-doubt, who gives way as he grows old to a sudden euphoria, to prolixity, should stop writing immediately: the time has come for him to lay aside his pen." — Colette

And what of this canvas? What tools might we use to express our ideas.

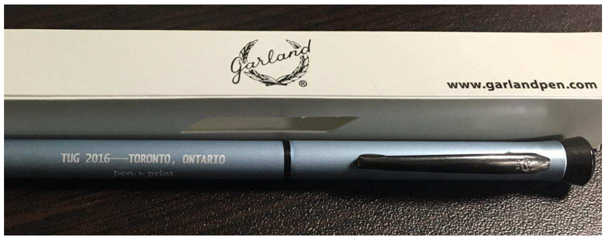
This is how we landed up on one of the themes of this conference, which you will find on the pens that are part of the registration kit for the conference.



This theme is "Pen to Print". It acknowledges the connection between artist and artisan that allows enduring works of art to be created. The source of an idea often has its earliest expression in ink.

We were privileged, yesterday, to have visited Tim and Elke Inkster's unique printing house, Porcupine's Quill. Craft and Quality should never be allowed to be taken for granted.

Coming back to the pen itself, these pens were made by the Garland Pen Company of Providence, RI, the same location as where TUG was founded. Some of you might notice that there are three fonts used on the inscription, and the em-dash is shown in "code" — that is, as a triple hyphen. In this audience, this is akin to having used Comic Sans, in all its garish glory, when putting out the UN Declaration of Human Rights.



But here too, a small puzzle to tickle the mind. The font family used on the pen is Lucida, and I only wish Chuck Bigelow and Kris Holmes were here so that we might have been able to call them out in person. No doubt we can query Chuck further on this when he arrives tomorrow. The first line, identifying the conference, is done in Lucida Grande Mono DK which is the special edition dedicated to

Don Knuth. It acknowledges the essential step that to get from the handwriting font used for the word *Pen*, to the book font used for the word *Print*, the content must first be coded. We used the province rather than country to draw extra attention to the subtle distinction between the shapes of letter 'O' and the digit zero, without resorting to explicit marking, one of B&H's design goals.

2 Passport

I have spoken at length of canvases, and over which of these canvases TEX's powerful expressiveness still has sway.

Now let me talk about the passport. These days it has become an ID document — something that identifies the holder as to who they are. A means to get through ever tighter security when entering an airport, boarding a plane or crossing over a border. It is nowadays treated as a gateway key.

But to me, a passport conjures up a much more romantic image of a more innocent time. It speaks of exotic lands, and people on faraway shores; of adventurous travels, and the chance to shed our skins of the familiar patterns of daily life. To stand in awe of unfamiliar vistas, and know with certainty that they will form indelible impressions upon the backdrop of our memories.

At its essence a passport allows us to delight in the stories of others, and by doing so connect our own to theirs. To my mind, that is all there really is. We are both made and then defined by our stories.

When I put, as the title to this talk, the "Passport to the TEX Canvas" I am referring to TUG. I think of TUG not so much as an organization, but as a collection of conversations told as stories. Of talks given formally at conferences such as this, but more importantly, discussions carried on in the margins.

Conversational canvases bristle with activity, exciting and sometimes enraging the participants even as they sit in silent, electronic isolation. In contrast, the TEX canvas — the *Page* — which demands of the reader quiet, concentrated effort, can allow the same reader to escape noisy and boisterous surroundings.

As it did with me late one night on Rue Sainte-Catherine in Montréal, overflowing with revellers, and where I found myself in an open-to-the-street cafe reading Northrop Frye's *Myth and Metaphor*. This canvas, without the transactional pull from advertisements interjecting themselves, allows us to soar alongside stories into our imaginations.

3 Stories

So let me tell you about a story. A story about this bag. It was given to me by Pavel Striz as a parting gift at the ConTEXt meeting in Brejlov, Czech Republic shortly after my first TUG conference, which would have been in San Francisco. Of all the people here, I think only Arthur and I attended. Have I missed anyone?

4th International ConTeXt Meeting Brejlov Czech Republic September 13 – 18 2010 theme: Documentation

On it is the inscription "Karlovy Vary International Film Festival". I had no idea of its importance when I received it.

A few months later I was taking some training in Atlanta, Georgia. I was sitting at a table, head down, working out some assigned problem. This bag was on the table. All of a sudden I heard a voice beside me: "Did you enjoy Karlovy Vary?" Actually, I have no idea what precisely was said, as it didn't really register at all, because it was so out of context.

I turned to find a giant of a man. I was easily dwarfed in his shadow. And his voice had a southern drawl. It took another few attempts by him for me to finally comprehend that he was talking about this bag. Well he went on to explain that he had grown up in rural Georgia, and ended up marrying a Czech woman. His father-in-law had a property in Karlovy Vary, and so this man along with his family spent a portion of their summers in the Czech Republic. He had learnt to speak fluent Czech.

And we got to talking about typesetting, TEX or in this case ConTEXt, and the arts scene in and around Prague. What an extraordinary conversation it led to!

Lest you think that all stories come this easily, let me tell you of another chance encounter on the same trip, and a missed opportunity.

I was at the home of a family member, and they took delivery of some furniture. On the truck was the company name Grissom.

Now I imagine that many of us were mesmerized by the Apollo programme. For me, the Apollo astronaut who somehow caught my imagination was Gus Grissom, who died tragically on the launchpad in Apollo 1. I think it was both because he had an engineering background, and also because there is hardly a photograph of him that doesn't show him laughing. (Photo courtesy of NASA.)

I mentioned this to one of the delivery persons, and that there was a high school in Alabama dedicated to him. This person didn't recognize the name, but wishing, I think, to continue on with the conversation mentioned that the moving company was his uncle's, Marquis Grissom. I didn't recognize that name, but I should have since Marquis Grissom played for the Montréal Expos, and is by all accounts a class act through and through.

So there we were, telling our stories past each other. If only I had stopped and asked him more about his uncle!

Stories are around us, and they can bind us together in kinship.

It is the hope of the conference organizers that TUG act as your passport, and the messenger bag that each of you have received act as a repository to many new, and wonderful stories. And that it may inspire some of you to put at least some of those stories down on the TEX canvas for others to enjoy for years to come.

Welcome to TUG 2016!

◇ Pavneet Arora
Waroc Informatik
`pavneet_arora (at) waroc dot com`
`waroc.com`

TUG 2016 — Toronto, Canada

Sponsors

TeX Users Group ▪ DANTE e.V.
The Porcupine's Quill
River Valley Technologies — UK
with special assistance from individual contributors. *Thanks to all!*

Special guests

Charles Bigelow, Bigelow & Holmes ▪ Robert Bringhurst, Quadra Island, BC ▪ Kevin Larson, Microsoft

Conference committee

Pavneet Arora ▪ Karl Berry ▪ Jim Hefferon ▪ Robin Laakso ▪ Steve Peter

Bursary committee

Taco Hoekwater, chair ▪ Jana Chlebikova ▪ Kaja Christiansen ▪ Bogusław Jackowski ▪ Alan Wetmore

Participants

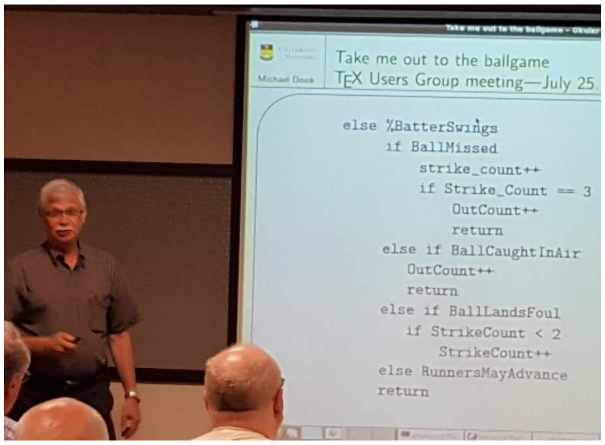
Pavneet Arora, Bolton, ON
Amartyo Banerjee, TNQ, India
Abdelouahad Bayar, Cadi Ayyad University, Morocco
Kaveh Bazargan, River Valley Technologies, UK
Nelson Beebe, University of Utah
Barbara Beeton, AMS, Providence, RI
Karl Berry, Bandon, OR
Johannes Braams, Zoetermeer, The Netherlands
Jozo Capkun, Caledon, ON
David Casperson, University of Northern British Columbia
Jaeyoung Choi, Seoul, Korea
Joe Clark, Toronto, ON
Jennifer Claudio, Oak Grove High School, CA
Paulo Ney de Souza, BooksInBytes
Sue DeMeritt, Center for Communications Research, La Jolla, CA
Mercedes Dollard, Pittsburgh, PA
Michael Doob, University of Manitoba
Behdad Esfahbod, Google
Yukitoshi Fujimura, Ichikawa-shi, Japan
Christian Gagné, Université Laval
Federico Garcia-De Castro, Alia Musica Pittsburgh
Peter Giunta, MIT
John Goyo, Acton, ON
Steve Grathwohl, Duke University Press
Katie Harding, Dartmouth College
Jim Hefferon, Saint Michael's College
Tim Inkster, The Porcupine's Quill
Steve Izma, Between The Lines Publishing
John Eddie Kerr, Wellington Law Association Library
Stefan Kottwitz, Lufthansa Industry Solutions
Robin Laakso, TeX Users Group
Richard Leigh, St Albans, UK
Lothar Meyer-Lerbs, Bremen, Germany
Frank Mittelbach, LaTeX3 Project
T Rishikesan Nair, River Valley Technologies, India
Kim Nesbitt, Canadian Journal of Economics
Steve Peter, TUG
John Plaice, Montreal, Canada
Cheryl Ponchin, Center for Communications Research, Princeton, NJ
Geoffrey Poore, Union University, Tennessee
Norbert Preining, Ishikawa, Japan
C V Rajagopal, River Valley Technologies, India
Arthur Reutenauer, London, UK
Volker RW *Schaa,* DANTE e.V.
Herbert Schulz, Naperville, IL
Heidi Sestrich, Carnegie Mellon University
Michael Sharpe, UC San Diego
A M Shanmugam Pillai, River Valley Technologies, India
Keiichiro Shikano, Tokyo, Japan
Matthew Skala, IT University of Copenhagen
Michael Sofka, Rensselaer Polytechnic Institute
Christina Thiele, Nepean, ON
David Tulett, Memorial University, Newfoundland
Boris Veytsman, George Mason University
David Walden, East Sandwich, MA

TUG 2016 program

Monday July 25	8:00 am	*registration*	
	8:45 am	Pavneet Arora, Bolton, ON	*Opening: Passport to the TeX canvas*
	9:30 am	Geoffrey Poore, Union Univ.	*Advances in PythonTeX*
	10:00 am	Stefan Kottwitz, Lufthansa Industry Sol.	*TeX in industry I — programming Cisco switches using TeX*
	10:30 am	*break*	
	10:45 am	Stefan Kottwitz	*TeX in industry II — designing converged network solutions*
	11:15 am	Boris Veytsman, George Mason Univ.	*Making ACM LaTeX styles*
	11:45 am	Frank Mittelbach, LaTeX3	*Alice goes floating — global optimized pagination including picture placements*
	12:45 pm	*lunch*	
	1:45 pm	Michael Doob, Univ. of Manitoba	*baseball rules summary*
	2:00 pm	Amartyo Banerjee, S.K. Venkatesan, TNQ	*A Telegram bot for printing LaTeX files*
	2:30 pm	Norbert Preining, Ishikawa, Japan	*Security improvements in the TeX Live Manager and installer*
	3:00 pm	Arthur Reutenauer, Royal Opera House	*The TeX Live M sub-project*
	3:45 pm	*break*	
	4:00 pm	Kevin Larson, Microsoft	*Reading between the lines: Improving comprehension for students*
	≈ 5 pm	*end*	
Tuesday July 26	8:25 am	*announcements*	
	8:30 pm	Kaveh Bazargan, River Valley Technologies, UK	*A graphical user interface for TikZ*
	9:00 am	Matthew Skala, IT Univ. of Copenhagen	*Astrological charts with* `horoscop` *and* `starfont`
	9:30 am	David Tulett, Memorial Univ.	*Development of an e-textbook using LaTeX and PStricks*
	10:00 am	Christian Gagné, Univ. Laval	*An Emacs-based writing workflow inspired by TeX and WEB, targeting the Web*
	10:30 am	*break*	
	10:45 am	Frank Mittelbach	*In memoriam: Sebastian Rahtz*
	11:00 am	Jim Hefferon, Saint Michael's College	*A LaTeX reference manual*
	11:15 am	Arthur Reutenauer, Mojca Miklavec	*Hyphenation past and future:* `hyph-utf8` *and* `patgen`
	11:45 am	Federico Garcia-De Castro, Alia Musica	*TeXcel?*
	12:15 pm	Jennifer Claudio, Oak Grove High School	*A brief reflection on TeX and end-user needs*
	12:45 pm	*lunch*	(with typeforming video, 40 min)
	2:00 pm	Jaeyoung Choi, Seoul, Korea	*MFCONFIG: Metafont plug-in for the Freetype rasterizer*
	2:30 pm	Michael Sharpe, UC San Diego	*New font offerings — Cochineal, Nimbus15 and LibertinusT1Math*
	3:00 pm	*break*	
	3:15 pm	Robert Bringhurst, Quadra Island, BC	*The evolution of the Palatino tribe*
	4:15 pm	*TUG Annual General Meeting*	
	≈ 5:15 pm	*end*	
	5:15 pm	Herbert Schulz, Naperville, IL	*Workshop: TeXShop tips & tricks*
Wednesday July 27	8:25 am	*announcements*	
	8:30 am	Jennifer Claudio	*The case for justified text*
	9:00 am	Boris Veytsman	*Are justification and hyphenation good or bad for the reader?*
	9:30 am	Charles Bigelow, Bigelow & Holmes	*Looking for legibility*
	10:30 am	*break*	
	10:45 am	David Walden, East Sandwich, MA	*Some notes on the history of digital typography*
	11:15 am	Tim Inkster, The Porcupine's Quill	*The beginning of my career*
	12:15 pm	*lunch*	(with road painting video, 10 min)
	1:45 pm	*group photo*	
	2:00 pm	Joe Clark, Toronto, ON	*Type and tiles on the TTC*
	3:00 pm	*break*	
	3:15 pm	Abdelouahad Bayar, Cadi Ayyad Univ.	*Towards an operational (La)TeX package supporting optical scaling of dynamic mathematical symbols*
	3:45 pm	John Plaice, Montreal, QC	*Zebrackets: A score of years and delimiters*
	4:15 pm	Charles Bigelow	*Probably approximately not quite correct: Revise, repeat*
	≈ 5:15 pm	*end*	
	6 pm	*Type and Tile Subway Tour, Joe Clark*	(typography discussion at 3–5 subway stops)

Outside The Porcupine's Quill, in Erin, Ontario with proprietor Tim Inkster (right) and Steve Izma (left), Between The Lines Publishing (using **groff**).

Invited speakers Kevin Larson, Robert Bringhurst and Charles Bigelow.

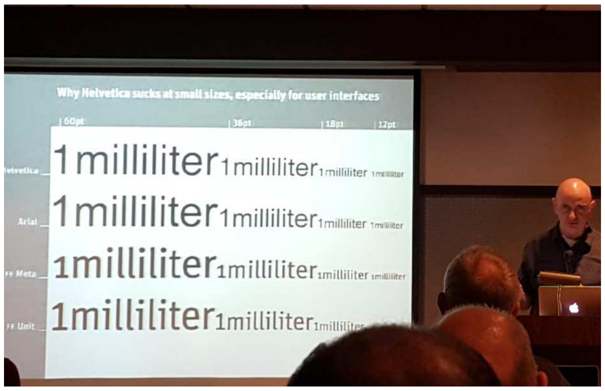


```
else %BatterSwings
     if BallMissed
          strike_count++
          if Strike_Count == 3
               OutCount++
               return
     else if BallCaughtInAir
          OutCount++
          return
     else if BallLandsFoul
          if StrikeCount < 2
               StrikeCount++
     else RunnersMayAdvance
     return
```

Michael Doob explains baseball to an international audience.

TUG at the Toronto Blue Jays vs. San Diego Padres game with Frank Mittelbach, Kevin Larson, Michael Doob, Stefan Kottwitz; Arthur Reutenauer, Johannes Braams, Yukitoshi Fujimura, Chris and Herb Schulz.

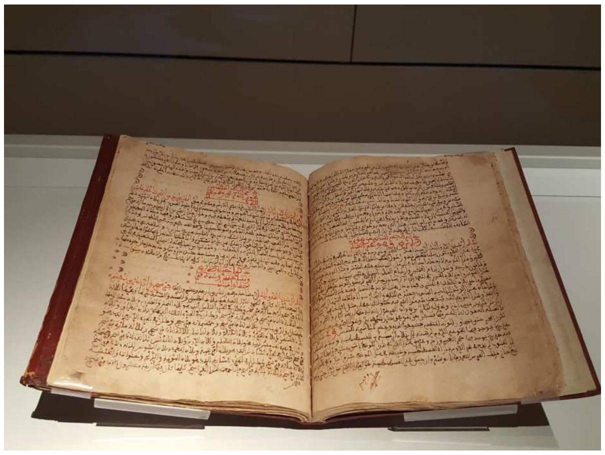
A sampling of design and font catalogs, Toronto Public Reference Library.


Why Helvetica sucks at small sizes, especially for user interfaces
60pt
36pt
18pt
12pt
Helvetica
Arial
FF Meta
FF Unit
1milliliter 1milliliter 1milliliter 1milliliter

Joe Clark demonstrating why Helvetica (first line) is not a good font for signage.

Rare Book Room exhibit in the Toronto Public Reference Library, guided by librarian Steven Shubert.

The Arthur Conan Doyle Room, in the Toronto Public Reference Library.


One Hundred and One Nights, Aga Khan Museum.

Cruise departure.

Banquet on the cruise.

Pavneet receiving the original conference drawing.

Photos courtesy of Pavneet Arora, Jennifer Claudio, Kim Nesbitt, Norbert Preining, Volker RW Schaa, Michael Sofka and Christina Thiele.

TUG 2016 in Toronto

Norbert Preining

In 2016, the TUG conference was held in Toronto, Canada. The following was originally published on my blog (`preining.info/blog/tag/tug2016`) and edited for publication. So you want to know what you missed if you weren't able to be there? Here are my very personal recollections!

1 First pre-conference excursion

This year, the TUG conference was held in Toronto, Canada, and our incredible host Pavneet Arora managed to put together a busy program of excursions and events around the real conference. The −1st day (yeah, you read right, the minus-first day), that is two days before the actual conference started, was dedicated to an excursion to enjoying wines at the wine estate Château des Charmes, followed by a visit to Niagara Falls.

What can I say, if the first thing after getting out of the bus is a good wine, then there is nothing to go wrong ...

I had arrived in Toronto already two days earlier in the late afternoon, and spent Friday relaxing, recovering from the long flight, walking the city a bit, trying to fix my cold, and simply going slowly. Saturday morning we met at a comfortable 10am in the morning (though still too early for me, due to jet lag and a slightly late evening before), but the first two hours of the bus drive allowed us to relax. Our first stop was the Château des Charmes, an impressive building surrounded by vineyards.

We were immediately started off with a sandwich lunch with white, red, and ice wine. Good start! And although the breakfast wasn't that long ago, the delicious sandwiches (at least the vegetarian ones I tried) were a good foundation for the wine.

After replenishing our energy reserves, we were ready to start our tour. Our guide, very, if not over, enthusiastic, explained that practically everything related to wine in Canada has been started at this Château from the current owner — the château system where farmer and wine producer are the same, import of European grapes, winter protection methods, wine making — I was close to forgetting our Roman and Greek ancestors.

At least she admitted that the ice wine was brought over by an Austrian — but perfection was done here, where the controls of the government are much stricter than anywhere else ... hmmm, somehow I cannot completely believe all this narrative, but at least it is enjoyable. So now that we know all about the history, we dive into the production process area, and the barrel space, always accompanied with extensive comments and (self-)praise.

After this exhaustive and exhausting round, we are guided back to the patio to taste another three different wines, a white (bit too warm, not so much my taste), a rosé (very good), and a red made from a new grape variety that has mutated first here on the Château (interesting). As I didn't have enough, I tried to get something out of the big containers directly, but without success!

Happy and content, and after passing through the shopping area, we boarded the bus to continue towards Niagara Falls. Riding by some quite nice houses of definitely quite rich people (although Pavneet told me that houses in Toronto are far more expensive than those here — how can anyone afford this?), we first have a view onto the lower Niagara river. A bit further on we are let out to see huge whirlpools in the river, where boat tours are bringing sightseers on a rough ride into the pool.

Only a slight bit further on we finally reached the falls proper, with a great view of the American Falls at full power, and the Horseshoe Falls further up.

We immediately boarded a boat tour making the short trip to the Horseshoe Falls. Lining up with hundreds and hundreds of other spectators, we prepare for the splash with red rain wear (the US side uses blue; forbid that any side would rescue a wrong person and create an illegal immigrant!). The trip passes first under the American Falls and continues right into the mist that fills all the area in the middle of the Horseshoe Falls. Spectacular impression with walls of water falling down on both sides.

Returned from the splash and having dried our feet, we walk along the ridge to see the Horseshoe Falls from close up. The amount of water falling down these falls is incredible, and so is the erosion that creates the brown foam on the water down in the pool, made up from pulverized limestone. Blessed as we were, the sun was smiling all day and we got a nice rainbow right in the falls.

The surroundings of the falls are less impressive — Disneyland? Horror cabinet? Jodel bar? A wild mixture of amusement park style locations squeezed together and overly full with people — as if enjoying the nature itself would not be enough. All engulfed by ever-blasting loudspeaker music. The only plus I could find in this encampment of forced happiness was a local craft beer brewer where one could taste eight different beers — I made it only to four, though.

Finally night was falling, and we moved down to the falls again to enjoy the illumination of the falls.

After this wonderful finish we boarded the bus and back to Toronto, where we arrived around midnight. A long but very pleasurable Day Minus One!

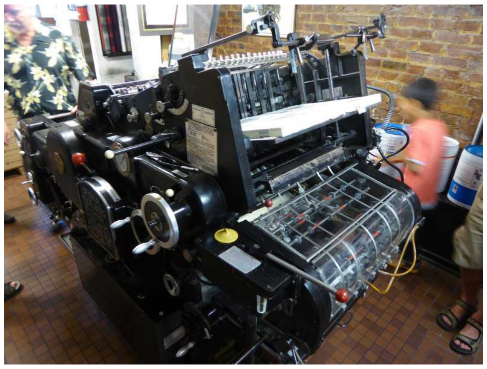
2 Second pre-conference excursion

The second pre-conference day was dedicated to books and beers, with a visit to an exquisite print studio, and a beer tasting session at one of the noted craft breweries in Canada. In addition we could get a view into the Canadian lifestyle by visiting Pavneet's beautiful house in the countryside, as well as enjoying traditional style pastries from a bakery.

In short, a perfect combination for us typography and beer savvy freaks!

This morning we had rather an early start from the hotel. Soon the bus left downtown Toronto and entered countryside of Ontario, large landscapes filled with huge (for my Japanese feeling) estates and houses, separated by fields, forests and wild landscape. Very beautiful and inviting to live there. On our way to the printing workshop we stopped at Pavneet's house for a very short visit of the exterior, which includes mathematics in the bricking. According to Pavneet, his kids hate to see math on the wall — I would be proud to have it.

A bit further on we entered Erin, where the Porcupine's Quill is located. A small building along the street, one could easily overlook this rare jewel! Even more so considering that according to the owners, Google Maps has a bad error which would lead you to a completely different location. This printing workshop, led by Tim and Elke Inkster, produces books in a traditional style using an old Heidelberg offset printing machine.



Elke introduced us to the sewing of folded signatures together with a lovely old sewing machine. It was the first time I actually saw one in action.

Tim, the head master of the printing shop, first entertained us with stories about Chinese publishers visiting them in the old cold-war times before diving into explanations of the actual machines present, like the Heidelberg offset printing machine.

In the back of the basement of the little studio is a huge folding machine, which cuts up the big signatures of 16 pages and folds them into bundles. An impressive example of tricky engineering.

Due to the small size of the printing studio, we were split into two groups, and while the other group got its guided tour, we grabbed coffee and traditional cookies and pastries from the nearby Holtom's bakery. Loads of nice pastries with various filling, my favorite being the slightly salty cherry pie, and above all the rhubarb-raspberry pie.

To my absolute astonishment I also found there a Viennese "Kaisersemmel", called "Kaiser bun" here, but keeping the shape and the idea (but unfortunately not the crispy crackly quality of the original in Vienna). Of course I got two of them, together with a nice jam from the region, and enjoyed this "Viennese breakfast" the next day morning.

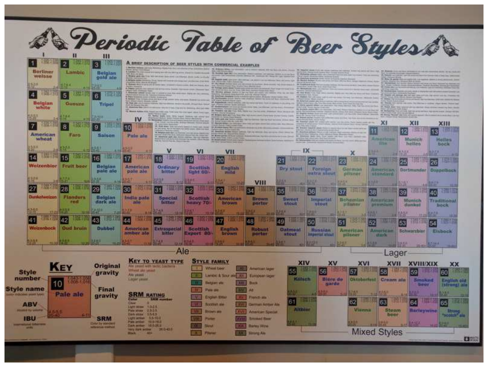
Leaving the Quill, we headed for a lunch in a nice pizzeria (I got Pizza Toscana) which also served excellent local beer — how I would like to have something like this in Japan! Our last stop on this day's excursion was Stone Hammer Brewery, one of the most famous craft breweries in Canada.

Although they won't win a prize for typography (besides one page of a coaster there that carried a nice pun), their beers are exquisite. We got five different beers to taste, plus extensive explanations on brewing methods and differences. Now I finally understand why most of the new craft breweries in Japan are making ales: ales don't need a long process and are ready for sale in rather short time, compared to e.g., lagers.)

Also at the Stone Hammer Brewery I spotted this very nice poster on the wall of the toilet. And I cannot agree more, everything can easily be discussed over a good beer — it calms down aversions, makes even the worst enemies friends, and is healthy for both the mind and body.

Filled with excellent beer, some of us (notably an unnamed US TEXnician and politician), stocked up on beers to carry home. I was very tempted to get a huge batch, but putting cans into an airplane seemed not to be an optimal idea. Since we are talking cans, I was surprised to hear that many craft beer brewers nowadays prefer cans due to their better protection of the beer from light and oxygen, both killers of good beer.

Before leaving we took a last look at the Periodic Table of Beer Types, which left me in awe about how much I don't know and probably never will know. In particular, I heard the first time of a "Vienna style beer" — Vienna is not really famous for beer, better to say, it is infamous. So maybe this is a different Vienna than my home town that is meant here.



Another two hour bus ride brought us back to Toronto, where we met with other participants at the reception in a restaurant of Mediterranean cuisine, where I could enjoy for the first time in years a good tahina and hummus.

All around another excellent day, now I'd just like to have two days of holidays; guess I'll need to relax in the lectures starting tomorrow.

3 First day

The first day of the conference itself started with an excellent overview of what one can do with TEX, spanning from traditional scientific journal styles to generating router configuration for cruising ships.

All this was crowned with an invited talk by Kevin Larson from Microsoft's typography department on how to support reading comprehension.

Pavneet Arora, Passport to the TEX canvas Pavneet, our never-sleeping host and master of organization, opened the conference with a philosophical introduction, touching upon a wide range of topics ranging from Microsoft, Twitter to the beauty of books, pages, and type. I think at some point he even mentioned TEX, but I can't remember for sure. His words set a very nice and all-inclusive stage, a community that is open to all kind of influences without any disregard or prejudice. Let us hope that this reflects reality. Thanks Pavneet.

Geoffrey Poore, Advances in PythonTEX Our first regular talk was a report on recent advances in PythonTEX, a package that allows including Python code in your TEX document. Starting with an introduction to PythonTEX, Geoff discussed an improved verbatim environment, `fvextra`, which patches `fancyvrb`, and improved interaction between Ti*k*Z and PythonTEX.

As I am a heavy user of `listings` for my teaching on algebraic specification languages, I will surely take a look at this package and see how it compares to `listings`.

Stefan Kottwitz, TEX in industry I: Programming Cisco network switches using TEX Next was Stefan from Lufthansa Industry Solutions, who reported first about his working environment, cruise ships (i.e., small floating towns) with a very demanding IT infrastructure which he has to design and implement. Then he introduced us to his way of generating IP configurations for all the devices using TEX. The reason he chose this method is that it allows him to generate at the same time proper documentation.

It was surprising for me to hear that by using TEX he could far more efficiently and quickly produce well designed and easily accessible documentation, which both helped the company as well as made the clients happy!

Stefan Kottwitz, TEX in industry II: Designing converged network solutions After a coffee break, Stefan continued his exploration into industrial usage of TEX, this time about using Ti*k*Z to generate graphics representing the network topology on the ships.

Boris Veytsman, Making ACM LATEX styles Next up was Boris, who brought us back to more traditional realms of TEX when he guided us into the abyss of ACM LATEX styles he tried to maintain for some time, until he plunged into a complete rewrite of the styles.

Frank Mittelbach, Alice goes floating: global optimized pagination including picture placements The last talk before lunch (probably strategically placed, otherwise Frank could continue for hours and hours) was on global optimization of page breaks, using an algorithm analogous to TEX's line breaking. This has been a wish among TEXies for decades! Frank showed us what can and cannot be done with current (Lua)LATEX, and how to play around with global optimization of pagination, using Alice in Wonderland as a running example. We can only hope that his package is soon available for us to at least play around with.

Thai lunch Pavneet organized three different cuisines for the three days of the conference. Today's was Thai with spring rolls, fried noodles, interesting orange noodles, and chicken something.

Michael Doob, baseball rules summary After lunch Michael gave us an accessible explanation of the most arcane rules a game can have — the rules of baseball — by using pseudocode. I think the total number of loc needed to explain the overall rules would fill more pages than the New York phonebook, so I am deeply impressed by those who can understand these rules. Some of us even wandered off in the late afternoon to see how a real game matched up with Michael's explanations.

Amartyo Banerjee, A Telegram bot for printing LATEX files Next up was Amartyo who showed a Telegram (as in messenger application) bot, running on a Raspberry Pi, which receives (LA)TEX files and sends back compiled PDF files. While it is not ready for general consumption (if you sneeze the bot will crash!), it looks like a promising application. Furthermore, it is nice to see how open APIs (like Telegram) can spur development of useful tools, while closed APIs (including threatening users, like WhatsApp) hinder this.

Norbert Preining, Security improvements in the TEX Live Manager and installer Next up was my own talk about beefing up the security of TEX Live by providing integrity and authenticity checks via GnuPG, a feature that has been introduced with the recent release of TEX Live 2016.

The following discussion gave me several good ideas on how to further improve security and usability.

Arthur Reutenauer, The TEX Live M subproject (and open discussion) Arthur presented the TEX Live M (where the M stands for Mojca, who couldn't attend, unfortunately) project: Their aim is to provide a curated and verified subset of TEX Live that is sufficiently complete for many applications, and easier for distributors and packagers.

We had a lively discussion after Arthur's short presentation, mostly about why TEX Live does not have an "on-the-fly" installation like MiKTEX. I insisted that this is already possible, using the "tex-on-the-fly" package which uses the `mktextex` infrastructure script, but also caution against using it by default due to delays induced by repeatedly reading the TEX Live database. I think this would be a worthwhile project for someone interested in learning the internals of TEX Live, but I am not sure whether I want to invest time into this feature myself.

Another discussion point was about testing infrastructure, which I am currently working on. This is in fact high on my list, to have some automatic minimal functionality testing — a LATEX package should at least load!

Kevin Larson, Reading between the lines: Improving comprehension for students Having a guest from Microsoft is rare in our somewhat Unix-centered environment, so big thanks to Pavneet again for setting up this contact, and big thanks to Kevin for coming.

Kevin gave us a profound introduction to reading disabilities and how to improve reading comprehension. Starting with an excursion into what makes a font readable and how Microsoft develops optimally readable fonts, he then turned to reading disabilities like dyslexia, and how markup of text can increase students' comprehension. He also toppled my long-term belief that dyslexia is connected to the similar shape of letters which are somehow visually malprocessed — this was the scientific theory from the 1920s through the 70s, but since then all researchers have abandoned this interpretation; dyslexia is now linked to problems linking shape to phonemes.

Kevin did an excellent job with a slightly difficult audience — some people being picky about grammar differences between British and US English and trying to derail the discussion, and even more the high percentage of typographically somehow sophisticated participants.

After the talk I had a lengthy discussion with Kevin about if/how this research can be carried over to non-Roman writing systems, in particular Kanji/Hanzi based writing systems, where dyslexia probably shows itself in different context. Kevin also mentioned that they want to add interword space to Chinese to help learners of Chinese (children, foreigners) parse text, and studies showed that this helps a lot in comprehension.

On a meta-level, this talk bracketed with the morning introduction by Pavneet, describing an open environment with stimulus back and forth in all directions. I am very happy that Kevin took the time to come in his tight schedule, and I hope that the future will bring better cooperation — at the end we are all working somehow toward the same ends — only the tools differ.

Dinner After the closing of the session, one part of our group went off to the baseball game, while another group dived into a nearby Japanese-style Izakaya where we managed to kill huge amounts of sake and quite an amount of food. The photo shows me after the first bottle of sake, while just sipping an intermediate small amount of genshu (a strong undiluted sake) before continuing to the next bottle.

An interesting and stimulating first day of TUG, and I am sure that everyone was looking forward to day 2.

4 Second day

The second day of TUG 2016 was again full of interesting talks, spanning from user experiences to highly technical details of astrological chart drawing, and graphical user interfaces for Ti*k*Z to the invited talk by Robert Bringhurst on the Palatino type family.

With all these interesting things there is only one complaint — I cannot get out of the dark basement and enjoy the city ...

After a evening full of sake and a good night's sleep we were ready to dive into the second day.

Kaveh Bazargan, A graphical user interface for Ti*k*Z The opening speaker of Day 2 was Kaveh. He first gave us a quick run-down on what he is doing for business and what challenges publishers are facing in these times. After that he introduced us to his new development of a command line graphical user interface for Ti*k*Z. I wrote "command line" on purpose, because the editing operations are short commands issued on a kind of command line, which give an immediate graphical feedback. The base of the technique is a simplified Ti*k*Z-like meta language that is not only easy to write, but also easy to parse.

While the set of supported commands and features of Ti*k*Z is not complete, I think the basic idea is a good one, with plenty of potential.

Matthew Skala, Astrological charts with horoscop and starfont Next up was Matthew who introduced us to the involved task of typesetting astrological charts. He included comparisons with various commercial and open source solutions, where Matthew of course, but me too, felt that his charts came off quite well!

As an extra bonus we got some charts of famous singers, as well as the TUG 2016 horoscope.

David Tulett, Development of an e-textbook using LaTeX and PStricks David reported on his project to develop an e-textbook on decision modeling (lots of math!) using LaTeX and PStricks. His e-book is of course a PDF. There was a lot of very welcome feedback — legally copyable (CC-BY-NC-ND) textbooks for sciences are rare and we need more of them.

Christian Gagné, An Emacs-based writing workflow inspired by TeX and WEB, targeting the Web Christian's talk revolved around editing and publishing using `org-mode` of Emacs and the various levels of macros one can use in this setup. He finished with a (sadly) incomprehensible-to-me vision of a future equational logic-based notation mode. I have used equational logic in my regular job, and I am not completely convinced that this is a good approach for typesetting and publishing — but who knows, I am looking forward to a more logic-based approach!

Frank Mittelbach, In memoriam: Sebastian Rahtz (1955–2016) Frank recalled Sebastian's many contributions to a huge variety of fields, and recalled our much-missed colleague with many photos and anecdotes.

Jim Hefferon, A LaTeX reference manual Jim reported about the current state of an unofficial LaTeX reference manual, which tries to provide documentation orthogonal to the many introduction and user guides available, by providing a straight down-to-earth reference manual with all the technical details required. He urged potential contributors to take a look (`http://home.gna.org/latexrefman`).

As I also had to write a reference manual for a computer language, it was very interesting to see how this dealt with many of the same problems I am facing.

Arthur Reutenauer, Hyphenation past and future: hyph-utf8 and patgen Arthur reported on the current state of the hyphenation pattern project, and in particular the license and usage hell they recently came into with large corporations simply grabbing the patterns without proper attribution. In a second part, he gave quick rough sketch of his design of a reimplementation of `patgen`.

Federico Garcia-De Castro, TeXcel? As an artist organizing large festivals Federico has to fight with financial planning and reports. He seemed not content with the abilities of the usual suspects, so he developed a way to do Excel-like bookkeeping in TeX. Nice idea! I hope I can use this system for the next conference I have to organize.

Jennifer Claudio, A brief reflection on TeX and end-user needs The last speaker of the morning was Jennifer who gave us a real-world end-user's view of the TeX environment, and the respective needs. This sort of talk is a very much welcomed contrast to technical talks and hopefully all of us developers take her suggestions to heart.

Jaeyoung Choi, MFCONFIG: Metafont plug-in module for the Freetype rasterizer Jaeyoung reported about an impressive project to make Metafont fonts available to `fontconfig` and thus windowing systems. He also explained their development of a new font format Stemfont, which is a Metafont-like system that can work also for CJK fonts, and which they envisage to be built into all kinds of mobile devices.

Michael Sharpe, New font offerings: Cochineal, Nimbus15 and LibertinusT1Math Michael reported on his latest font projects. The first two being extensions of the half-made, half-butchered, rereleased URW fonts, as well as an extended math font project.

I talked to him over lunch one day, and asked him how many man-days he need for these fonts, and his answer was, a lot: For the badly messed up new URW fonts, like Cochineal, he guessed about five man-months of work, while other fonts only needed a few days. We all can be deeply thankful to all the work he is investing into all these font projects.

Robert Bringhurst, The evolution of the Palatino tribe The second invited talk was from Robert Bringhurst, famous for his wide contributions to typography, book culture in general, as well as poetry. He gave a quick historic overview on the development of the Palatino tribe of fonts, with lots of beautiful photos.

Unfortunately, I was a bit disappointed that the presentation was more a listing of historical facts than his own ideas and thoughts. Of course, a person as accomplished as Robert Bringhurst is so full of anecdotes and background knowledge that it was still a great pleasure to listen and lots of things to learn, I only hoped for a bit more enthusiasm.

TUG Annual General Meeting The afternoon session finished with the TUG Annual General Meeting; Stefan Kottwitz wrote a separate report, following this one.

Herbert Schulz, Optional workshop: TeXShop tips & tricks After the AGM, Herb from MacTeX and TeXShop gave a workshop on TeXShop. Since I am not a Mac user, I skipped.

Another late afternoon program consisted of an excursion to Eliot's bookshop, where many of us stocked up on great books. This time again I skipped and took a nap.

Dinner In the evening we had a rather interesting informal dinner in the food court of some building, where only two shops were open and all of us lined up in front of the Japanese curry shop, and then gulped down from plastic boxes. Hmm, not my style I have to say, not even for informal dinner. But at least I could meet up with a colleague from Debian and get some GPG key signing done. And of course, talking to all kinds of people around.

The last step for me was in the pub opposite the hotel, with beer and whiskey/scotch selected by specialists in the field.

5 Third day

The last day of TUG 2016, or rather the last day of talks, brought four one-hour talks from special guests, and several others, where many talks told us personal stories and various histories. A great finish of a great conference.

Jennifer Claudio, The case for justified text Due to a strange timezone bug in my calendar program, I completely overslept a morning meeting and breakfast, as well as the first talk, so unfortunately I don't have anything to report about this surely interesting talk comparing justification in various word processors and TeX.

Boris Veytsman and Leila Akhmadeeva, Are justification and hyphenation good or bad for the reader? Still half dizzy and without coffee, I unfortunately couldn't follow this talk (with Leila joining us via video from Russia), and only woke up near the end when there was a lot of interesting discussion about speed reading and its non-existence (because it is simply skimming over text), and improvements on reading comprehension.

Charles Bigelow, Looking for legibility The last special guest, Charles Bigelow, presented a huge pool of research and work on readability, and how attitude and usage of fonts change over time. A very involving and well laid out talk, full of interesting background images and personal opinions and thoughts. Chuck also touched on topics of readability on modern devices like e-readers and mobiles. He compared recent developments in font design for mobile devices with their work on Lucida 20+ years ago, and concluded that both arrived at the same solutions.

A very educating and amusing talk packed full with information on readability. I will surely revisit the recording in a study session.

David Walden, Some notes on the history of digital typography David touched on many topics of the history of digital typography which he has experienced himself over the years: First the development of newspaper production and printing, then the evolution from simple text editors over word processors to full-fledged DTP programs. Finally he touched on various algorithmic problems that appear in the publishing business.

Tim Inkster, The beginning of my career Tim, our fantastic guide through his print shop the Porcupine's Quill on the second excursion day, talked about his private ups and downs in the printing business, all filled with an infinite flow of funny stories and surprising anecdotes. Without slides, or anything but his voice and stories, he kept us hanging on his words without a break. I recommend watching the recording of his talk because one cannot convey the funny comments and great stories he shared with us in this simple and so entertaining talk.

Joe Clark, Type and tiles on the TTC Joe unveiled the history of the rise and fall of underground types and tiles in Toronto. It is surprising to me that a small metro network as in Toronto can have such a long history of changes of design, layout, presentation. Some of the photos completely stymied me — how can anyone put up signs like that? I was thinking. To quote Joe (hopefully I remember correctly):

You see what happens without adult supervision.

Abdelouahad Bayar, Towards an operational (LA)TeX package supporting optical scaling of dynamic mathematical symbols A technical talk about an attempt to provide optical scaling of mathematical symbols. As far as I understand it tries to improve on the TeX way of doing extensible math symbols by gluing parts together at the font level. It seems to be highly involved and technically interesting project, but I couldn't completely grasp the aim of it.

John Plaice, Zebrackets: A score of years and delimiters John introduced us to Zebrackets, stripped parentheses and brackets, to help us keep track of pairing of those beasts. But as we know, zebras are very elusive animals, ... and so we saw lots of stripped brackets around. The idea of better markup of matching parentheses is definitely worth developing.

Charles Bigelow, Probably approximately not quite correct: Revise, repeat Chuck's second talk, this time on the history of the Lucida fonts, from the early beginnings drawn on graph paper to recent developments using FontLab producing OpenType fonts. A unique crash course through the development of one of the biggest families of fonts, and one of the first outside Computer Modern with support for proper math typesetting in TeX.

Aggressively legible!

This was one of the key phrases that popped up again and again — aggressively legible — mostly with negative connotations, toward too-fat symbols or too-big Arabic letters. But for me this font family is still close to my heart. I purchased it back then from Y&Y for my PhD thesis, and since then have upgraded to the TUG version including the OpenType fonts, and I use them for most of my presentations. Maybe I like the aggressive legibility!

Chuck slid in lots of nice comments about his partner Kris Holmes, the development practices in their work, stories of business contacts, and many more, making this talk a very lively and amusing, and at the same time very educating talk.

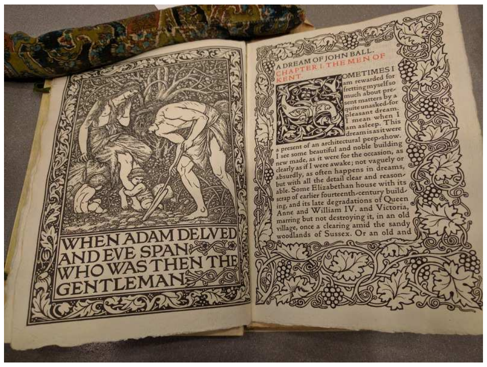
Joe Clark, Type and Tile Subway Tour This concluded the TUG conference talks, and we thanked Pavneet for his excellent organization. But since we still have up to two days more of excursions, many people dispersed quickly, just to meet again for a optional Type and Tile Tour — 3–5 subway stops with discussion of typesetting there.

This guided tour through the underground of Toronto, guided by Joe Clark who spoke in the morning on this topic, was simply too popular. I think there were around 25 participants when we left. I thought that this will not work out properly, and decided to leave the group and wander around alone.

Dinner The last program point for the day was dinner with a blues music concert at the nearby Jazz Bistro. Excellent live music in a bit slick and sophisticated atmosphere was a good finish for this excellent day. With Herb from MacTEX and his wife we killed two bottles of red wine, before slowly tingling back to the hotel.

6 Fourth day

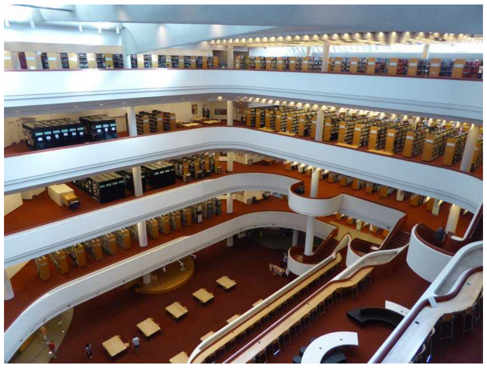
Talks have finished, and as a special present to the participants, Pavneet has organized an excursion that probably was one of the best I ever had. First we visited the Toronto Reference Library where we were treated to a delicious collection of rare books (not to mention all the other books and architecture), and then a trip through the Ismaili Centre Toronto and the Aga Khan Museum.


A DREAM OF JOHN BALL.
CHAPTER I. THE MEN OF KENT.
WHEN ADAM DELVED AND EVE SPAN WHO WAS THEN THE GENTLEMAN

Kelmscott press edition from 1892 of William Morris' A Dream of John Ball.

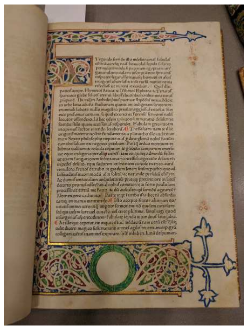
All these places were great pieces of architecture with excellent samples of the writing and printing art. And after all that and not to be left out, the conference dinner evening cruise!

Our first stop was the Toronto Reference Library. Designed by Raymond Moriyama, it features a large open atrium with skylights, and it gives the library an open and welcoming feeling. We were told that it resembles a teacup that needs to be filled — with knowledge.



The library also features running water at several places — the architect had the idea that natural ambient noise is more natural for a library than the unnatural silence that never happens anyway.

Originally there was lots of greenery hanging into the atrium, resembling the Hanging Gardens, but that has been scrapped due to financial reasons. But there was still this beautiful green oasis-like wall in a corner of the library.



We were guided first to the fifth floor where the special collection is housed. And what a special collection. The librarian in charge had laid out about 20 exquisite books starting from early illuminated manuscripts over incunabula to high pieces of printing art from the 18th and 19th centuries. Here we have an illuminated manuscript in Carolingian minuscule.

It was surprising for all of us in this special collection that all these books were simply laid out in front of us, that the librarian touched them and flipped pages without gloves, and above all, that he told us that if one wants, it is common practice to check out these books for study sessions and enjoy them on the spot in the reading room. I don't know any other library that allows you to actually handle such rare and beautiful specimens!

In one of the books I found by chance a map of my hometown of Vienna. On this map from very old times, the place where I grew up is still uninhabited somewhere in the far upper right corner of the map. Times have changed.

After we left this open and welcoming treasure house of beautiful books, we moved to the Aga Khan Museum and Ismaili Centre Toronto, which are standing face-to-face separated by some water ponds in the Aga Khan park a bit outside of central Toronto. Below is the Ismaili Centre as seen from the Aga Khan Museum entrance. The big glass dome is the central prayer room, and is illuminated at night. Just one detail — one can see in the outer wall one part that looks like glass, too. This is the prayer alcove in the back of the prayer hall, and is made from huge slabs of onyx that are also lit up in the night.

The Ismaili Centre, designed by Charles Correa, combines modern functional and simple style with the wonderful ornamental art of the Islam heritage. The inside of the Ismaili Centre features many pieces of exquisite art — calligraphy, murals, stone work.

Following the Ismaili Centre we turned to the Aga Khan museum which documents Islamic art, science, and history with an extensive collection. We didn't have much time, and in addition I had to do some firefighting over the phone, but the short trip through the permanent collection with samples of excellent calligraphy was amazing.

Banquet cruise After returning from this lovely excursion and a short break, we set off for the last stop for tonight, the dinner cruise. After a short bus ride we boarded our ship and off we went. Although the beer selection was not on par with what we were used to from craft breweries, the perfectly sized boat with two decks and lots of places to hang around invited us to many discussions and chitchats. And finally we could enjoy also the skyline of Toronto.

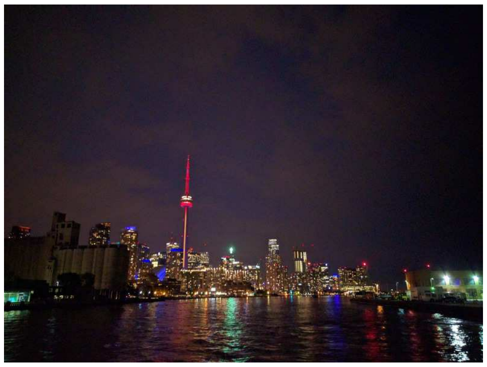
After the dinner we had some sweets, one of which was a specially-made cake with the TUG 2016 logo on it. I have to say, it was not only this cake but the whole excellent, overwhelming, food we had during all these days, that will make me go on a diet when I am back in Japan. Pavneet organized for the lunch breaks three different style of kitchens (Thai, Indian, Italian), then the excursions to local brewers and and and... If it wouldn't be for TeX, I would call it a "Mastkur".

During the cruise we also had a little ceremony thanking Jim for his work as president of TUG, and above all Pavneet for this incredible, well organized conference. I think everyone agreed that this was the most exceptional TUG conference in some time.

During this, Pavneet also announced the winners of the TUG 2016 fountain pen auction. These pens have much history and travel behind them (`tug.org/tug2016/pens.html`), and were presented to the special guests of the conference. Two remaining pens were auctioned with proceeds going to TUG. The first one was handed over to Steve Grathwohl, and — to my utter surprise — the second one to myself. So now I am a happy owner of a TUG 2016 fountain pen. What a special feature!

Just one more detail about these pens: They are traditional style, so without ink capsules; one needs to insert the ink with a syringe. I guess I need to stock up a bit at home, and more importantly, train my really ugly handwriting, otherwise it would be a shame to use this exquisite tool.

We returned to the harbor around 10pm, and back to the hotel, where there was much greeting and thanking at the end of a wonderful day.



I will leave on Friday morning to meet with friends, thus I will not be participating in (and not reporting on) the last excursion of TUG 2016 to the Georgian Bay area. I will leave Toronto and TUG 2016 with (nearly) exclusively good memories of excellent talks, great presentations, wonderful excursions, and lots of things I have learned. I hope to see all of the participants at next year's TUG meeting — and I hope I will be able to attend it.

One more thanks to Pavneet, you have done an incredible job. And last but not least, thanks to your lovely wife for letting you do all this, I know how much time we stole from her.

◇ Norbert Preining
Ishikawa, Japan
`norbert (at) preining dot info`
`http://www.preining.info`

Excursion to Georgian Bay

On Friday, although many participants had left, there was one more excursion, to the Georgian Bay area. This large bay, extending off of Lake Huron, was the inspiration to the Canadian impressionist landscape painters known as the Group of Seven, who were active from the period before the first World War until the early 1930s.

We first visited a public beach on the bay — more like a large lake — where several of our group took advantage of the opportunity to swim in the calm water. (They assured me that it was quite pleasant.) The rest of us enjoyed the scenery, woods and islands as far as the eye could see, with small cottages along the shore, and imagined what it would look like in other seasons.

From Georgian Bay we made our way to the McMichael Canadian Art Collection, a public gallery in the village of Kleinburg, devoted largely to the Group of Seven. Built around the collection of Robert and Signe McMichael, and housed in their much-expanded home, the collection, buildings, and property on which it is located were donated in 1965 by the McMichaels to the Province of Ontario. Since then, the collection has been augmented by other Canadian works donated by collectors, as well as by artists themselves. It now includes many contemporary pieces, and both traditional and contemporary works by First Nations and Inuit artists.

Our guide first introduced us to the works of the Group of Seven. Although many of the Collection's holdings of this Group were temporarily away for exhibit in larger cities' museums, the core collection on display showed a remarkable sensitivity to the Canadian landscape in all its moods. The building itself was designed by the Canadian architect Leo Venchiarutti to be an appropriate home for the collection, as well as (before it became a gallery) for the McMichaels. Many of the windows look out on the beautiful woods surrounding the building.

In addition to the permanent collection, several special exhibits were on display. These included a showing of contemporary textile art by Colleen Heslin; a "studio" selection of colorful paintings by Jack Bush; and drawings and paintings from the period of the World War by A.Y. Jackson (one of the Group of Seven) and Tom Thomson (closely affiliated with the Group of Seven, though not a member; he died before the Group got its name) showing their influences on one another.

One area we didn't have time to explore adequately was the sculpture garden, which occupies the grounds of the Collection. The pieces nearest the main building, and those we could glimpse farther away in the wooded parkland, are a compelling invitation to return.

This is the 50th anniversary of the McMichael Collection as a public institution. The website, `mcmichael.com`, is well worth a visit.

◇ Barbara Beeton

TUG 2016 Annual General Meeting informal report

Stefan Kottwitz

The TUG Annual General Meeting for 2016 was in the afternoon of the second day. Jim Hefferon, the current TUG President, moderated it. He started with a few slides. First, he gave a summary of the TUG bylaws and goals, that are, further summarized, maintaining TeX, supporting TeX users and caring for fine typography. Following those objectives, the TUG sponsors conferences, development of fonts, and specific activities and projects such as CTAN and LuaTeX development.

Jim introduced the board of directors. Everybody on the board in the room stood up, so everybody knows who they are. At last year's meeting in Darmstadt they sat in front of us, this year they just stayed in the audience. Jim skipped the financial information on purpose, saying that probably few people in this audience were interested in financial details; he'll provide them for anybody interested. They are also publicly available on the TUG web site, since TUG has to publish them as a tax-exempt organization. As a general remark, he mentioned that TUG maintains its budget very conservatively.

However, there's the challenge that the number of members has fallen steadily. In 2000, we had 2211 members, in 2015 only 1260, and 1124 as of June 2016. The membership fees have been raised over time. One might see this kind of connected, either could be partially a consequence of the other, but: as long as TUG provides public services such as CTAN support and TeX Live development and more, also for non-members, and the TUG office, with revenue mainly from membership fees, the fees may get higher. As long as there's no relevant change of the model.

This led us to an open discussion, with Jim as the moderator. It started similarly to last year, and raised some of the same questions or suggestions: what should be changed, what could be done, up to whether the existence of TUG as an organization is still relevant. The latter was quickly answered. How to get developers together, such as at the present conference, how to get funding and to finance projects, without an organization?

The question was raised of how many members we would like to have, what would be desirable — stay small or grow — before doing anything about it. Somebody said, and that's good: we should be much bigger to be representative of the very many people using TeX. Several people confirmed that there's a general decline in membership numbers at many societies. Today, young people seem to be less interested in societies and paper journals.

We had a members-bring-members activity last year. We had tried different things. It wasn't summarized what has been done, I missed that. It would be good to touch ground before new suggestions come. [*Editor's note*: Although not mentioned at the meeting, Boris Veytsman reported last year's results in "The continuing TUG membership drive", *TUGboat* 37:1, pp. 6–8, `tug.org/TUGboat/tb37-1/tb115veytsman.pdf`. The campaign continues this year: `tug.org/membership`.]

One such new suggestion was a lifetime membership. But if we all use that, there would be no subsequent membership fees at all. The 5-years-limited lifetime membership was discarded as nobody wants to kill anyone. We started to look beyond member numbers ...

The suggestion came to raise the *TUGboat* journal from a member's journal to a premium journal with subscription options. It would not take too much, it was believed, we would have the ability to produce a high quality journal. Libraries usually don't have the option to become a society member, but would be able to subscribe to a journal. Institutions where we study, teach, and work, could subscribe. [*Editor's note*: There was no chance at the meeting to discuss it, but *TUGboat* currently has, and has always had, a subscriber option available.]

Also not new: the suggestion to improve the TUG website came up. To attract users to return to the site on a regular base, such as by a blog. Well, also blogs experienced a decline. I try to support and to encourage blogging: on the one hand I maintain `TeXample.net` with its blog aggregator to keep up with blog posts, on the other hand the three web forums `LaTeX-Community.org`, `TeXwelt.de`, and `goLaTeX.de` present recent blog posts in their side bar, so any TeX user jumping in (e.g., from Google) anywhere in the forum can see the posted list. You post on your blog, and the world can see it at various places.

I have other thoughts too, though it's just not my thing to stand up in public. I am sending a few suggestions to the TUG board ...

◇ Stefan Kottwitz
`latex-community.org`

Sebastian Rahtz (1955–2016): A brief memoir

Lou Burnard

I wish I could discuss this with Sebastian. I know that if I could, it would be a better piece, because everything I have worked on with Sebastian has always been better as a result. He had that rare ability to understand what you were trying to achieve, perhaps better than you did and to push you in the right direction, if you were pointed that way, or gently dissuade you if you were not. He saw things clearly, and he had opinions about the right and the wrong way of going about a thing, which in some people might have been insufferable, but in him was not. Far from it. No-one who enthused about Dr Who and about dark Scandinavian thrillers on the telly, about Bach and about Wagner, about the Moomins, and Arthur Ransome, and Rudyard Kipling could be considered insufferable.

I think I must have first met Sebastian at the start of the 1980s, when he was working in the Oxfordshire Archaeological Unit. He was one of the small number of proto-geeks frequenting Oxford University Computing Services who managed to make its pioneering Lasercomp Typesetting System sit up and say Uncle (or in his case the equivalent in Greek). I got to know him better when he left Oxford and became a lecturer in something called Humanities Computing at Southampton University in 1985 or thereabouts. We had a cheerfully irreverent email correspondence making fun of our elders and betters and bickering about what he called 'Sludgemull', the ancestor of XML. We also engaged in data-trafficking of dubious legality. His students extended my transcriptions of the complete works of Bob Dylan, and I provided him typesetting tapes of English dictionaries to reformat. (Yes, dear reader, this was back in the day when the most reliable way of transferring more than a megabyte or so of data between different computer systems involved huge reels of magnetic tape in different proprietary formats.)

In 1985, he organised one of the first UK conferences about how to teach IT skills to humanities students. This was remarkable at the time because delegates were provided on arrival with a copy of the proceedings in the form of a decently typeset book. The subjects covered seem extraordinarily technical for a humanities focussed conference: embracing database technology, information modelling, and even logic programming, then much in vogue. He was however skeptical about whether 'humanities computing' actually meant anything much and remained a fearless critic of some of its more pretentious advocates on the email discussion lists and bulletin boards which were the only kind of social media we had in those distant days.

But the first big thing in Sebastian's professional life was not 'Humanities Computing' as such; it was TeX and the TeX community. For about fifteen years his professional energies were devoted to developing and promoting that celebrated open source typesetting system. He became a world-recognized mover and shaker within its community, setting up its first online archive, producing numerous distribution packages, and writing two or three best-selling textbooks. Others know much more than I do about this period of his life; I note simply in passing what an excellent preparation it provided for his work with the Text Encoding Initiative. Because TeX is not only a typesetting system, but also a community of enthusiasts, empowered by the system's openness to tweak and modify it into a state of perfection. Or confusion.

In the 1990s, Sebastian had a short spell working outside academia, first as a consultant at CERN, where he witnessed first hand the arrival of the World Wide Web, and then at Elsevier, where he was actually paid to work on TeX. But at the start of the present century, Oxford University Computing Services (as it then was) recruited him, initially with the brief of reorganising its chaotic documentation systems. The right answer, Sebastian decided, with only a little prompting from me, was to convert everything to XML, more specifically TEI. And so began his second major international collaboration, in which I am very proud to have been involved.

The Text Encoding Initiative had been in use amongst a small and rather various band of cognoscenti for more than a decade; its declared goal was to define a common format for the representation of written texts of every kind, in all languages, from all periods of time, for every kind of scientific application. Naturally, this was expressed as a very complicated modular SGML schema, the full ramifications and internal workings of which possibly a handful of people in the world understood, on a good day with a following wind. By the end of the nineties, and with the arrival of XML, to say nothing of Unicode or the web, the TEI was starting to look decidedly antiquated: an elegant piece of research perhaps but hardly a practical technology.

Sebastian paid the TEI the compliment of taking it seriously, and worked hard at making it realise its full potential. He asked awkward questions about how all that elegant text encoding was actually supposed to be processed, and (when I waved my hands about by way of response) both proposed and implemented solutions, real solutions, using actual software. He led the development of a new technical framework within which the TEI re-expressed itself as a modular and customisable XML schema, and he wrote the library of XSLT stylesheets which enabled both publication and maintenance of succeeding versions of the system, from 2005 onwards.

Remarkably, he did this in a way that was entirely faithful to the TEI's original design goals of accessibility and uniformity of documentation, but taking advantage of the vastly improved range of infrastructural tools and methods which had become available since that initial design. We should not forget that although designed before the existence of the World Wide Web, the TEI anticipates, even takes for granted, the wide availability of web technologies which only came into being many years later; it anticipates, for example, the kind of intimate linking between documents and data we now recognize as linked open data. Michael Sperberg-McQueen, the original principal editor of the TEI Guidelines, liked to say that when confronted by a choice between expressing in one's encoding what is true of a document and what is expedient for processing, truth should always take precedence. Sebastian's work reminded us that the claims of expedience should not be entirely neglected.

His contribution was not only technical however. During the long drawn out process by which the TEI transformed itself from short-term well-funded research project into long-term self-sustaining research infrastructure, he played a major role, working closely with both Technical Council and Board of Directors of the new TEI Consortium. Of course he was not alone in orchestrating this transformation, but his voice was the one consistently nudging the TEI to adopt both open licencing policies and open working practices, thus doing all that could reasonably be done to ensure its longevity. I remember his being quietly jubilant, as we stood on a railway platform waiting for the train back to the Gare du Nord after a session at the French national standards body AFNOR where the fledgeling TEI Council had agreed once for all to licence all its products under the GPL.

I am lucky to have shared many such moments with him. I remember a long bus journey in Norway during which we reviewed and fixed all the outstanding problem areas in an ancient TEI working paper concerning the move from SGML to XML. I remember thrashing out details of what became the ODD specification language with him, in countless email messages, several airport lounges, and at least three different Eurostar terminals. And I remember the evening following an exhaustive TEI training workshop in Alicante during which (after rather a lot of rioja) we planned out the structure and content of the definitive TEI training manual.

His technical contributions were prodigious: it became a standing joke in the Technical Council whenever a particularly thorny issue was being discussed that by the time the Council — not the least argumentative bunch of people — had formed a consensus as to how it should be resolved, Sebastian would have already implemented and tested an XSLT stylesheet to do the job. But he was also and always a collaborative animal: he hated what he called 'magic' in software systems — secret by-ways in the code depending on undocumented or special cased data or situations. He wanted there to be a reasonable possibility that a reasonably intelligent person should be able to take over and run with everything he had developed. Sadly, this goal is now something the TEI Technical Council has to put to the test.

I think his later career at Oxford was marked by the same insights. He obtained national funding for a project called OSS Watch, which investigated the role of Open Source software in academia, and developed over the years into a consultative service, providing reliable and objective data about the role of open software provision in the academic context. He became a well-liked and respected member of the senior management team at OUCS in 2012, surviving the department's many vicissitudes and reorganisations to become the University's Chief Data Architect, with strategic responsibility for many aspects of policy and practice across the University. I won't try to list all the different projects and services that benefited from his expertise. I will however say that he took each one seriously, so long as it was going somewhere, but was ready to move on as soon as it reached fruition. That seriousness, that commitment, was surely a major cause for the real affection and respect which his colleagues felt for him, not only in the University, but in each of the many scientific communities in which he participated, all of which I think felt equally bereft when he was taken from us earlier this year.

Sebastian's personal life was full of happiness and incident and variety, and he was blessed with a wonderful loving family. There are many who count themselves fortunate to have shared some small part of his domestic life, whether IRL or elsewhere (for he was a great Facebooker), to have eaten his excellent bread, to have enjoyed his unstinting hospitality, to have witnessed his joy and pride in his children, his love of life, of running, of great art, and of all that makes up our shared culture.

And perhaps the most important lesson he taught us was the need to engage with our fellows, no matter how contrary they may seem. We are all dead in the long run. Only by engaging the support of our fellows can we hope to make possible any kind of continuity for all those things that (like him) we care so much about.

R.I.P. — S.P.Q.R Sebastian Patrick Quintus Rahtz (13.2.1955–15.3.2016)

Frank Mittelbach and Joan Richmond (translator)*

There are a small number of people in the worldwide TEX community whose activities in the nineties have been instrumental in shaping the TEX world as we know it today. Sebastian Rahtz, known also by the abbreviation SPQR, was without any doubt one of this group.

Sebastian was born as the fifth child (Quintus) of a family of archaeologists, which shaped the whole of his future life. He began his career with an M.A. in archaeology and as late as 2009 he was still describing the Protestant graveyard in Rome as the place in which he most enjoyed being [5].

Sebastian, colorful as always in one of his striking outfits — 2005

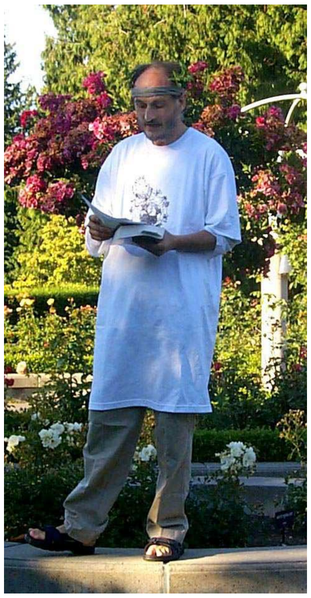
Already during his early time as a university lecturer in Humanities Computing at Southampton University, he became involved with typesetting classical texts, such as "The Lexicon of Greek Personal Names" (LGPN), and at some point he came in contact with TEX, which then became the centre of his work for almost two decades.

After Southampton, he worked for some time as a freelance consultant at several places, including CERN, where he worked with Michel Goossens. After that he took a post with Elsevier, where he was principally involved with TEX and its use in scientific publishing. Roughly at the turn of the millennium he moved over to become Information Manager at the Oxford University Computing Services (OUCS), and in the following years undertook various positions in higher management there.

Sebastian was a pragmatist, who did things when they had to be done, so it is not surprising that a number of things which were significant for the spread and further development of TEX and LATEX can be traced back to him. A great deal of this was confined to "behind the scenes" operations, as community service, and consequently is not necessarily known by the substantial majority of today's TEX users, but all these activities were hugely important at the time and exercised a decisive influence on the way TEX is presented to its users today.

In the eighties crucially important aspects were the further spread of TEX, the porting of the programs to new operating systems, the development of printer drivers for new devices and the like. Distributions existed as "tapes" for individual operating systems and Sebastian began his TEX life as a coordinator and distributor of tapes for SunOS.

His engagement in the maintenance and expansion of the Aston TEX archive (working with Peter Abbott and others) was a logical next step, from which CTAN (the Comprehensive TEX Archive Network) was subsequently born, so that Sebastian is effectively one of the founding fathers of CTAN as we know it today.


Sebastian at the "Poetry Contest" — Vancouver 1999

In those days, an Internet connection for most TEX users was either rudimentary (a 300kbps modem was usual) or not available at all, so that the distribution of TEX software via floppy discs, CDs and later DVDs became of decisive importance with the rise of distributions for Atari, PC, Amiga and other computer systems. Anyone who still owns a TEX Live disc from that period can verify that Sebastian's name appears as editor. He was the one who constructed the first TEX Live installation (based on the Unix distribution of Thomas Esser), and over the next few years, together with a few other volunteers, he continually improved it. The first seven or eight TEX Live productions carry his name as the responsible editor into the year 2004.

As I see it, both CTAN and TEX Live are vitally important milestones in the history of TEX, without which we would perhaps no longer know and use TEX and LATEX today, since they were absolutely decisive for wider distribution of the software.

But Sebastian's name is also linked to several macro packages which are still important today, despite the fact that he played down the importance of this work in an interview with Dave Walden [5]: these

* Ms Richmond kindly agreed to translate the German text that originally appeared in *TEXnische Komödie*, the journal of the German TEX user group, DANTE e.V. [4]. She is the author of the book *Nine Letters from an Artist, The Families of William Gillard.*

include the early versions of `graphics`, produced with David Carlisle, and the monumental package `hyperref` which subsequently saw further development by Heiko Oberdiek.

In the new millennium Sebastian turned his attention to new tasks outside the world of TEX, among which are his activities on TEI (Text Coding Initiative) and OSS Watch. It is fair to say that the XML world fascinated and absorbed him in the same way that the TEX world had done decades before.

Sebastian was in many respects a model for us to imitate. He never left construction sites behind him (such as one encounters all too often in the world of TEX and elsewhere), but always left behind him work that was ordered and finished, and — when he no longer had enough energy or interest in something — he never clung to it but happily handed it on in good time and in good condition to a successor.

Sebastian "preaching" about XML and Passive TEX, with a t-shirt from San Francisco (1997); in our hearts we are all traditionalists, I still own this too — Oxford 2000

I no longer recall exactly when I got to know Sebastian personally, but it must have been sometime at the beginning of the nineties. In the course of the following years we attacked various projects together. The most important of these are the NFSS (New Font Selection Scheme for LATEX), in which he undertook the integration of the 35 standard PostScript fonts and later the book projects *The LATEX Graphics Companion* [1, 2] and *The LATEX Web Companion* [3].

Ever since then I have valued him as both colleague and friend, and there are a number of activities which I would like to remember. To start with, the nightly coding sessions (via e-mail), during which I once asked him how he could combine that with the care of his little daughter. To which he replied that she kept him awake, lay on his lap and helped him. Whereupon he sent me a picture of Matilde (6 months old) and himself "working together". Or a walk on a free day during a conference, during which he dragged me forcibly into a book shop, ran straight to the children's section, hunted out a book that was completely unknown to me, and said "Buy". It looked rather odd, with the picture of a boy wearing glasses, but I followed his advice, bought it, and began to read it that same evening. By four o'clock the following morning I had finished it, and would rather not want to know how my lecture went that morning. The book was by a still fairly unknown woman writer at that time, and was called "Harry Potter and the Philosopher's Stone".

Sebastian was a "philosopher in his own right", with a lovable, dry English sense of humor, which often delighted me. This spring, his journey ended due to cancer. My thoughts are with him and his family.

Rest In Peace, S.P.Q.R.

References

[1] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LATEX Graphics Companion: Illustrating Documents with TEX and PostScript*. Addison-Wesley series on Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, 1997.

[2] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voß, editors. *The LATEX Graphics Companion*. Addison-Wesley series on Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, second edition, 2008.

[3] Michel Goossens and Sebastian Rahtz. *The LATEX Web Companion: Integrating TEX, HTML, and XML*. Addison-Wesley series on Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1999. With Eitan M. Gurari and Ross Moore and Robert S. Sutor.

[4] Frank Mittelbach. Sebastian Patrick Quintus Rahtz. *Die TEXnische Komödie*, 2/16, S. 24–27, Mai 2016.

[5] David Walden. Interview with Sebastian Rahtz, 2009. `https://tug.org/interviews/rahtz.html`.

Sebastian among his family: Inês, Leonor and Matilde — 2014

Frank Mittelbach and Joan Richmond

Interview: Pavneet Arora

David Walden

Pavneet Arora has participated in TUG annual conferences since 2010 and is the local organizer of TUG 2016 in Toronto.

Dave Walden, interviewer: Please tell me a bit about yourself.

Pavneet Arora, interviewee: I have lived a somewhat nomadic life. I was born in Madras, India, far from our nominal family home in Punjab as my father was an Army Engineer in the EME Corps (electrical and mechanical engineers). At a young age, my mother, by then a widow, emigrated to Canada on her own, and we established ourselves here.

My education was a product of the times, I suppose, coming as it did right after the 1960s. This was all in the public school system in Toronto, Canada.

I grew up in an inner city neighborhood well before it became fashionably gentrified, but all along I have had inspiring teachers. In primary school, we had a wonderful teacher, Mr. Leeming, who was exploring open concept, mixed-grade classrooms with huge — and I mean huge — spaces dedicated to terrariums and aquariums filled with animals. I mean, how many Grade 5 students can say that their classroom housed its very own boa constrictor, and had watched it feed? He would regularly, after getting permission from our parents, pile a bunch of us kids in a Ford Econoline and take us out to a rural farm for a week, just to have us see that the planet is more than concrete and steel. It was a very hopeful time.

Later, in high school, I had similar experiences where teachers lifted me from the conventional path. I got into calculators at an early stage as my mother had purchased a TI SR-50A — the SR stood for Slide Rule so you can imagine the era — for my mathematician grandfather who declined, saying that he preferred the one that he had been given at birth, and simultaneously handed it to me. My electronics teacher, Mr. Keen, observing that interest, one Friday sent me home with an HP-35 (!) and the booklet *Enter vs. Equals*, and asked me to come back and let him know what I thought. (HP's advertising said to "ask your dealer" for the booklet, which explained entering arguments and operators for a calculation onto a stack for evaluation according to reverse Polish notation.)

DW: What did you think?

PA: It was an epiphany, the first of many. It was as if your vision had been limited to the end of your apartment balcony, and then someone pointed out the horizon. I began to dream RPN, and then pre-ordered an HP-34C programmable scientific calculator when it was announced. It took me four months of saving up from my part-time job to pay for it.

DW: What about your other teachers?

PA: My literature teacher, Mr. Speichert, thought nothing of chucking everything from French existentialists, to Russian heavyweight writers at a bunch of young teenagers. So I would come out of an astronomy unit in physics one minute, and then be reading Ivan Turgenev in the next. It all just seems so far-fetched to me looking back now. My history teacher, Dr. Heydeloff, guided us through European history and economic theory through the lens of his own childhood having grown up in Germany through two horrific wars. At a very young age we were being given the vocabulary to understand and question our wider world. We were the beneficiaries, I think now, of a confluence of two trends: the idealism of the 1960s, with those who entered teaching with it as its touchstone, and a wave of extraordinary European emigres who brought with them qualifications and life experiences well beyond the norm.

Perhaps I was just impressionable, but all of these imprinted on me the idea that one is truly free when one is able to explore knowledge unfettered, and that literacy is the most powerful force for the protection of personal liberty. Of course, it does get more and more difficult as one takes on quotidian responsibilities, but I have railed against accepting convention, and this has meant I have often lived with economic uncertainty throughout my professional career.

I am very excited to see that we are now celebrating young people who aspire to be tinkerers. I think the term "Maker" is perfect, and it is uplifting to see the success of efforts like Raspberry Pi, and the now ubiquitous availability and acceptance of open source software.

DW: How did you first become involved with TEX?

PA: Ironically, my typesetting history has been more aligned with troff than TEX. My father's younger brother worked at Bell Labs in its heyday, and so I was exposed to Unix at an early age. I must still have the documentation for Documenter's Workbench Tools somewhere in my library along with a full printed set of System V man pages. I remember him recounting to me that the secretaries, which I realize is now a term relegated to near antiquity, took training courses in troff and were quite adept at cranking out documents using troff markup.

And then when I was working on my Master's thesis, the university I attended had a format package for troff, so I gravitated towards that. It was at that time that one of my grad student friends chose, instead, to use TEX, which was novel, but I was too immersed in my own work.

DW: What university was that?

PA: It was at the University of Waterloo that I did both my undergrad and Master's in the Dept. of Systems Design Engineering, a programme that focused on inter-disciplinary aspects of engineering. I ended up being fascinated by statistical pattern recognition, and so that is what I pursued for my Master's.

DW: What did you do after university?

PA: I soon recognized, after my first few jobs, that I was not a good fit for the corporate world. Along with my business partner, we had a small venture that, truth be told, wasn't very successful. The short explanation is that what I had to offer, the market rejected. I say this with very few regrets, and without the slightest wish to sound maudlin. The most that one can reasonably expect is, to put it in baseball terms, the chance to step up to the plate. After that it is a combination of one's skill, judgment, and timing that determines the outcome. Whatever the outcome, though, it doesn't diminish in the slightest that to take bat in hand and swing for the fences is in itself a privilege: this is what gives entrepreneurship its vibrancy.

I wrote an early text-based email agent using the curses programming library, which my partner and I hoped would find a place with "business" Unix users. And then, for a decade we latched onto NeXTSTEP and OpenSTEP, and developed applications for it. NeXTSTEP arrived with native OOPS, Display PostScript, Bit blit printing technology, Unix, the very elegant syntax of Objective-C, and applications like Lotus Improv. This, not too long after Microsoft Windows had just managed to go from tiled windows to overlapping ones! We thought that we had stepped into the future.

We misjudged just how enduring the beige box hegemony would be, and how long it would last. I think that the PC era is mistakenly associated with personal computing. In reality, DOS and Windows were adopted mainly by corporations, whose employees then brought them into their homes. The real personal computing revolution only happened with mobile.

We had been unsuccessfully bashing our heads against the Microsoft juggernaut for so very long. You dig a financial hole, and it is hard to ever really catch back up. I was thoroughly worn out and dejected with the technology sector by this time. Somehow I ended up landing in construction.

This was a tremendously gratifying experience. It released me from being bound by the very narrow definition of success that I had. I was doing something practical with my hands, and it all hearkened back to the idea of Quality that Robert Pirsig talked about in *Zen and the Art of Motorcycle Maintenance.* I met some amazing craftsmen, many of whom I consider my friends today.

Breaking communal bread, so to speak, on construction sites led to some unusual conversations. I remember once someone initiated a conversation about the Higgs boson; not long earlier I had read Herman Wouk's *A Hole in Texas* which was dedicated, I think, to his scientist brother Victor.

It also amazed me to find a very high level of numeracy amongst the trades in spite of their often having struggled in school. This being North America, there was a fluency in fractions — for example, 1/16 inch and 1/32 inch with carpenters — that I found simply dazzling, and it convinced me that some parts of mathematics, at least, should be accessible to nearly everyone given enough practice. You hold a combined metric/Imperial units measuring tape — the type common in Canada — and you are holding an everyday example of nomography. How amazing is that?

It also got me into architecture, and in particular the works of Frank Lloyd Wright. I ended up visiting quite a few of his houses around this continent before finally building a home inspired by his Prairie style some years later on. I named it Arkinia, which is a play on words on the ideas of "a home for 'our kin" ', and the "ark in" which we could find shelter or sanctuary.

Here is a photo of the brass plate that I had made in an Arts and Craft font, in 2011, while I was in India for TUG Trivandrum:

Allow me to relate a story about this....

I had walked by a hole-in-the-wall sign shop located deep in the pedestrian bazaar near my mother's apartment in Gurgaon just a couple of days before returning to Toronto. The proprietor wasn't there, but I got his mobile number from the person manning the shop on his behalf.

Later that day I called him from IGI (Delhi) airport on my way out to Pune where I was going just for a day, and asked him how long it might take to have a plate made and how I might get him artwork. He said three days, and a TIFF file.

When I asked him about a deposit he said not to worry, just get him the details and he would work on it — I could pay him when I came to visit him at the shop! I also explained that three days would be too long, as I would be leaving before then. He assured me that he would get it done before I was to leave.

I prepared the artwork while waiting for my flight. Initially, I attempted it in LaTeX, but was unable to get the font to work, and so switched to GIMP under Ubuntu on my laptop, outputting to TIFF. I then sent it along to him via email, all while sitting in the departure lounge. IGI had free WiFi long before many other airports did, and it was commonplace to see people pounding their laptop keyboards.

Well, upon my return to Delhi I phoned the shop, and of course the sign wasn't ready. I was told that it would be a few hours yet, and if it was later than that he would deliver the plate to my mother's apartment. I was to leave for Toronto that evening with not much time remaining before my taxi was to arrive. I went to the shop in order to at least pay for the plate, and fortunately by the time I got there it was waiting for me packed for travel. Although, I have no doubt that he would have even delivered the plate to me at the airport should it have been later!

DW: Please say a few more words about what your business does.

PA: For the past several years, I have undertaken projects related to high-end audio/video along with a huge dollop of automation and controls, in both residential and commercial construction. This is transitioning at this point as all of these are transferring to IP, and so placing a heavier load on the network infrastructure.

So we have some interesting work coming up, I hope, in teleconferencing, digital home healthcare, and the Internet of Things (IoT). We continue to emphasize heterogeneous solutions following the Unix philosophy, although I must say that the "new Microsoft", with its support of open source solutions and the willingness to make its own offerings interoperable with them, also has great appeal in our framework. So, for instance, we use OpenWrt, pfSense, FreeNAS, rsync, etc., even if our clients are completely unaware that these are the components which are driving the solution.

Ultimately, I try my best to embrace each and every opportunity as it comes.

DW: Let's return to the question of when you first (truly) got involved with TeX and seriously interested in typography?

PA: One of my career stops was at the Royal Ontario Museum in their nascent IT department where my manager, Mark Dornfeld, was the one who inspired me to take typesetting seriously. He is an archaeologist by training who fell into Unix. He brought SCO Xenix into the ROM, and then SoftQuad's troff. His vision was to leverage the multi-user capabilities of Unix to develop collaborative workflows to guide exhibit documentation, and project management schedules.

At some point later in my self-employment phase, I wanted to bring a typesetting structure to my documentation, and so I downloaded MiKTeX onto my Windows laptop, at which point I started to use it for general document creation. It took until my desire to create a web resource to aid young people struggling with mathematics to really embrace TeX as a way of life. I presented that work at TUG 2010 in San Francisco: Using LaTeX to generate dynamic mathematics worksheets for the web[1]. That paper is dedicated to my mathematician grandfather, Prof. Bansi Lal, who is renowned for his textbooks in the areas of geometry, algebra, and calculus.

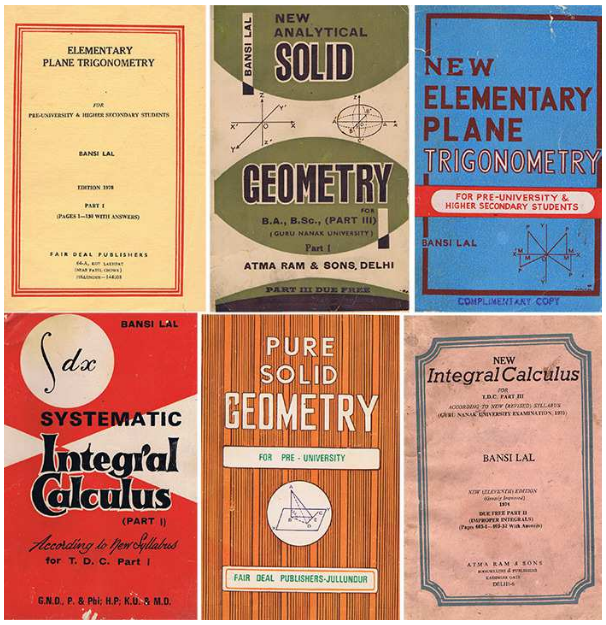
During my project's development, I had flashbacks of my childhood visiting my grandfather's printing press in Jullunder, Punjab, where all of that math was typeset traditionally in a dusty, noisy

Prof. Bansi Lal was honoured at his university with a student apartment block named for him.

press room, and where technicians would use tweezers to pull characters from letterboxes to put together pages of complex mathematics. I still am left trying to figure out from where he might have ordered the metal type for his books. I only wish I had had a chance to talk with him about the world that was all consuming to him.

I can only dream of what would have been some very amazing conversations. Isn't it the curse of the young to perpetually miss the opportunity to see the remarkable lives of their elders as their own older selves will one day see them. It fills me with great regret, but at the same time it connects me to that family history.

DW: Was your grandfather who wrote mathematics


ELEMENTARY PLANE TRIGONOMETRY
BANSI LAL
FAIR DEAL PUBLISHERS
NEW ANALYTICAL SOLID GEOMETRY
BANSI LAL
ATMA RAM & SONS, DELHI
NEW ELEMENTARY PLANE TRIGONOMETRY
FOR PRE-UNIVERSITY & HIGHER SECONDARY STUDENTS
BANSI LAL
COMPLIMENTARY COPY
BANSI LAL
SYSTEMATIC Integral Calculus
(PART I)
According to New Syllabus
for T. D. C. Part I
PURE SOLID GEOMETRY
FOR PRE - UNIVERSITY
FAIR DEAL PUBLISHERS-JULLUNDUR
NEW Integral Calculus
BANSI LAL
ATMA RAM & SONS

Covers of books that Prof. Bansi Lal wrote and printed.

books the same person as your grandfather who printed books?

PA: He was one and the same. He authored the books, but was dissatisfied with the contracted output. So he ended up buying a printing press, and taking over the process end-to-end. Doesn't that sound similar to what led DEK to the development of TeX? I have vivid memories of my grandfather coming home, while we were visiting him, and with a dip pen and different colours of fountain pen ink marking up proofs in the evenings, and then going back to the press to have the corrections made.

DW: The work you presented in San Francisco was done in LaTeX. Is that still what you use, e.g., for the work described in the following papers:

- YAWN — A TeX-enabled workflow for project estimation[2]
- TANSU — A workflow for cabinet layout[3]
- SUTRA — A workflow for documenting signals[4]

PA: Since then, I have been using mainly ConTeXt to develop what I like to term as "specification-driven documentation". That is, the specification is in a form other than TeX and its dialects, with an emphasis on YAML, along with scripts to generate the desired output on demand.

DW: Please say another word or two about YAML for those of us who don't know what it is.

PA: YAML is a simple markup language that has a close association with basic data structures such as hashes, arrays, and strings. Support for it is available in nearly all modern languages — Ruby, Java, C#, etc. I was looking for a structured representation for my specifications that was human readable. The hope was that in the field, I should need at most Vim to create and edit specifications, and even pen and paper would suffice should it come to that. I found XML too unwieldy, and preferred the minimalism of YAML. The specification creation is often done with little time available, but I was still seeking a consistent representation that could capture information.

Here's a small excerpt (from one of my articles) from a YAML specification, for a cabinet:

```
:cabinets:
  :subcategory: Cabinets
  :items:
    -
      :model: BD24
      :width: 24"
      :height: 30 1/4"
      :depth: 23 5/8"
...
```

It is a plain text file, like TEX source, but as can be seen, structured and machine-processable.

DW: Why do you prefer ConTEXt to LATEX for what you are doing with YAML, and in general? Can you give a couple of examples of aspects of ConTEXt that particularly appeal to you?

PA: I met Hans Hagen, the creator of ConTEXt, in San Francisco and was impressed with the structured nature of ConTEXt. It fit the way that I was thinking about page elements as well as documents.

One specific example is the rich set of attributes with which one can typeset tables in ConTEXt. Following the previous example, here is how a table of such information about cabinets can look:

Base Cabinets

Model No.	Description	Price
HD30844D	30" x 84" tall cabinet with 4 drawers	$1,192.89
B2D24	2-pot drawer 24"W base cabinet	$362.98
S24	24"W base cabinet opening	$250.64
BSD30	24"W sink cabinet with drawer face	$344.77
HD1584	15" x 84" tall cabinet	$501.75
	Sub-total	$2,653.03

In another area of construction, a great need in electrical work is the proper labelling of panels. They often have a combination of single-pole, double-pole, and tandem breakers, and with this setup, horizontal alignment of the breakers with the panel slots isn't regular, which places some heavy constraints on vertical cell merging when typesetting. So I have used ConTEXt's natural tables to typeset panel documentation for line voltage, alarm sensor, and data network port panels.

In one instance, I had put together some rather detailed panel documentation for a complex alarm system, and when that system went haywire in the middle of the night (naturally) due to a failed water sensor, the client was able to diagnose the problem with the aid of those panel diagrams. I imagine that had they been forced to trace back wire labels through a mess of wire bundles, that their experience may have been less than satisfactory.

It does put a smile on my face to think that somewhere out there, electricians are referring to documents created using GUST Type Foundry fonts and ConTEXt!

DW: Do you keep up with the latest versions of ConTEXt (`context-minimals`)?

PA: I was fortunate to start with ConTEXt MkIV, and so that is where I have stayed. As such I wasn't forced to undergo a migration across major versions. My needs don't change very often, although ConTEXt does. I update only occasionally. I have a set of core templates that have evolved with my projects, and I try to stick to the capabilities within.

My ConTEXt literacy is still rather low, I feel, and so I rely heavily on the assistance of those on the ConTEXt mailing list to help me. Hans has encouraged me to do a deeper dive into LuaTEX, and I hope to carve out some serious time to do that. I have some ideas about projects, but don't have enough mastery of ConTEXt to have the confidence to undertake them quite yet.

DW: Do you remain happy with your choice of ConTEXt? I ask this because sometimes I have made choices that I later regretted, but so much of what I do depends on what I am doing by then that I cannot change to something that would have been better.

PA: This is an excellent point. An entrenched path can become the default choice after a while even as one begrudges that choice. In my case, however, I continue to be very happy with my committing to ConTEXt. Even with my limited skill it has allowed me to solve the problems that I am trying to solve.

DW: Please tell me how you came to be "officially" involved with TUG.

PA: I hadn't thought about it at all really. It was suggested to me that my use of TEX in industrial applications, and I guess my business experience might be of use to the organization. From my end, I am extremely grateful for all of the community contributions that make up TEX, and so wanted to offer up whatever experience that I had to the furthering of that community.

DW: What do you see as the value of the TEX community and TUG?

PA: I like that the TEX community is peer-to-peer with little hierarchy. Ideas and contributions flow freely across the entire community, and that too its reach is international. I see TUG as a facilitator for the information flow from and to its members, while also documenting and communicating notable developments.

DW: Your use of TEX, or rather ConTEXt, seems a bit out of the ordinary. In any case, what is your motivation for writing-up and presenting/publishing what you do with TEX?

PA: Personally, I don't find it that out of the ordinary. My interest is in workflows, and so the typesetting forms only one part, albeit a crucial one, of that. When large data-base management systems ruled, one was forced to learn different "report writers". These were very often proprietary in nature and offered only crude formatting functionality — really little more than grouping and aggregation. As I mentioned at TUG Boston, fitting engineering problems into a Model-View-Controller framework makes it natural to see TEX as a good fit for generating sophisticated (and beautiful) views.

DW: Please tell me about your aims in organizing TUG 2016.

PA: For TUG'16, I wanted to emphasize the contributions of local artisans in the fields of typesetting, printing as well as publishing, even if they weren't directly TEX related. It was my hope, ultimately realized even though I felt that it was a bit of a long shot, that we might also be able to attract some special speakers to provide a wider view about typesetting beyond TEX.

Over the years I have become less timid about reaching out to people, something that shocks those who knew the younger, painfully shy me. So I spent a great deal of time phoning and writing to people in order to make them aware of just what a unique group of individuals attend TUG conferences, and how wonderful an opportunity it is to come and meet them, and to participate in the discussions.

DW: What do you foresee in the future — with TEX and TUG and with your life more generally?

PA: Without trying to be flippant, I do sometimes think back to my reading of the words that J.D. Salinger gave to Esmé when she addressed Sergeant X in *For Esmé with Love and Squalor*. Taking liberty with the text: I hope to get through with my faculties intact. That, to me, seems a not unworthy goal.

My children are my counterbalance: my foibles provide them with an endless source of amusement. Just in the past year, I have gotten on a small dirt bike because of my son, and had discussions with him about the beauty of the twelve-string guitar: he is an excellent guitar player. My daughter continues to amaze me with her fierce independence. Recently, she and I have had some wonderful talks about the notes composition of artisanal scents, and the "noses" who crafted them.

I have a number of TEX-related project ideas, and hope to present the results at upcoming TUG conferences. If I have a wish, professionally speaking, it would be that I could have an opportunity to make meaningful contributions in the IoT space. That would allow me to bring to bear all of my experience in automation and controls, combining it with my background in AI to leverage the machine learning capabilities now accessible through cloud services. At TUG Darmstadt, I talked about ToT or TEX of Things so this would have a TEX tie-in as well.

DW: Thank you very much for taking the time to participate in this interview. I look forward to seeing you at TUG 2016 in Toronto.

[Interview completed 2016-05-24]

Links

[1] `http://tug.org/TUGboat/tb31-2/tb98arora.pdf`
[2] `http://tug.org/TUGboat/tb33-2/tb104arora.pdf`
[3] `http://tug.org/TUGboat/tb34-3/tb108arora.pdf`
[4] `http://tug.org/TUGboat/tb35-2/tb110arora.pdf`

◇ David Walden
`http://tug.org/interviews`

Type in the Toronto subway

Joe Clark

Here's Nina Bunjevac's artwork at the Art Gallery of Ontario, entitled "The Observer: The Ascent, Dundas Subway, Sunny Days." It's one of two images on walls in front of you and behind you when you stand in the gallery.

DUNDAS
DUNDAS

That really is what the type on the walls of Dundas subway station looks like. (It's Toronto's second-busiest station.) What Bunjevac has given us is a hand-drawn facsimile of the typeface that's really on the walls, Univers.

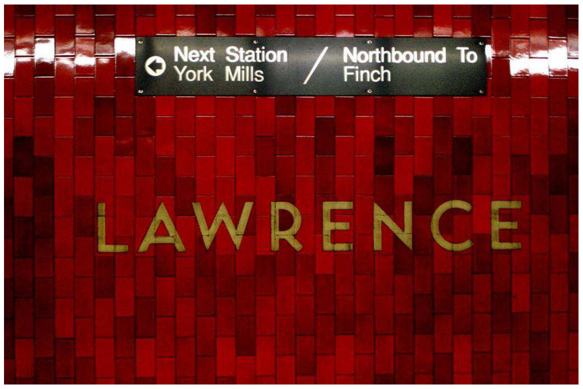
But type in the Toronto subway is much more than Univers on one station wall.

1 Fundamentals

The subway in Toronto is run by the TTC, the Toronto Transit Commission.

The story of type in the Toronto subway is a story about:

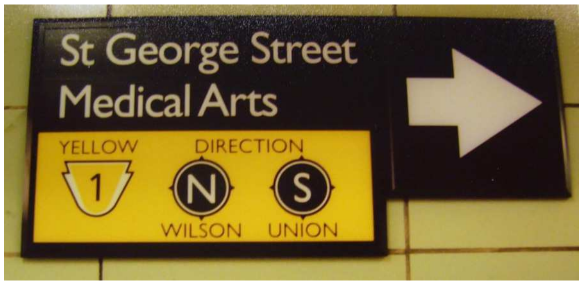
1. A 50-year-old custom font of almost unknown origin.



Station sign LAWRENCE in TTC typeface, with Helvetica sign added above

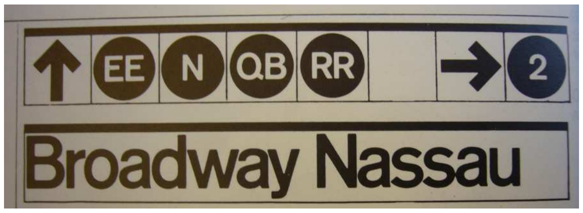
2. A subway lined with washroom tiles.

3. A system that hired a wayfinding expert, paid him to install and test a new signage system, then ignored that new system after it tested better than the old one.



Black sign with white Gill Sans lettering and pictographs

4. A billion-dollar corporation that cloned Massimo Vignelli's work for the New York subway from 40 years ago — but won't admit it.



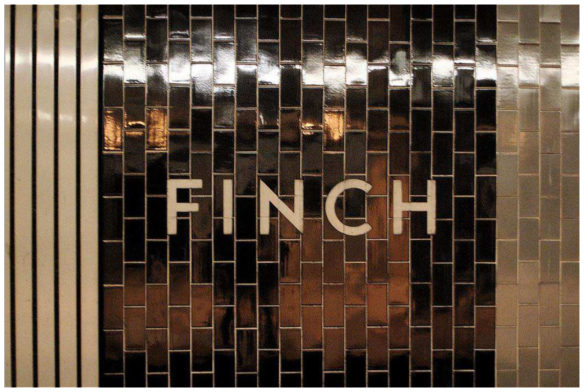
5. A billion-dollar corporation that refuses to test its signage.
6. A billion-dollar corporation that uses as its main font a Helvetica clone that came free with CorelDraw.

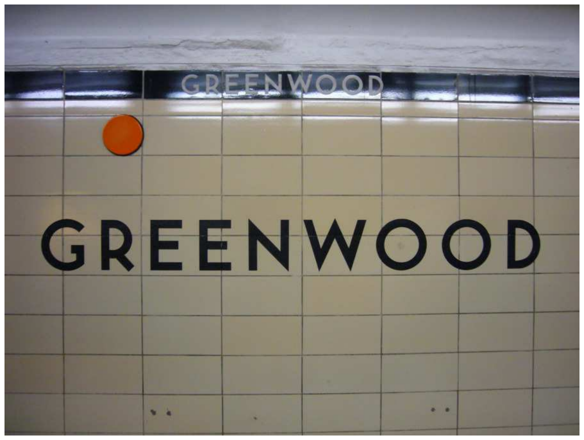
This is the story of a unique typographic heritage that the TTC is *totally blowing.*

2 The subway

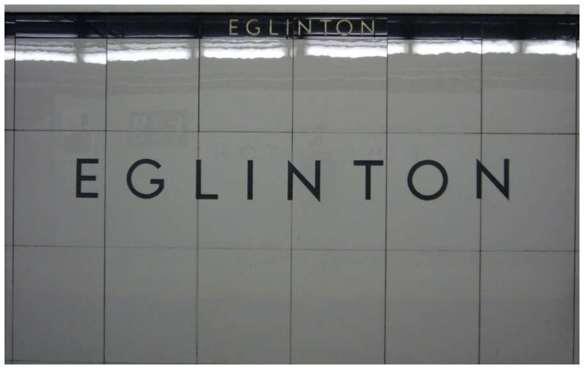
Toronto's subway consists of 69 stations on four lines. The subway opened in 1954, and from the very start we've had a unique font on the walls.

The TTC's custom subway font is usually sandblasted into the walls.



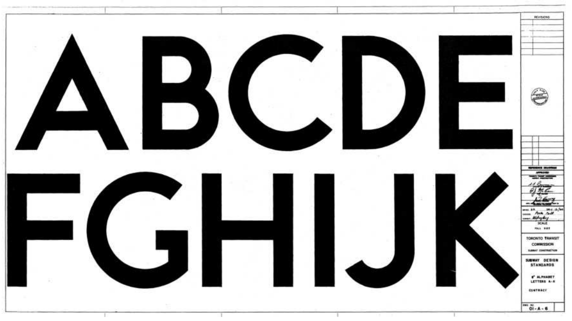


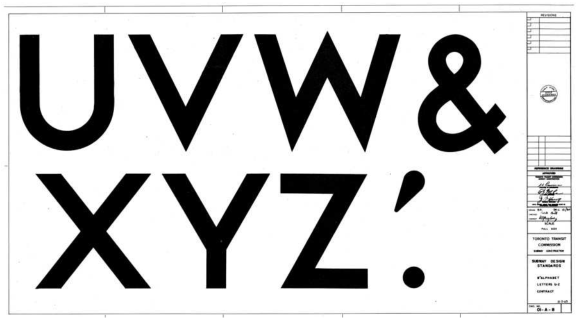
The walls themselves are interesting. In nearly all cases, the walls are finished in tiles. Originally we used glossy large-format Vitrolite tiles, then different kinds of tiles later.

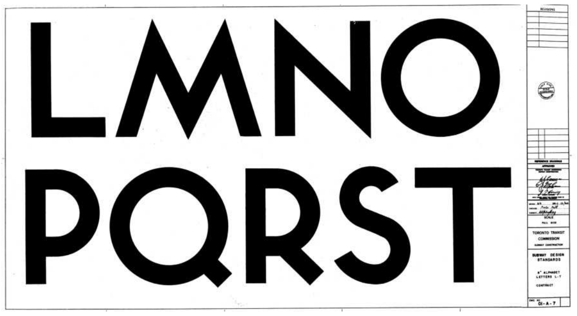


EGLINTON letters embossed into large glossy grey tiles

The typeface itself is a geometric sansserif, upper case only, with some unusual features:





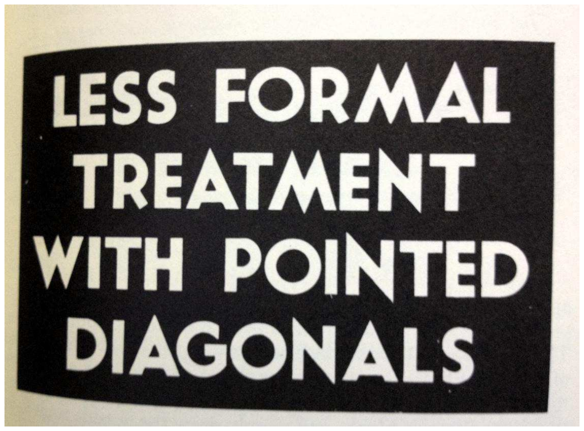


1. Low waist of the R.
2. Points of A V N W M that extend past the baseline or cap height.
3. What we'd consider nowadays to be quite a heavy weight for signage, though you also find some rare usages of a light weight (as seen in the Eglinton example above) for which we don't have any drawings.

The font doesn't have a name and nobody knows who designed it. What I have shown above are believed to be the original drawings for the typeface, but they're dated 1960, six years after the subway opened.

But we now have a few clues about the origins of the font.

1. First, a little birdie found this very similar face in an old book of type specimens:



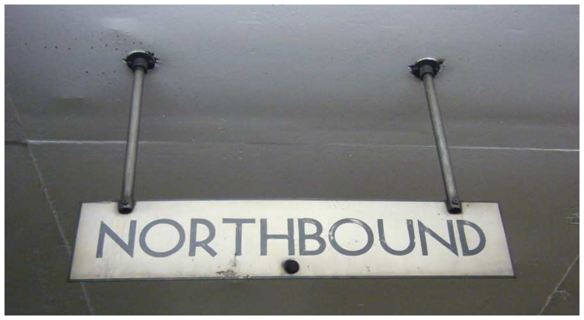
That's maybe halfway to the TTC font. Typefaces similar to each other have been designed more or less simultaneously even though the designers in question had no knowledge of each other. Helvetica and Univers were a case of that. So this general type family might just have been in the air in the 1950s.

2. Next we've got a bit of skullduggery from an online forum. Someone named simply Brent looked more carefully than I ever did at the signatures on some of the old TTC drawings. And Brent says:

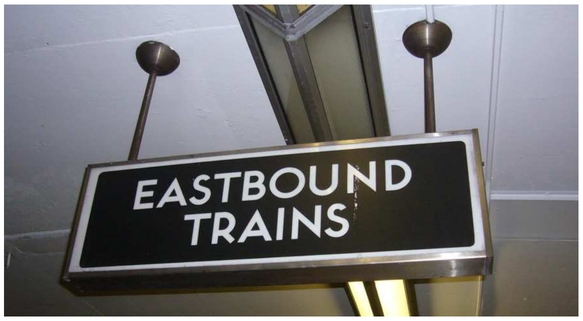
> The drawing for the 4″ standard alphabet indicates that it was drawn by a P. Butt, and reviewed/checked by a W.F.G. Godfrey…
>
> A little digging leads to William Frederick George Godfrey (b. London, England, 1884; d. Toronto, 1971). He was a Toronto artist who did engravings and other line drawings, but he was originally trained as an architect.
>
> I am guessing that Godfrey was the designer, and that Butt was the draftsman.

There are different kinds of signfaces that use the TTC font, not just sandblasted letters.

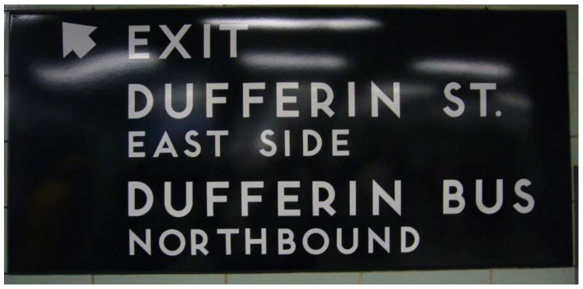
1. Very early white signs with black letters.

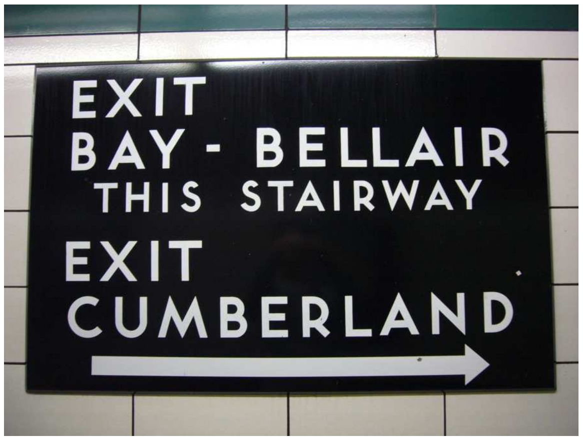


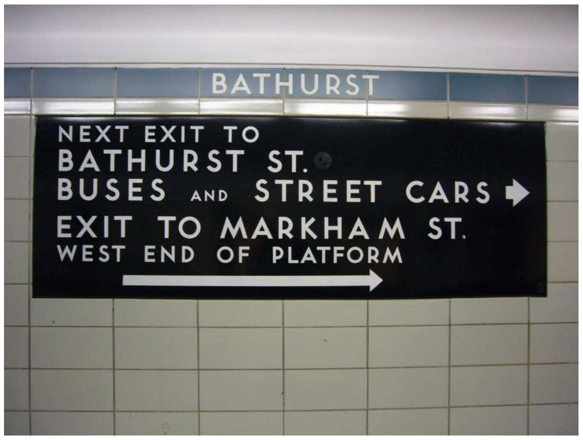
2. Backlit box signs with white type on black.



3. And the most cherished of all, massive enamelled-steel plates that have lasted almost without a blemish for 40 years or longer.







These signs have never really been tested, but they *appear* to be mostly functional.

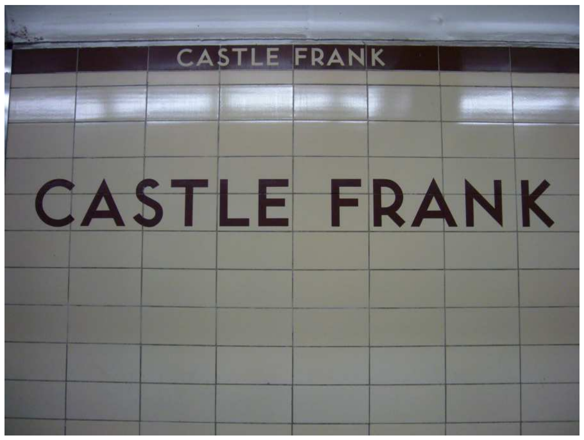
Nonetheless, the TTC is run by jumped-up motormen and engineers and old guys who think anything related to "print" or "design" is girly and decorative.

You know what these people are like. They think design is the icing on the cake. They don't know that design is the *recipe for* the cake. They don't know that the cake *is* design.

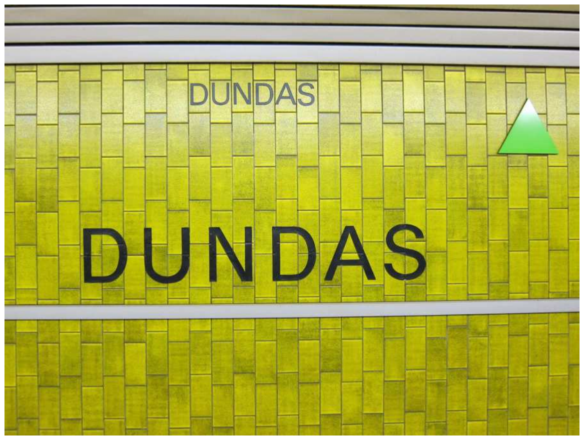
At any rate, starting in the 1970s, the TTC began to pollute its nice tidy uniform design.

They extended the first subway line north and also south around a loop at the southern point of downtown.

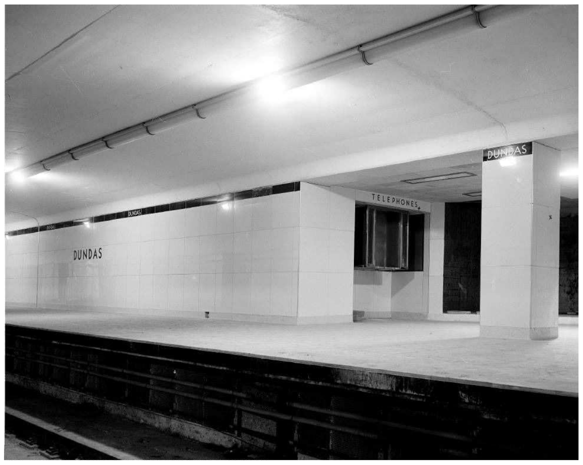
They opened a crosstown line with the original fonts.

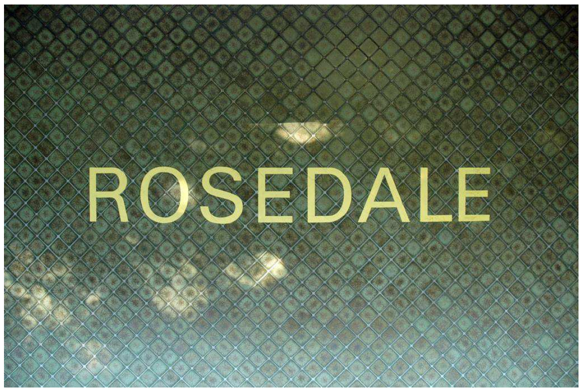


They renovated some of the *original* subway stations. They destroyed the original Vitrolite tiles in all but one of them and replaced them with haphazard tiles and haphazard fonts. I've got bad and good examples here.

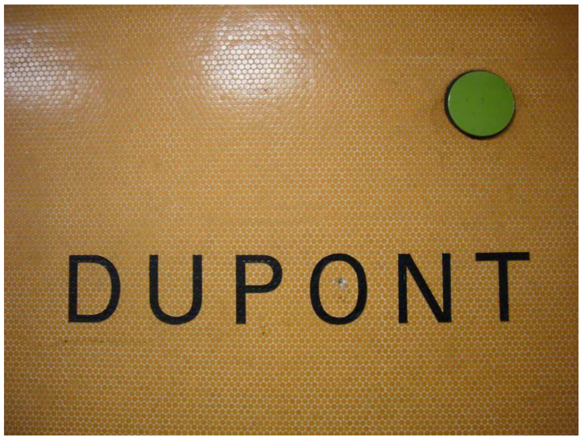


For reference, Dundas station originally looked like this (with pale yellow walls):

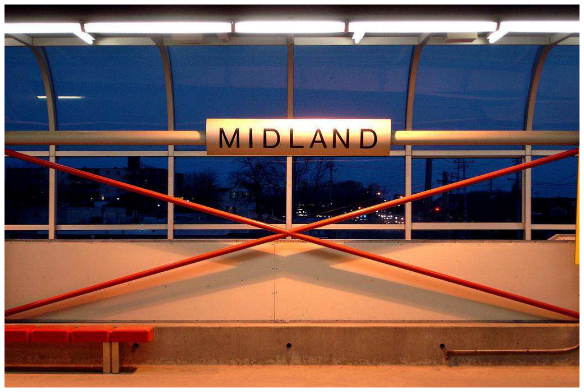




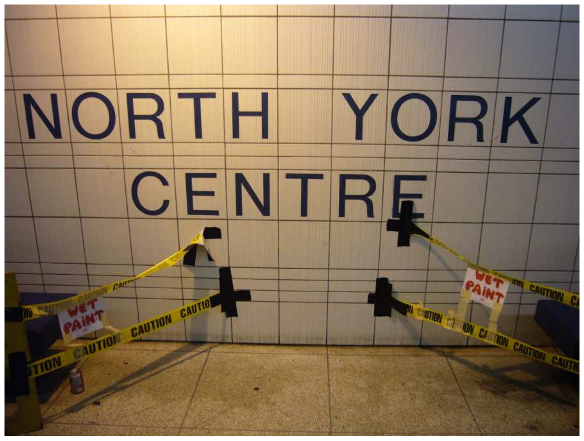
Then they extended the first line again, with each station ostensibly using nothing but Univers.



Then they opened a suburban line using toy trains. It uses signage in Helvetica on curved-metal blades.

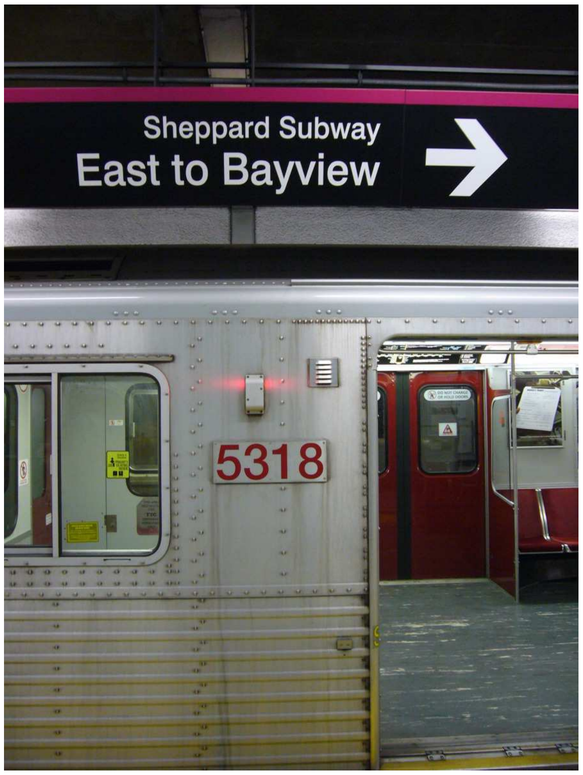


Then they opened a couple of extra stations here and there using Helvetica.



All the while, behind the scenes they were replacing signage with whatever they could get their hands on, mostly Helvetica.

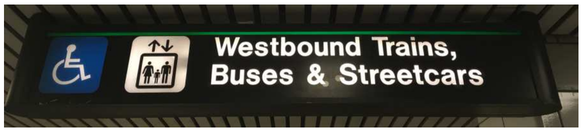
And finally they spent nearly a billion bucks on a new *five-station* subway line to nowhere using fake Helvetica — or fake Helvetica, as we shall soon see.



What we've got now is a completely unplanned mixture of signs in the true TTC typeface, and signs in Helvetica, fake Helvetica, Univers . . . and Arial.

Now, it's really hard to get this point across to the TTC, but when your signage is all hither and yon like this then (a) people get lost, especially tourists and people who haven't learned the system the hard way, and (b) your entire subway system looks *undesigned* and people are encouraged not to believe a word your signs say.

And in some cases that's literally true:

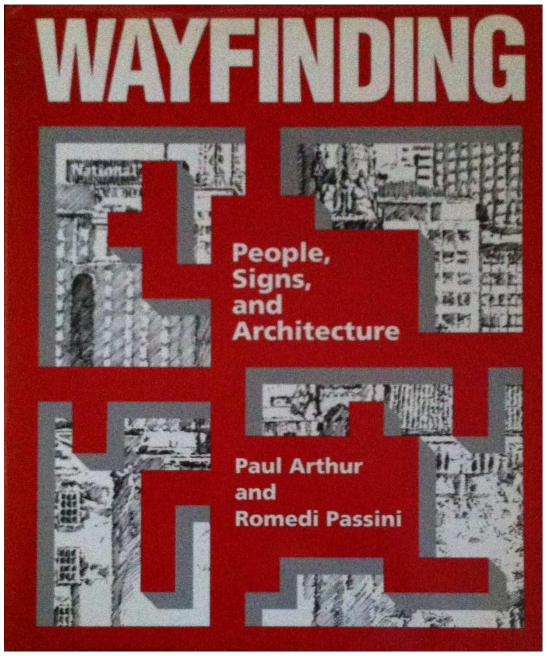


The only westbound vehicles from this station (Bathurst) are trains. Buses and streetcars don't travel west. This is a sign that lies to you.

3 But, along the way, they *did* try to fix it

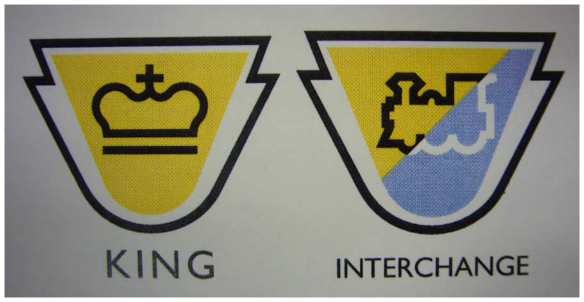
In the early 1990s, the TTC hired Paul Arthur to develop a new signage system. Paul Arthur, a British-born Canadian graphic designer, died in 2001. He left behind a substantial legacy that is unknown to designers outside Canada.

1. Paul Arthur was a pioneer of signage and wayfinding. He designed the pictographs at Expo '67 in Montreal, widely seen as the first high-profile use of pictographs in a public setting.
2. He cowrote a couple of books, the most important of which is *Wayfinding: People, Signs and Architecture* (McGraw-Hill, 1992, reprinted 2005 by Focus Strategic Communications; `amazon.com/dp/0075510162`).


WAYFINDING
People, Signs, and Architecture
Paul Arthur and Romedi Passini

TTC spent about a quarter of a million dollars coming up with new designs with Paul Arthur at the helm.

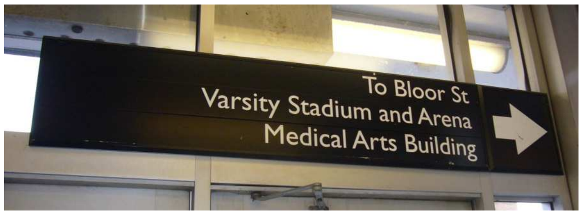
Lance Wyman helped out. (You may know him from Mexico City Olympic signage.) For the TTC, Lance Wyman drew most of a set of new pictographs for subway stations.



Paul Arthur's project remade one half of one station, St. George, an interchange between two lines. The entire east end of the station, on all levels, was made over with the new Paul Arthur signs, while the west end was left intact.

Some of the features of the Paul Arthur system:

1. He used Gill Sans.



   Paul Arthur was English and this was really a holdover from his childhood. He considered all sansserifs to be equally legible, which obviously they *are not*.

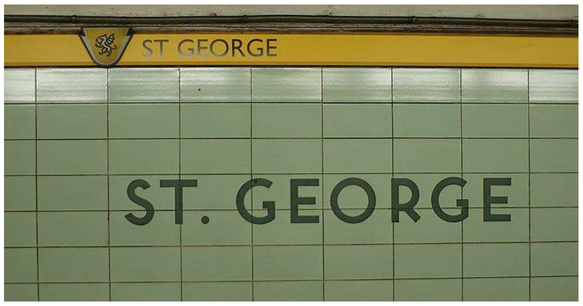
   Gill Sans in this case was too light a weight for signage, though they did expand the tracking.

   As ever, there is the notorious difficulty of distinguishing *I*, *l*, and *1* in Gill Sans. Some of Paul Arthur's drawings show the straight-line *1*, others the real numeral *1*.

2. Subway lines would no longer have names, which admittedly are ridiculous in Toronto. They tend to relate to the streets under which the subway runs, which themselves aren't accurate. We've got the Yonge–University–Spadina line (yes, three names for one line), the Bloor–Danforth line, and the Sheppard line. The Scarborough RT runs through the neighbourhood, and former city, named Scarborough, and RT means "rapid transit." The nomenclature is a mess.

   In Arthur's new system, lines would each get a colour and a number. And the colour would be written out in words to be accessible to colourblind people.

3. Every station had a strapline above the tracks on the train-wall side in the line colour, with the name written out and the station's custom pictograph. In principle, even if you couldn't read you could at least find your station.



Paul Arthur tested the St. George prototype with four groups — the "general population," meaning riders without disabilities who could read English; the visually impaired; a "multicultural" group, that is, English-as-a-second-language speakers; and an English-speaking group "with a low level of literacy," who were often students.

1. The low-vision people hated all the signs, but they hated the new ones less, and all the other groups preferred the *new* signs.
2. This was just an opinion survey, not a test of tasks and performance. Nonetheless, the *new* signs were deemed better.

So the TTC ignored the results. Literally. It would have cost about $8 million to convert the whole subway to the new system, but the Toronto Transit Commission never voted on doing that. It was never brought to the elected commissioners. It was killed internally, and there are no records of how that happened.

And many of the Paul Arthur signs were simply left in place. They're still there a decade and a half later!

But, as of three years ago, TTC started phasing out line names in favour of numbers and colours.

1. Obviously the numbers are in Helvetica.
2. And just as obviously, this male-run organization picked a set of colours that colourblind people cannot necessarily tell apart. 4% to 8% of the male population, and some females, have colour deficiency, but over and over again TTC picks green, yellow, and orange as colours.

The point here is that, nearly 20 years on, the TTC finally adopted one of Paul Arthur's ideas, but, in true Toronto fashion, they half-assed it all the way.

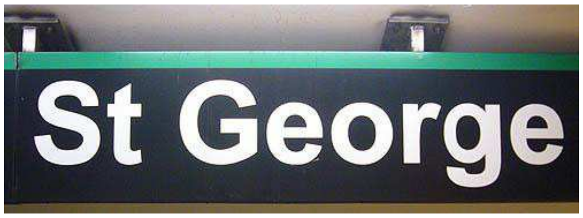
4 Then there was the Sheppard subway

TTC and the City of Toronto proposed an expansion of subway lines in the 1990s. The plan was to run two new lines across midtown Toronto on Eglinton Ave. West and on Sheppard Ave. East and West.

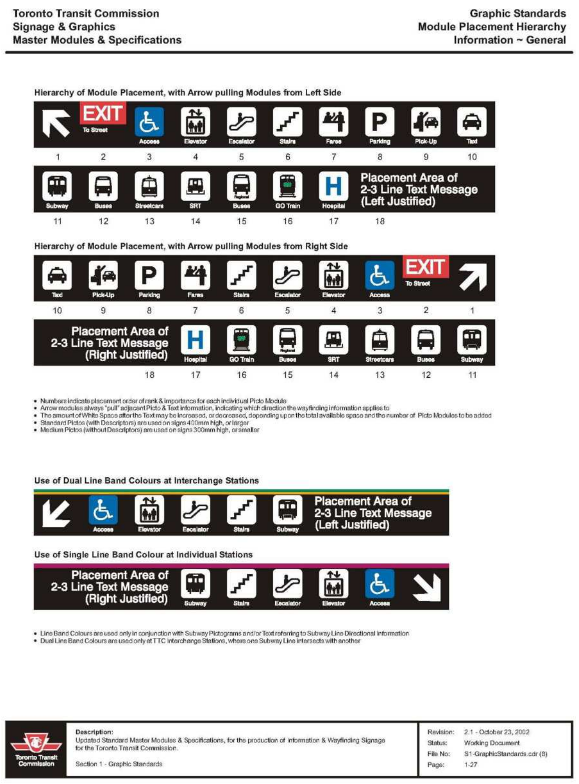
But a new provincial government was elected that hated Toronto. It tried to scotch the whole project. What we ended up with was five stations on Sheppard Ave. East that end in the middle of nowhere. And this *five*-station Sheppard line cost *$933 million* to build.

For a nice new subway line, you need nice new signs. So, guess what, the TTC ignored the Paul Arthur designs they'd already paid for and cooked something up in-house.

1. They threw together two prototype overhead signs and installed them — where else? — at St. George station. And of course they're still up today!
2. And the biggest type on those signs is set in ... Arial.
3. One of the signs could not even construct a lower-case *g* correctly.



4. So, to recap St. George station: It's got more than half of its original signs, or at least signs from the 1980s, plus many of the Paul Arthur prototype signs from the early '90s, plus the Sheppard prototype signs. Still. Today.
5. The TTC threw together these fake-Helvetica signs and ran them by a dozen people. That was their testing. And from that they wrote a 350-page instruction manual on how to clone Massimo Vignelli's designs for the New York City subway in the '60s.



Toronto Transit Commission
Signage & Graphics
Master Modules & Specifications

Graphic Standards
Module Placement Hierarchy
Information ~ General

Hierarchy of Module Placement, with Arrow pulling Modules from Left Side

EXIT
To Street
Access
Elevator
Escalator
Stairs
Fares
Parking
Pick-Up
Taxi
1 2 3 4 5 6 7 8 9 10
Subway
Buses
Streetcars
SRT
Regional Buses
GO Train
Hospital
Placement Area of 2-3 Line Text Message (Left Justified)
11 12 13 14 15 16 17 18

Hierarchy of Module Placement, with Arrow pulling Modules from Right Side

Taxi
Pick-Up
Parking
Fares
Stairs
Escalator
Elevator
Access
EXIT
To Street
10 9 8 7 6 5 4 3 2 1
Placement Area of 2-3 Line Text Message (Right Justified)
Hospital
GO Train
Regional Buses
SRT
Streetcars
Buses
Subway
18 17 16 15 14 13 12 11

- Numbers indicate placement order of rank & importance for each individual Picto Module
- Arrow modules always "pull" adjacent Picto & Text information, indicating which direction the wayfinding information applies to
- The amount of White Space after the Text may be increased, or decreased, depending upon the total available space and the number of Picto Modules to be added
- Standard Pictos (with Descriptors) are used on signs 400mm high, or larger
- Medium Pictos (without Descriptors) are used on signs 300mm high, or smaller

Use of Dual Line Band Colours at Interchange Stations

Use of Single Line Band Colour at Individual Stations

- Line Band Colours are used only in conjunction with Subway Pictograms and/or Text referring to Subway Line Directional information
- Dual Line Band Colours are used only at TTC interchange Stations, where one Subway Line intersects with another

Toronto Transit Commission

Description:
Updated Standard Master Modules & Specifications, for the production of information & Wayfinding Signage for the Toronto Transit Commission.

Section 1 - Graphic Standards

Revision:	2.1 - October 23, 2002
Status:	Working Document
File No:	S1-GraphicStandards.cdr (8)
Page:	1-27

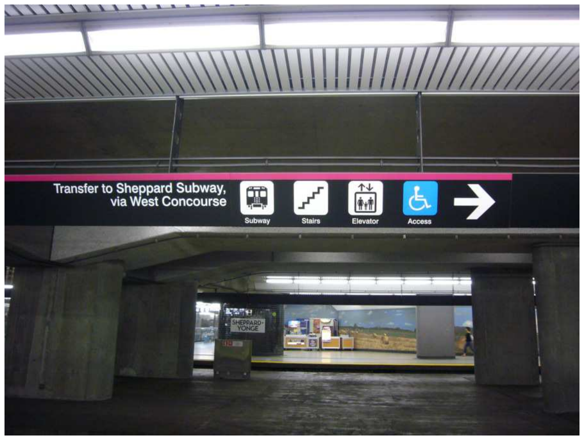
You see, Toronto has an inferiority complex. Still. Today. Deep down, we wish we were as good as New York. The fact that we're better than New York on a lot of scores means nothing. New York is the summit of a mountain we can never reach. But it also means that anything New York does is axiomatically the best.

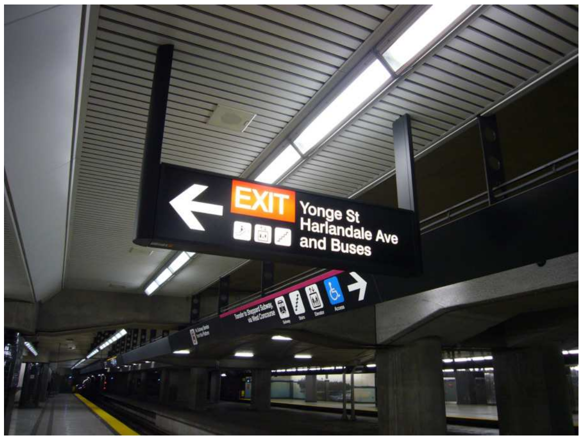
That further means the use of Helvetica for transit signage. Now, in the 1960s, Massimo Vignelli chose Helvetica because he's an arch-Modernist. Although of course the typeface he chose was really Standard or Akzidenz-Grotesk, because Helvetica wasn't available at the time in the formats he needed. (You should read Paul Shaw's book *Helvetica and the New York City Subway System*, `helveticasubway.com`.)

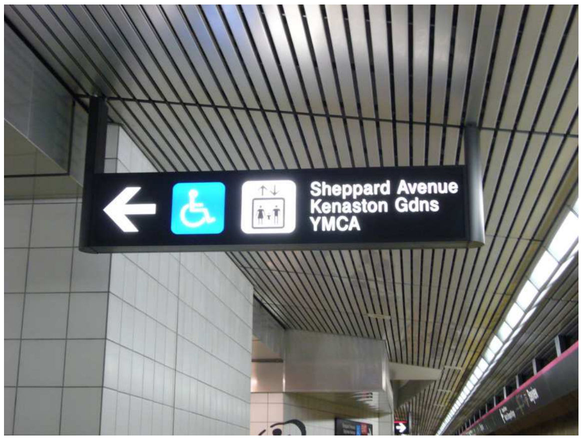
But anyway, we don't live in the 1960s. We have engineered signage fonts now, and we can design new engineered sign fonts if we need them, *and* we know more about testing.

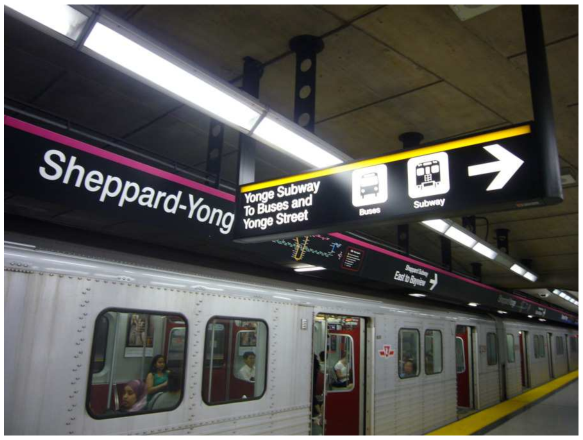
But TTC staff are visual illiterates and Windows users and they have no taste whatsoever. Their powers of analysis begin and end with "I can read it" and "It looks clean."

So what do we have in the Sheppard subway? Wall-to-wall Helvetica. And half the time it's backlit or electronically scrunched.





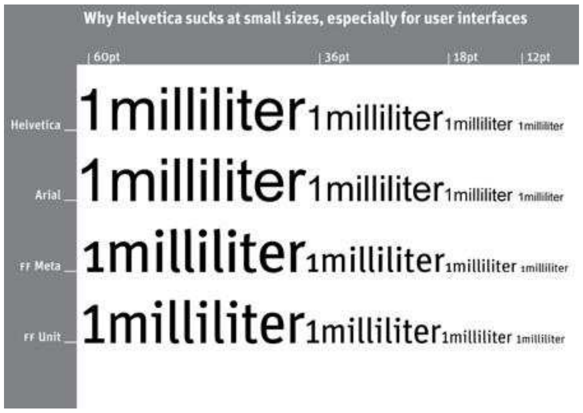




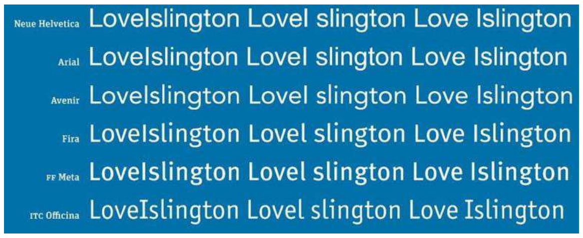
It looks pleasingly uniform compared to the mish-mash on the other lines, but is that enough? No, not for transit signage, because it has to *perform*.

Helvetica does not work well in signage. There are some reasons for that, which, while obvious, are not obvious enough to dissuade the TTC.

1. All the usual confusable characters, like *I*, *l*, and *1*, remain confusable. In fact, numerals are *really* confusable.
2. The whole thing sets too tight together by default. When you're at a distance from a piece of text, an optical phenomenon called crowding reduces the legibility of characters that are tightly spaced together. Helvetica sets too close together by default.
3. The now-retired type designer Erik Spiekermann has a couple of graphics that show these issues nicely:


Why Helvetica sucks at small sizes, especially for user interfaces
60pt
36pt
18pt
12pt
Helvetica
Arial
FF Meta
FF Unit
1milliliter

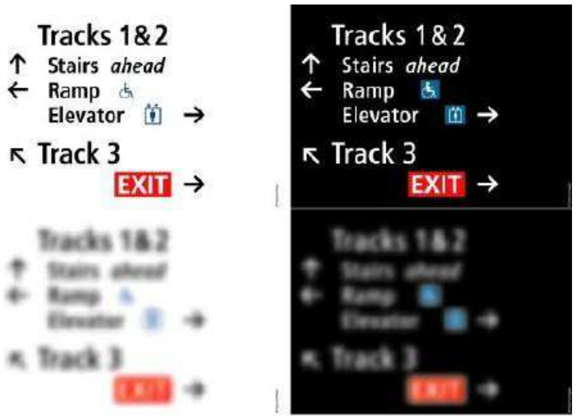
"1milliliter" in typefaces of differing legibility (Helvetica among the least legible)



"LoveIslington" in typefaces of differing legibility (Helvetica among the least legible)

Even if we didn't have evidence already that Helvetica is a lousy choice for signage, my business partner Marc Sullivan and I demonstrated it for another transit system here — GO Transit, the commuter-rail system.

We showed this other transit system a set of alternatives, in positive and negative, sharp and blurry, and proved that Helvetica doesn't work. So GO Transit went with something other than Helvetica. They went with the wrong font, but at least it wasn't Helvetica.



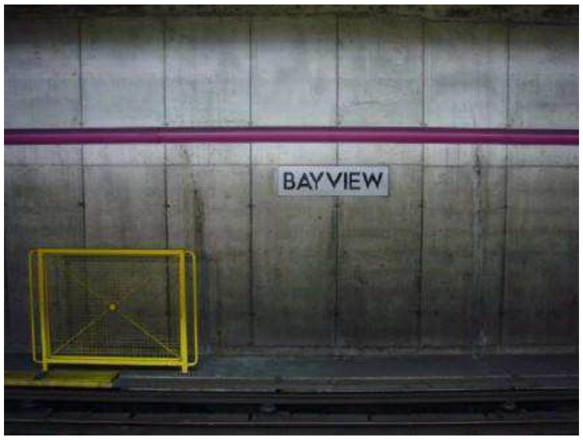
Four sign mockups, black on white, white on black, sharp, and blurry

And there's more: What the TTC is using isn't real Helvetica or Helvetica Neue. It is actually Swiss 721, the Bitstream clone that comes free with CorelDraw. If you've ever used an HP printer or any Corel software, you're familiar with Swiss 721 as Bitstream's Helvetica clone from the late 1980s. A very good copy, but it isn't the real thing. And in the best case one could argue with the propriety of a municipal transit agency's using a copy of a real font.

The most interesting point about the signs in the Sheppard subway is the fact that the man who developed them ultimately could not use his own signage.

Bob Brent was a TTC manager in charge at the time. Later, Brent had a hip operation that had him using a wheelchair and then a walker. And on two occasions, right there at Sheppard station, he could not find an accessible exit using his own signs.

The TTC did give us a little sop to the past in the Sheppard subway. The name of the station on the train-wall side uses the old TTC font. Except the font is too small and too tightly spaced. They couldn't even get that right, in other words.



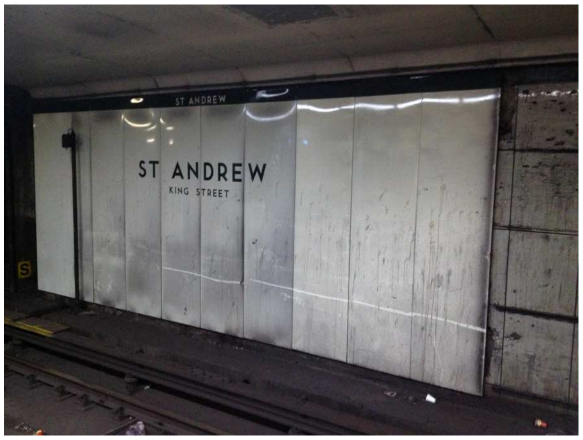
5 What's happening these days?

More than a few interesting things have happened in recent years.

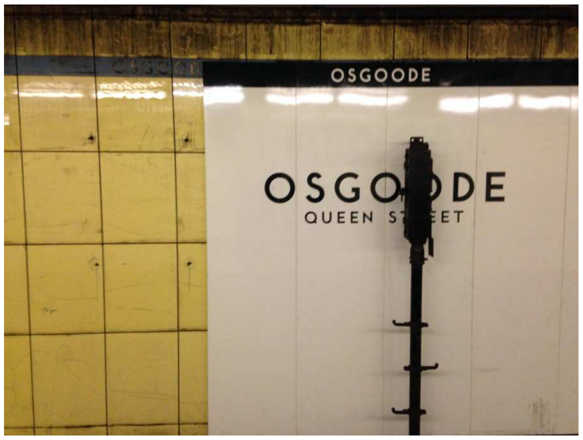
First, TTC now actually has a design department and I actually did some work for them. I set up a few tasks — common ones, unusual ones, and rare ones — and did a test to see if I could carry out those tasks using only existing signage. Usually I couldn't.

This design department does not inspire a lot of confidence. It spent almost a year being the world's only Helvetica truthers. TTC Design literally told people — in writing, on Twitter, in their own design manuals — that Swiss 721 is the real typeface and Helvetica is the clone. TTC Design wanted us to believe that those Swiss designers produced a typeface called Neue Haas Grotesk and then one called Swiss 721 and then, finally, a font called Helvetica. That didn't happen, and the Germans didn't win World War II, either.

Also, one of the TTC's junior architects took it upon himself to solve the problem of new wall coverings. They did a test at one downtown station, St. Andrew.



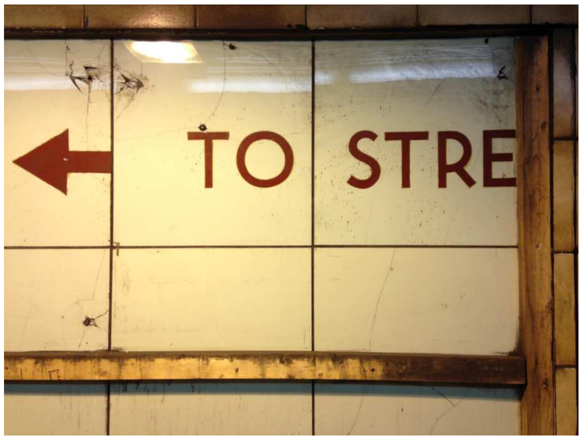
The first couple of installations didn't work, but they kept at it, and now we have a nice new material that looks a lot like the old Vitrolite tile, though the slabs are full height floor-to-ceiling. And in those installations, they're duplicating the original typography *almost* exactly. All that is *pretty good.*

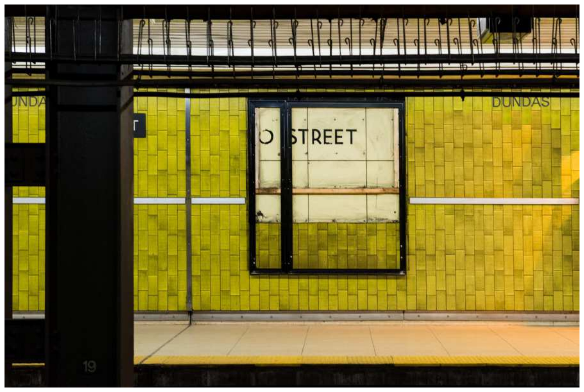


Panoramic fisheye view of original Vitrolite tiles at Osgoode station

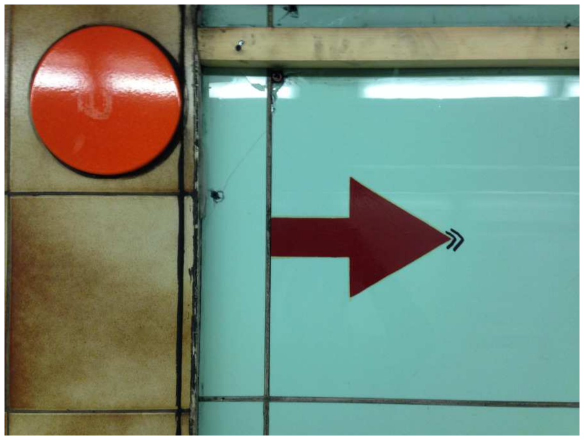
Panoramic fisheye view of new wall cladding at Osgoode station

And one more thing that's going on is digging out subway tiles and, apparently, putting them right back up again, albeit with different and bigger caissons, as they're called, for advertisements. But while the tiles are removed, we've all been able to see the true aboriginal TTC type that's behind them, on the *original* tiles, hidden for many many years.

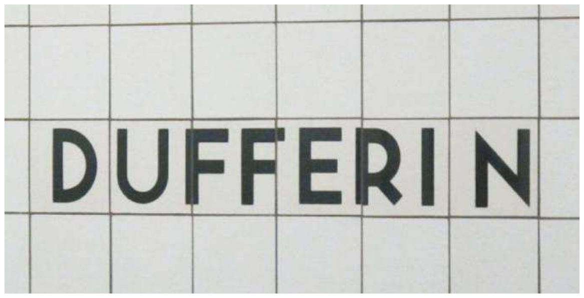




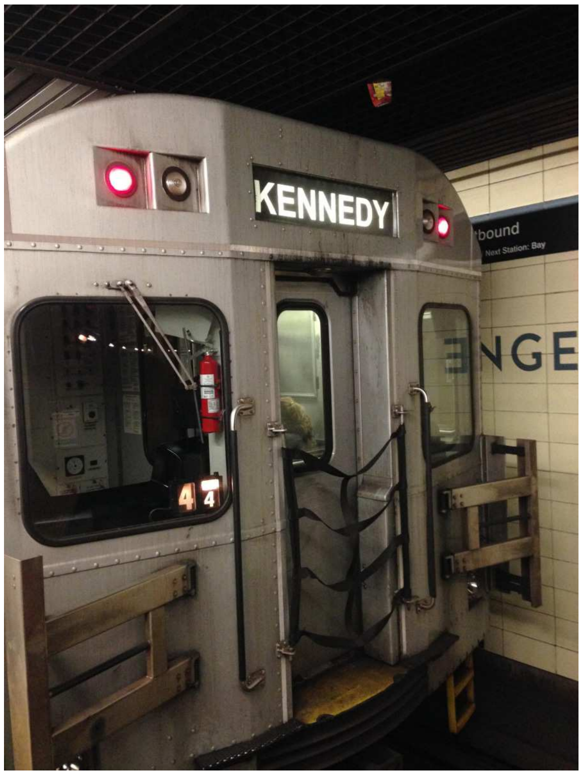
Ad frames at Dundas station reveal pale yellow square Vitrolite tiles, O STREET legend in light TTC typeface



And the TTC is renovating subway stations. They use their own in-house version of the TTC font and you can easily tell the difference. Plus there was that time they installed the letter *N* backwards at Dufferin station. That even made the papers. (It was fixed soon enough.)



Even many years after they should have known better and even after they hired me and well past any point where there was possibly any excuse, exactly a year ago some miscreant unknown gave the order to tear out the destination signs on subway trains and re-"typeset" them in Arial.



This is for real. Before this, the destination signs were set in Futura and some mid-century industrial gothics. Much better, and more historically accurate, and not the total abomination that Arial is.

And, while the TTC has its own in-house outline version of the subway typeface, Toronto designer David Vereschagin produced and sells his own version, entitled Toronto Subway (`www.quadrat.com/ts.html`).

6 Conclusion

Type in the Toronto subway is a story of just how much of a mess you can make without adult supervision. The TTC started out with something nobody else had, and then, through a combination of ignorance and bad taste, spent 50 years tinkering with it and diluting it.

Photo credits

Some photos by David Topping, Luke Tymowski, Craig James White, Lori Whelan, Bruce Reeve, Nathan Ng, and Michael Leland, all used by permission or under Creative Commons. Photo of Nina Bunjevac artwork by the author; reproduction of artwork by permission of the artist. Photo of 1954 Dundas Station courtesy Toronto Archives (Series 381).

◇ Joe Clark
`joeclark.org/tug`

Are justification and hyphenation good or bad for the reader? First results

Leila Akhmadeeva, Rinat Gizatullin and Boris Veytsman

1 Introduction

Since the early days of typography text justification was considered a necessary feature of well-typeset text. The type block should be rectangular and grey. Even the “New Typography” by Jan Tschichold [5], while rejecting many dogmas of classical typesetting, still retained text justification and, as a consequence, hyphenated text.

While hyphenated text has obvious aesthetic advantages, typography is not just about aesthetics: its main purpose is to help the author to convey her thoughts to the readers. Is hyphenated and justified text really better for the reading and comprehension?

In this work we try to answer this question.

2 Experimental methods

The experimental methods were a variation of the scheme used in our previous papers [1, 2, 7]. A group of $N = 300$ healthy volunteers (Bashkir State Medical University undergraduates) was given two texts, A and B. Each text had 282 words, typeset with LaTeX using ParaType Serif fonts. Half of the participants got text A justified and text B ragged right, while the other half got text B justified and text A ragged right. The participants were asked to read the text. After a minute they marked their current reading position. Immediately after the reading the participants were given a multiple choice test (10 questions with 4 variants of answers to choose from). To test the long-term memory, we repeated the test 60 minutes later. We compared the differences between the justified and ragged right tests.

3 Results

The difference between the results for justified and ragged right texts for the same subjects is shown in Figures 1 and 2. The results suggest that justified texts give slightly higher reading speed and slightly fewer correct answers on the delayed test (note that Figure 2 shows the *difference* between justified and ragged texts). However, the effect is small: we will see below that it is smaller than the difference between the texts A and B themselves and the individual differences between the subjects. To quantify the effect we use a Bayesian technique [3, 4].

In our model we assume that each participant has an individual reading speed and individual probability of correctly answering a question. Besides these individual propensities, there are corrections for the texts (A or B) and typesetting (ragged or justified), common for all participants. These corrections are what we want to determine.

More formally, let us introduce the parameters

$$\delta_a = \begin{cases} 1, & \text{Text } A \\ -1, & \text{Text } B \end{cases} \qquad \delta_j = \begin{cases} 1, & \text{Justified text} \\ -1, & \text{Ragged text} \end{cases} \tag{1}$$

Then we will model the reading speed v as a normal distribution with the mean

$$v = v_{\text{ind}} + \frac{1}{2}(\delta_a v_a + \delta_j v_j) \tag{2}$$

and standard deviation σ. Here v_{ind} is the individual reading speed, v_a is the difference between text A and text B, and v_j is the difference between justified and ragged texts. We need to estimate v_a and v_j.

To estimate the probability of a correct answer on a test (either immediate or delayed) we use the log odds function [6]:

$$L = \ln\left(\frac{p}{1-p}\right) \tag{3}$$

where p is the probability of correctly answering. When p changes between 0 and 1, L changes between $-\infty$ and $+\infty$.

Then we can write down the mean log odds as

$$L = L_{\text{ind}} + \frac{1}{2}(\delta_a L_a + \delta_j L_j) \tag{4}$$

where the parameters have the same meaning as in equation (2). We use separate estimates for immediate and delayed test.

The *a priori* distribution for the parameters is the following:

1. Normal for v_{ind} and L_{ind} with mean from the data and deviation equal to 100 times the data deviation.
2. Normal for v_a, v_j, L_a and L_j with zero means and deviation equal to 100 times the data deviation.
3. Uniform for all standard deviations from 1/1000 to 1000 times the data deviation.

We found that a number of participants had reading speeds higher than 282 words per minute, so they were able to read the whole text before the time was up. We used the censoring methods for Bayesian analysis to overcome this limitation [4].

We used multiple chain Monte Carlo simulations (16 chains with 10,000 samples each) for each model.

The results are plotted in Figures 3, 4 and 5. On the figures we plot the probability distributions of the parameters in equations (2) and (4); the x axis

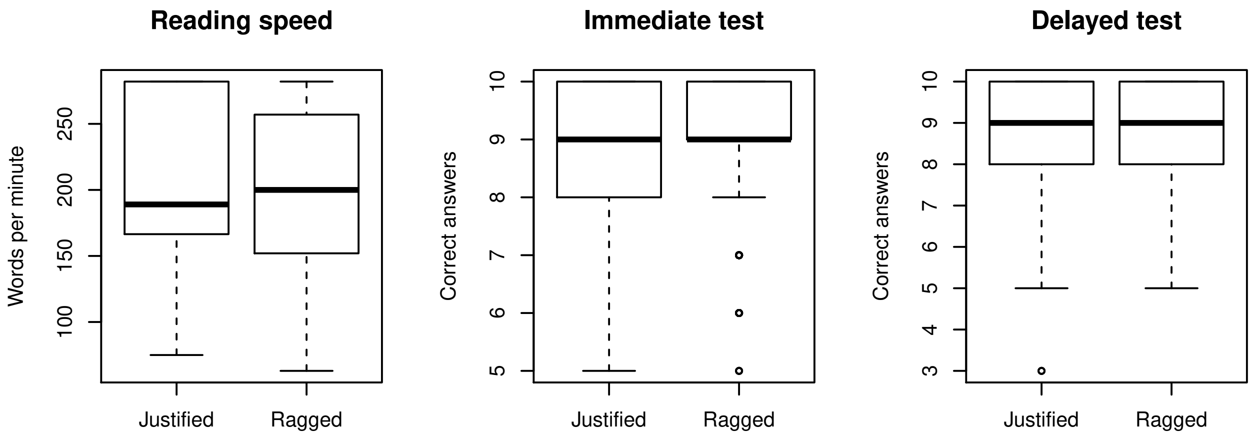


Figure 1: Distribution of reading speed and test results

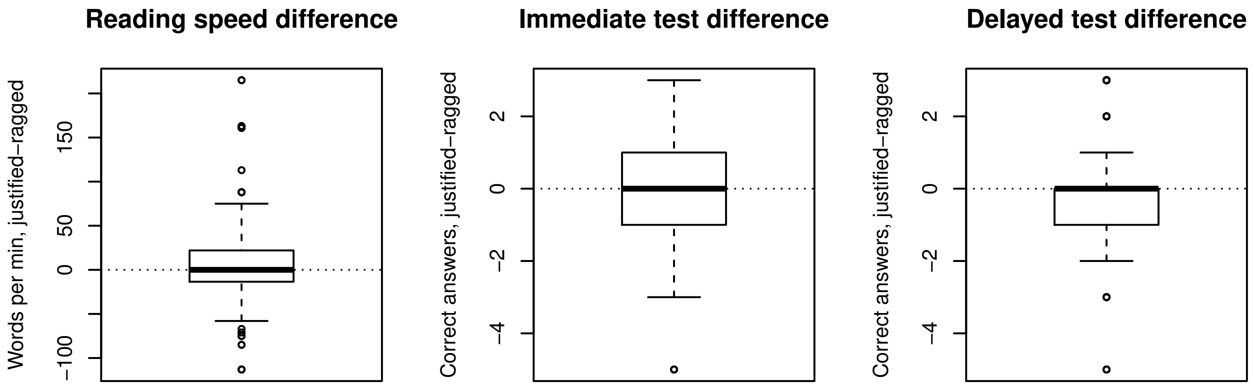


Figure 2: Difference of speed of reading and test results for justified and ragged right texts for the same subject

shows the values of the parameter, while the y axis shows the relative probability of this value according to the simulations. In all figures the first panel represents the individual differences, the second panel the differences between texts A and B, and the last panel the difference between the justified and ragged right texts. On the last panel we plot the zero line (no difference) and 95% interval for the parameters. The significance test of the usual statistics corresponds to the 95% interval being completely to one side of the vertical zero line [4].

The results show that the individual differences in all models dominate the other factors. The difference between the justified and ragged texts is small. However on the 95% level we can say that justified texts are being read faster than the ragged right (by about 7 words per minute), and, even more interesting, the results for delayed tests are better for the ragged right texts. If we convert the log odds to the number of correct answers, we can see that on a 100-question test with 90% correct answers the difference would be about 4 points.

4 Discussion and conclusions

We see that there is a small, but persistent difference between justified and ragged right texts: the former are read slightly faster, but on the delayed tests (when the text is committed to long term memory) give slightly worse results.

Does this mean that one should typeset exam materials in the ragged right fashion? Not necessarily. We do not know whether this effect is specific for our population: Cyrillic readers in Russian, with a significant proportion having Russian as the second language (many students of Bashkir State Medical University have Tatar or Bashkir as their first language). Still, our findings are very intriguing and should be further investigated. One way of interpreting the results might be the interplay between visual image of a word and its commitment to memory: a justified text has hyphenated words with "broken" visual image. If this is the case, the effect should be more pronounced for languages with longer words like Russian and German than for languages with shorter words like English.

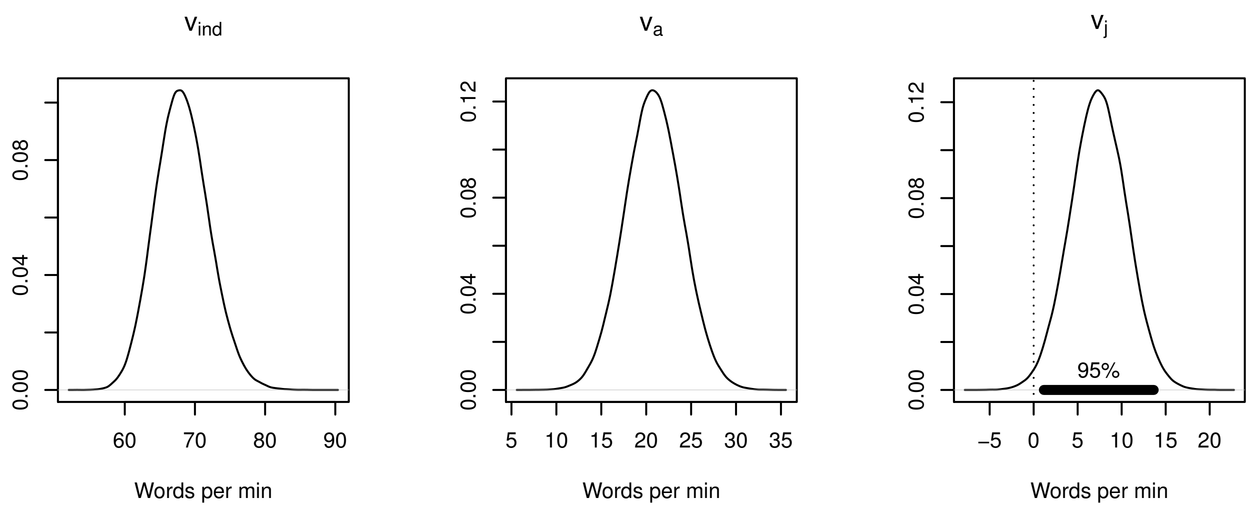


Figure 3: Bayesian estimate for the speed of reading model

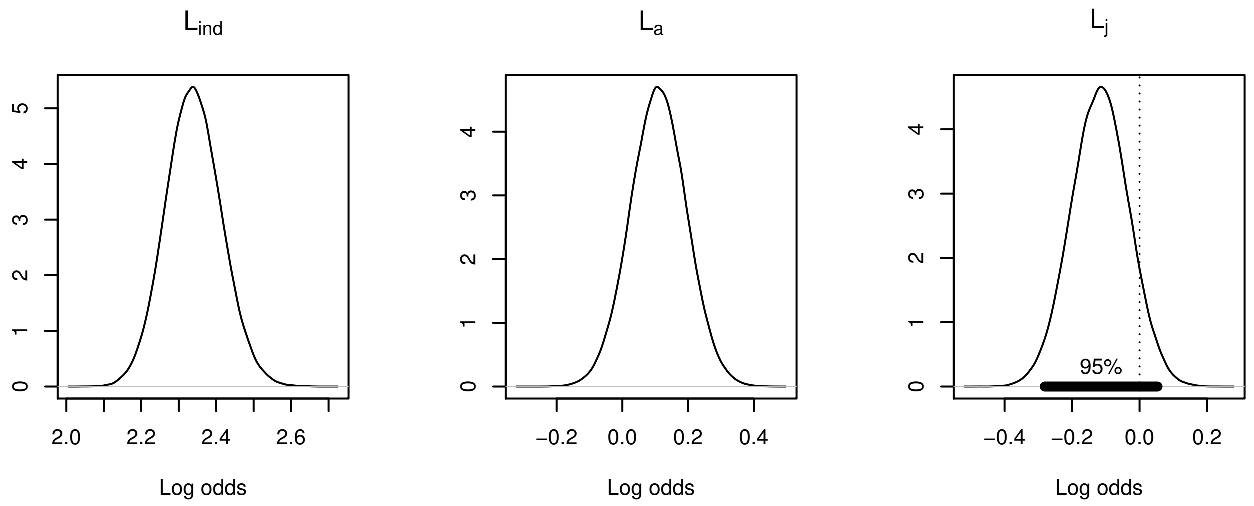


Figure 4: Bayesian estimate for the immediate test model

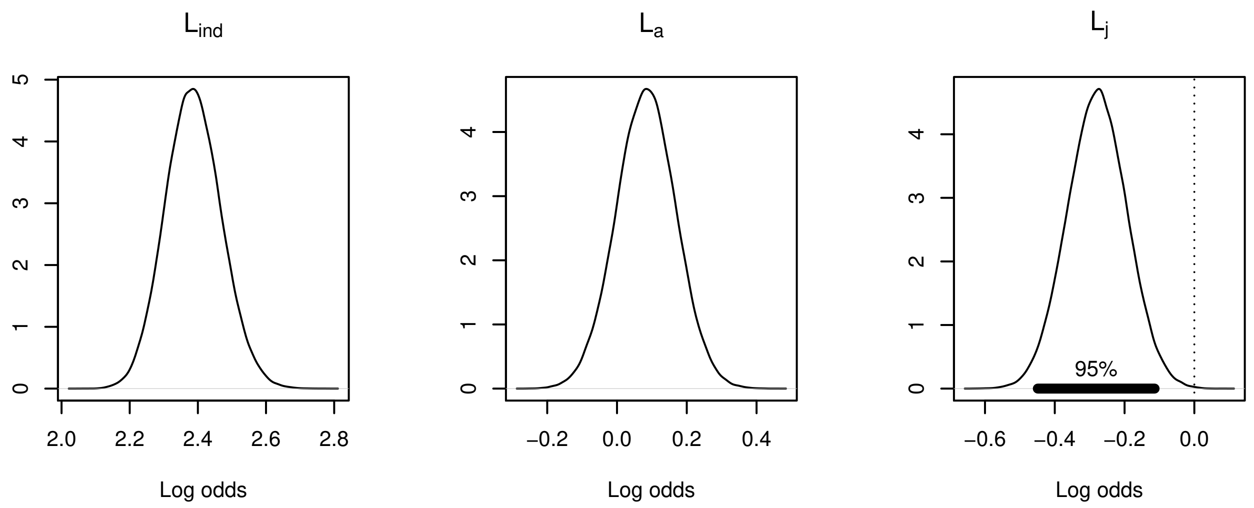


Figure 5: Bayesian estimate for the delayed test model

Acknowledgments

The Bayesian simulations for this work were performed on the George Mason University ARGO computing cluster.

One of the authors (RG) gratefully acknowledges the student travel grant from The European Academy of Neurology (EAN) which allowed him to present these results at the 2nd Congress of EAN (Copenhagen, May 2016).

As usual, the audience at TUG 2016 gave us many interesting ideas, which we will try to use in further research.

References

[1] Leyla Akhmadeeva, Ilnar Tukhvatullin, and Boris Veytsman. Do serifs help in comprehension of printed text? An experiment with Cyrillic readers. *Vision Research*, 65:21–24, 2012.

[2] Leyla Akhmadeeva and Boris Veytsman. Typography and readability: An experiment with post-stroke patients. *TUGboat*, 35(2):195–197, 2014. `http://tug.org/TUGboat/tb35-2/tb110akhmadeeva.pdf`.

[3] Peter D. Hoff. *A First Course in Bayesian Statistical Methods.* Springer, Dordrecht; Heidelberg; London; New York, 2009.

[4] John K. Kruschke. *Doing Bayesian Data Analysis. A Tutorial with R, JAGS, and Stan.* Academic Press, second edition, 2014.

[5] Jan Tschichold. *The New Typography.* University of California Press, Berkeley and Los Angeles, CA, 1998.

[6] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S.* Statistics and Computing. Springer, New York, fourth edition, 2010.

[7] Boris Veytsman and Leyla Akhmadeeva. Towards evidence-based typography: First results. *TUGboat*, 33(2):156–157, 2012. `http://tug.org/TUGboat/tb33-2/tb104veytsman-typo.pdf`.

◇ Leila Akhmadeeva
Bashkir State Medical University, 3 Lenina Str., Ufa, 450000, Russia
`la (at) ufaneuro (dot) org`
`http://www.ufaneuro.org`

◇ Rinat Gizatullin
Bashkir State Medical University, 3 Lenina Str., Ufa, 450000, Russia

◇ Boris Veytsman
Systems Biology School & Computational Materials Science Center, MS 6A2, George Mason University, Fairfax, VA, 22030, USA
`borisv (at) lk (dot) net`
`http://borisv.lk.net`

An informal look into the history of digital typography

David Walden

Introduction

I have always been interested in printed material, but I didn't begin to think explicitly about typography until about 20 years ago when I adopted LaTeX for drafting and formatting books and papers I write. I still didn't have any interest in the history of printing and typography until I prepared a presentation on Boston printing history for TUG 2012. Since then I have been reading (books, papers, Internet websites) and watching YouTube videos about the history of printing and typography that in time led into the digital era.

For TUG 2016 I sketched some of what I (think I) have learned in the hope that my study and thinking will be useful to someone else who is just starting to dig into this history and that people already knowledgeable about printing and typography history might help me understand better.

Several things became clear to me as I undertook preparing for my TUG 2016 presentation.

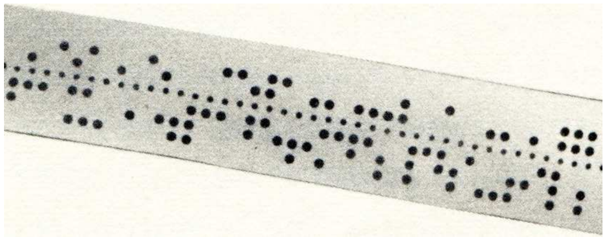
First, I had not previously thought about how printing has long been a massive business throughout the world. It's also a business with broad application: newspapers, periodicals, and books; pamphlets, reports, and legal and financial documents; sheet music; packaging (e.g., on can labels and cardboard boxes); stationery, cards, etc.; announcements, posters, etc.; art reproductions; money, stamps, etc.; cloth, wall paper, etc.; and, from the very earliest days, religious documents of all types. Even as printed materials are being replaced with images on electronic devices, printing remains a massive business. Furthermore, typography seems more relevant than ever, as it now has to address both printed material and a variety of electronic devices.

Second, the dimensions of how printing and typographic activity are accomplished can vary widely:

- large scale production such as big city newspapers; medium or small sized typesetting or print shops; individuals working interactively in their homes on their desktop or laptop computers;
- working with frequent tight deadlines; working with mutually agreed deadlines; working at one's own pace;
- seeking great typographic beauty; putting other considerations first.

One example: big newspapers such as the *Boston Globe* work with tight deadlines, and typographic beauty undoubtedly has to give way at times to more practical considerations. Another example: Donald Knuth being so concerned with typographic beauty that he delayed his work on *The Art of Computing Programming* for years while he developed a typesetting system for his personal use. And there are all combinations in between.

Third, contrary to my naive feeling that the move to digital happened fairly quickly, it now seems to me that the evolution to digital happened over a long time. For instance, many decades before what we now think of as the digital era, the mechanical Linotype machines were being driven by "digital" punched paper tapes, often transmitted by wire from remote locations.



And Monotype casting machines were driven by punched paper tapes since the late 1800s.

To make some sense of this massive field, I find it useful to consider the history of digital typography in terms of four areas that are somewhat overlapping but nonetheless seem able to represent the entire field. My taxonomy is:

1. moving toward digitization of newspapers (representative also of book and periodical publishing and the printing industry more generally);
2. development of digital typesetting for individual interactive use;
3. typesetting algorithms;
4. digital type.

So far my study covers aspects of the first three of these areas in some detail, while merely touching on the fourth area.

Main body of my presentation and paper

After TUG 2016 in Toronto, I prepared the paper based on my presentation there, but concluded that publishing it in *TUGboat* is not the best path. Instead, the current version of the paper has been posted at `tug.org/tug2016/walden-digital.pdf`.

If you are interested, please look at it there and give me your comments. It includes slightly expanded versions of the Introduction here (above) and Reflections here (below) along with more lengthy sections on each of the four topics in my above taxonomy. Section 1 of the paper looks at the history of newspaper typography up to the era when it became fully digital. Section 2 looks at interactive text processing and composition systems from those that ran on the earliest interactive computers (1950s) through contemporary desktop publishing systems. Section 3 looks at algorithms, particularly the history of justification and hyphenation. Section 4 contains my starter list of topics for future investigation into digital type.

All these sections include lots of references (including some marvelous videos) that for the most part are available on the web for ease of access to other new students of this history. The acknowledgements that were left out of my TUG 2016 presentation are also included. And there is a pointer to a private location of my presentation slides — private because it includes many images I used from around the web without bothering to think about licensing.

I am looking forward to continued investigation of the history of digital type and of the newspapers and equipment suppliers who pioneered the various stages of newspaper processes becoming all digital. So far I have found no single history of the latter topic, but I have been gathering pointers to books, papers, websites, and videos. Thus, I will likely make additions over time to my paper at the above url.

Reflections

As I pulled together my TUG 2016 presentation (and drafted the paper version), I have thought back on what I learned from my look into the history of digital typography. Of course, I learned all the stuff I report in the paper, but also plenty more that didn't fit into the paper. Along the way I formed some high level observations.

- What was happening in the four dimensions of my taxonomy have become more and more overlapping and interrelated as we have moved fully into the digital era.
- It was a continuing revelation to me throughout my study how the evolution to digital has been happening for so long; there has been so much intermixing over so many decades of mechanical, photographic, electronically digital technology.
- There is disintermediation, consolidation and despecialization all over the place. Typesetting and design used to be separate specialties, and now every typesetter is a designer or the reverse.

  For my mother-in-law's oral history that my wife produced in 1982, my wife typed and pasted up a photo-ready manuscript; she went to a photo and offset vendor to have the photos sized right and turned into half-tones and to have her 8 1/2 x 11 inch manuscript pages photo reduced to 6 x 9 and for printing of a few dozen copies; and then she went to a separate place for binding. Today, I can produce a book doing all the photo work myself in Photoshop (and the cover in Illustrator), typeset the book myself using LaTeX, produce a ready-to-print PDF, give it to a big printing company (e.g., Lightning Source) or little print shop (e.g., Copyman in southwest Portland, Oregon), and it comes back printed and bound. At the professional level, some people claim that designers have "replaced" printers as well as typographers.
- More generally I feel that the disintermediation, consolidation, and despecialization has led to a lowering of standards. The word processing and desktop publishing systems (and systems like groff and TeX et al.) put powerful typesetting tools in the hands of every amateur and full-time designer, many of whom are not truly professionals. With a little work, anyone can typeset a book or journal article. This lowering of standards is exacerbated by the myriad formats and display devices that must be supported today, for example, hardcopy, ebook, and HTML formats and digital screens of all sizes.
- I suspect that such disintermediation, consolidation, and despecialization is a done deal, and there will be no turning back in general. However, some people beyond the true professionals will still care about publishing aesthetics even if they like being able to do lots of the steps themselves. I have no illusion that the TeX world will again be important to the publishing world at large. I do look forward to seeing automatic aesthetics (such as the pagination work which Frank Mittelbach described in his TUG 2016 presentation) becoming more available to the TeX world — to the world in which I work; and hopefully a few ideas from the TeX world will continue to migrate into the mainstream systems as they have from time to time in the past.

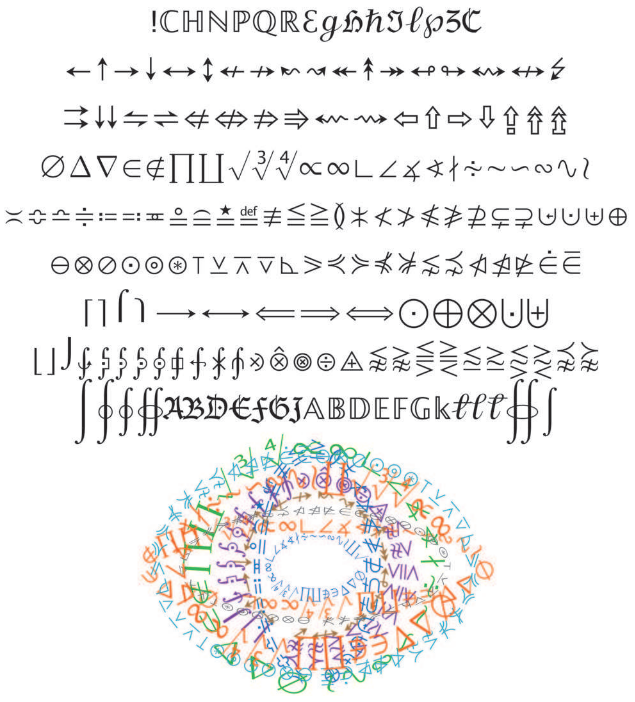
One concluding thought on the digital world. There has never been a better time for the independent researcher. In addition to traditional libraries (and library networks with inter-library borrowing privileges), we now have vast content available via YouTube, Google Books, and professional society and journal digital archives (some open access), and we have web search engines to help us find things. Our own TUG web server makes a significant contribution in the area of digital typography.

◇ David Walden
`walden-family.com/texland`

A short history of the Lucida math fonts

Charles Bigelow

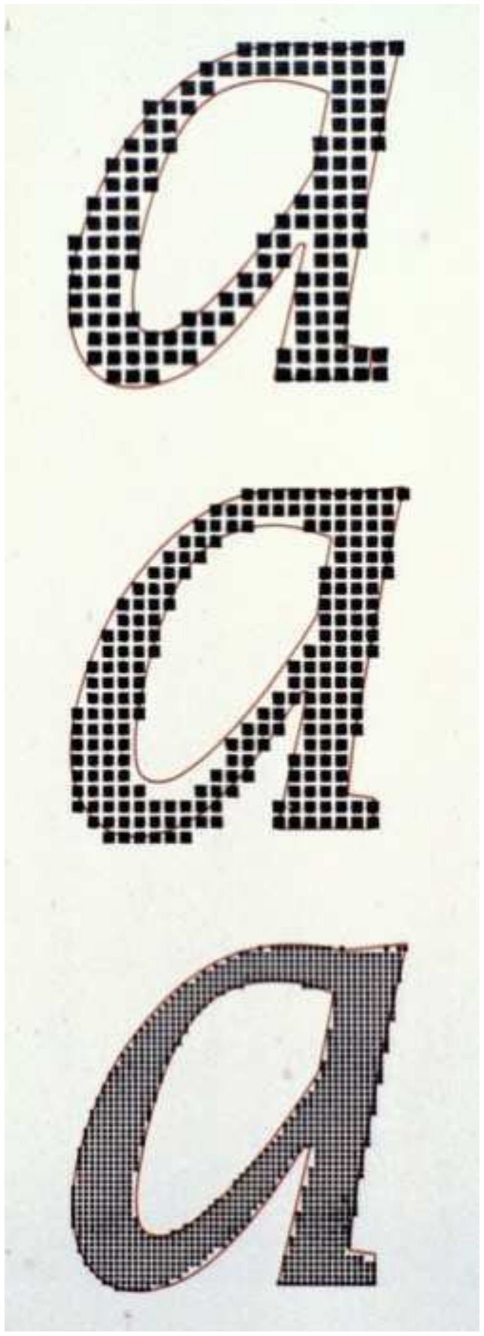
This talk is about the development of Lucida math fonts from the beginnings of Lucida in the early 1980s to the most recent Lucida OpenType math fonts of 2011 and later. The images included here, and more, are available online at `http://tug.org/tug2016/slides/bigelow-lucidamath.pdf`.



1. The above shows a sampling of characters in Lucida Math fonts. There are letter-like symbols and also different shapes and sizes of arrows, operators, relations, and delimiters.

Typography involves several kinds of harmony between graphical elements. Weight, height, width, stroke thickness, proportions, shapes, orientation, position, spacing, and stylistic features such as serifs, are the main relationships that may have to be harmonized. In the typography of mathematics, many symbols are derived from letters but have become separate semantic elements, not units necessarily combined into words. Within the formal language of mathematics, symbols become ideograms — representing ideas — and logograms — spoken as a word or phrase. Naturally, the spoken version changes from language to language, while the meaning of the symbol is constant. Moreover, symbols for mathematics may need adjustment for optical scale because some may be used large and small. And, of course they must be legible.

2. What were the principles on which Lucida was based? Although we were designing a typeface family for emerging technologies — laser printing and screen display — we tended to look backwards to the history of letter forms, in traditional handwriting or calligraphy, which goes back more than two thousand years for Latin scripts and for more than five hundred years before that for Greek.



3. We wanted the forms of the Lucida letters to be related to traditional scribal handwriting, but not necessarily "calligraphy" in its sense of "beautiful writing". Instead we aimed for simplicity because our goal was easy reading under difficult imaging conditions.

Our goal was to adapt traditional forms for simpler rendering at a range of resolutions. We developed the letters in outline formats, principally using Peter Karow's Ikarus system, but the outlines were then converted to bitmaps, either on-the-fly in printers or pre-rasterized for screens, in the technology of the early 1980s. Laser printers of the late 1970s and early 1980s had addressable resolutions around 240 to 300 dots per inch, and bitmap display screen resolutions were around 70 to 75 pixels per inch on screen.

We made many experiments constructing letters from bitmaps, and then used a few digital systems of the early 1980s to study the different pixel patterns when letters were rasterized at different resolutions. At high resolutions, the letters looked like analog typefaces, but at low resolutions, the letters became minimalist aggregations of dots, and many distinguishing features that separated one type design from another were obscured by the rough, rasterized stair-cased shapes and by data loss.

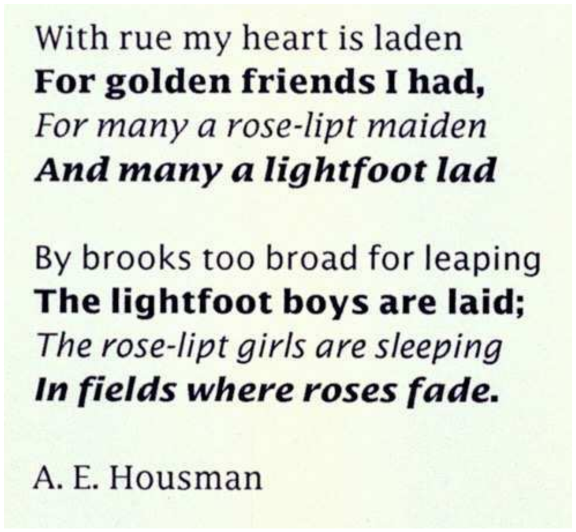
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

4. In 1984, Imagen Corporation released the first Lucida family of seriffed designs. They were simple in form and had several adaptations for lower resolutions, such as short serifs and wide inter-letter spacing to prevent collisions of character elements and clarity at small sizes and greater reading distances from screens, for which loose spacing is often helpful. We made the bold weight with twice the stroke weight of the normal weight, for unambiguous recognition of weight changes at low resolutions. Instead of a "just noticeable difference" we wanted a "dramatically noticeable difference".

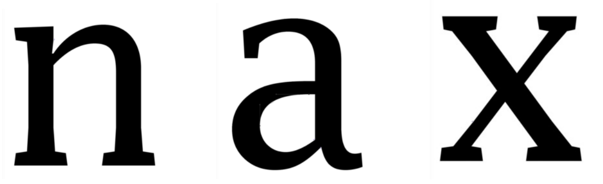
These designs constituted the first new family of types for laser printing and screens. When we developed Lucida, most typefaces being digitized were adaptations of existing metal or photo faces. The adaptations were sometimes made ethically, but often by plagiarism. We hoped that our development of new digital types would spur more original and more ethical approaches to type, and the Imagen firm, started by graduate students and AI researchers at Stanford, agreed. Another five years went by, however, before major firms began to produce original designs.


With rue my heart is laden
For golden friends I had,
For many a rose-lipt maiden
And many a lightfoot lad

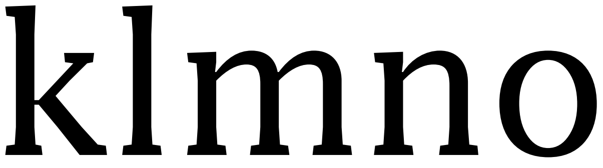
By brooks too broad for leaping
The lightfoot boys are laid;
The rose-lipt girls are sleeping
In fields where roses fade.

A. E. Housman

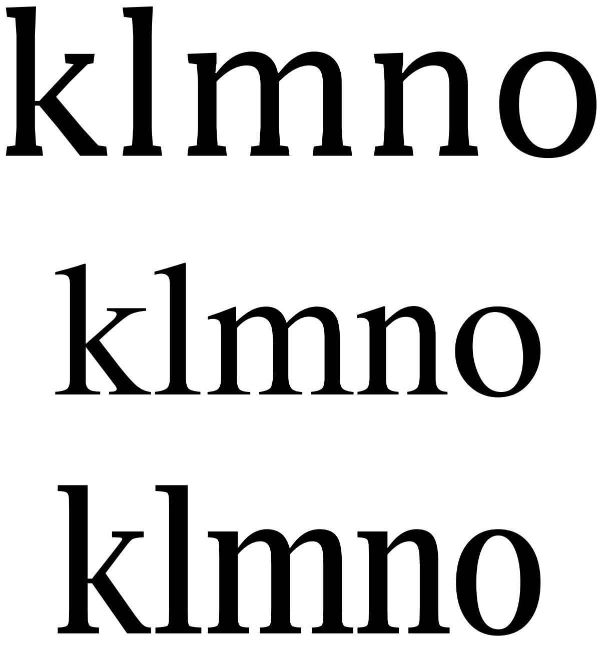
5. In 1985, also with Imagen, we brought out a sans-serif companion to the original seriffed Lucida family. (The second stanza here shows the sans-serif variants, compared to the seriffed originals in the first stanza.) Many of the sans-serif Lucida features, including slightly darkish weight, generous spacing, big x-height, simple forms, and a humanistic style dating back to legible letters of the Italian Renaissance, made the family ideal for screen displays and user interfaces. Lucida Sans Unicode and Lucida Console have been bundled with Microsoft Windows since the early 1990s, and a version of Lucida Sans, named Lucida Grande (because it had a much bigger character set including Greek, Cyrillic, Hebrew, Arabic, Thai, and many signs and symbols) were adopted as the user interface fonts for Macintosh OS X from the beta tests of 2000 until 2014, when Lucida Grande was replaced by a digitized version of a more traditional "grotesque" sans-serif.



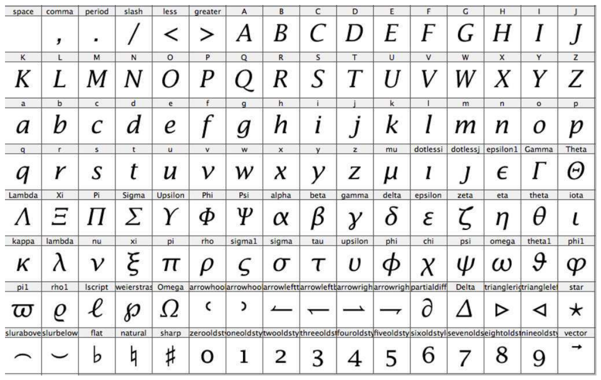
6. The individual letters of the first Lucida seriffed type were crafted in some unusual ways. Serifs and their brackets and some other features were polygonal rather than curved. At the time of its design, 1983–1984, some printing and imaging systems used polygonal outlines, whereas other systems used curved contours, including circular arcs or cubic curves in Bezier or Hermite forms. We believed that it would be better to design the polygonal serifs ourselves rather than let some program automatically render them from curved forms. But, we kept curves for the larger aspects of the letters, like arches and bowls.



7. Here we see diagonal, straight, arched, and fully curved letters in combination, with the characteristic open spacing of the original Lucida.

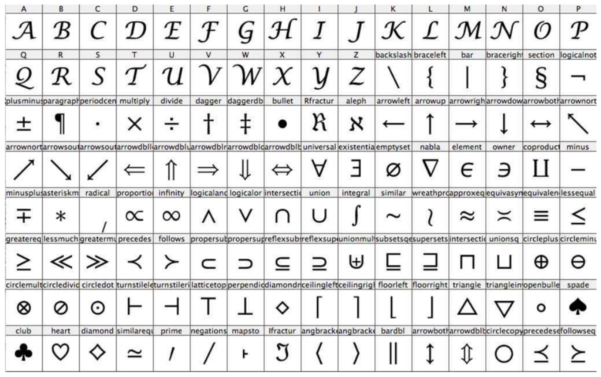


8. Comparison of the first Lucida (top) to Times Roman (middle), which appeared under its own name and in plagiarized forms under other names, to a modification of Lucida (bottom), fitted to the same widths as Times Roman and with modified serifs but keeping the x-height of Lucida (bottom). It looks condensed because the x-height is greater than that of Times Roman but the width is not. We later released a free version of this design under the name "Luxi Serif". It can still be found as a free font from web font firms and others. They are gratis but not open source, because we prohibited modification without our permission. We felt that the artistic aspects of the designs should be ours to control.



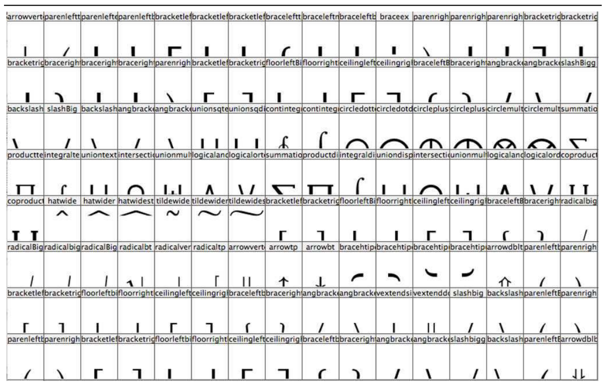
space	comma	period	slash	less	greater	A	B	C	D	E	F	G	H	I	J
	,	.	/	<	>	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
K	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*
q	r	s	t	u	v	w	x	y	z	mu	dotlessi	dotlessj	epsilon1	Gamma	Theta
q	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*	μ	ı	ȷ	ϵ	Γ	Θ
Lambda	Xi	Pi	Sigma	Upsilon	Phi	Psi	alpha	beta	gamma	delta	epsilon	zeta	eta	theta	iota
Λ	Ξ	Π	Σ	Υ	Φ	Ψ	α	β	γ	δ	ε	ζ	η	θ	ι
kappa	lambda	nu	xi	pi	rho	sigma1	sigma	tau	upsilon	phi	chi	psi	omega	theta1	phi1
κ	λ	ν	ξ	π	ρ	ς	σ	τ	υ	ϕ	χ	ψ	ω	ϑ	φ
pi1	rho1	lscript	weierstras	Omega	arrowhoo	arrowhoo	arrowleftt	arrowleftb	arrowrigh	arrowrigh	partialdiff	Delta	triangleri	trianglelet	star
ϖ	ϱ	ℓ	℘	Ω	˓	˒	↼	↽	⇀	⇁	∂	Δ	▷	◁	⋆
slurabove	slurbelow	flat	natural	sharp	zerooldst	oneoldsty	twooldsty	threeoldst	fouroldsty	fiveoldsty	sixoldstyl	sevenolds	eightoldst	nineoldsty	vector
⌢	⌣	♭	♮	♯	0	1	2	3	4	5	6	7	8	9	→

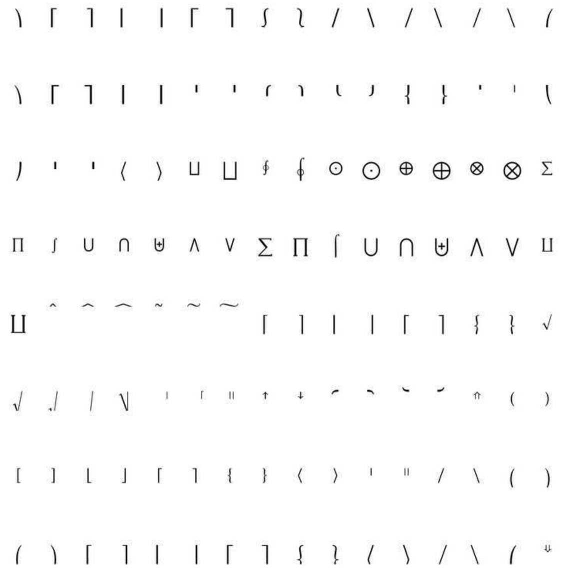
9. Lucida Math Italic, in the same character set as Computer Modern Math Italic. We began the design of math symbols for Lucida in 1985, in part because of the influence of Donald Knuth, who had invited me to join the Stanford faculty and to work with him. If there had been a guide to the various harmonizations needed for math fonts, it would have saved us a lot of effort and time, but there were none, except for examples from which some principles could be inferred. Also, we liked to figure out things by ourselves from first principles. We thought new technologies demanded new design ideas — a lesson we learned by studying with Hermann Zapf — so even had there been guides for math characters, we might have ignored them, being young.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
𝒜	ℬ	𝒞	𝒟	ℰ	ℱ	𝒢	ℋ	ℐ	𝒥	𝒦	ℒ	ℳ	𝒩	𝒪	𝒫
Q	R	S	T	U	V	W	X	Y	Z	backslash	braceleft	bar	braceright	section	logicalnot
𝒬	ℛ	𝒮	𝒯	𝒰	𝒱	𝒲	𝒳	𝒴	𝒵	\	{	|	}	§	¬
plusminus	paragraph	periodcen	multiply	divide	dagger	daggerdb	bullet	Rfractur	aleph	arrowleft	arrowup	arrowrigh	arrowdow	arrowboth	arrownort
±	¶	·	×	÷	†	‡	•	ℜ	ℵ	←	↑	→	↓	↔	↖
arrownort	arrowsou	arrowsou	arrowdbll	arrowdblu	arrowdblr	arrowdbld	arrowdblb	universal	existentia	emptyset	nabla	element	owner	coproduct	minus
↗	↘	↙	⇐	⇑	⇒	⇓	⇔	∀	∃	∅	∇	∈	∋	∐	−
minusplu	asteriskm	radical	proportio	infinity	logicaland	logicalor	intersecti	union	integral	similar	wreathpr	approxeq	equivasyn	equivalen	lessequal
∓	∗	/	∝	∞	∧	∨	∩	∪	∫	∼	≀	≈	≍	≡	≤
greatereq	lessmuch	greatermu	precedes	follows	propersub	propersup	reflexsub	reflexsup	unionmul	subsetsq	supersets	intersecti	unionsq	circleplus	circleminu
≥	≪	≫	≺	≻	⊂	⊃	⊆	⊇	⊎	⊑	⊒	⊓	⊔	⊕	⊖
circlemult	circledivid	circledot	turnstilele	turnstileri	latticetop	perpendic	diamondn	ceilingleft	ceilingrigl	floorleft	floorright	triangle	trianglein	openbulle	spade
⊗	⊘	⊙	⊢	⊣	⊤	⊥	⋄	⌈	⌉	⌊	⌋	△	▽	∘	♠
club	heart	diamond	similarequ	prime	negations	mapsto	Ifractur	angbrack	angbracke	bardbl	arrowboth	arrowdblb	circlecopy	precedese	followseq
♣	♡	♢	≃	′	/	⊦	ℑ	⟨	⟩	‖	↕	⇕	○	⪯	⪰

10. Lucida Math Symbol. We made the strokes of the math symbols fairly thick, in keeping with the general Lucida parameters, and we made many of the symbols fit in the standard figure widths, for easier composition of simple equations and usage in tabular composition, spreadsheets, etc. This tended to make the operators smaller than the letter symbols used for variables. We did this because we thought that the majority of uses of the Lucida math fonts would not be for traditional publishing of mathematics and science, but for simpler uses on screens.





11. We also produced a Lucida Math Extension font. In the first image, you can only see the tops of the characters, because to conform with Donald Knuth's approach with Computer Modern, many of the big delimiters and extensions "hang" from a central position in the character cell. The second image shows the full characters at a smaller size.

$a = b \quad c \neq d$

$p \geq q \quad v \div w$

$a \otimes b \quad c \oplus d$

$p \pm q \quad v \ominus w$

$A \leq B \quad C \supseteq D$

$P \sqsupseteq Q \quad V \cup W$

12. Above is the original Lucida math operators of various kinds with lower-case and capital letters. The weights are harmonized and the symbols are clear and resist digital degradation. We developed these math fonts in 1985, but they were not released until 1990 by Adobe. During that time, principally under the influence of PostScript, laser printing became an enabling technology for digital prepress for print publishing, and fonts intended only for lower-resolution printing were less favored. When the original Lucida and Lucida Math fonts were used for more traditionally printed math publishing, they seemed too strong and dark, although they held up well on screens and laser printers. As an editor at one academic press put it, "Lucida Math seemed too aggressively legible".

nn aa xx

13. For print publishing, and now for high resolution screen displays, we prefer legibility (at least when it is more subliminal than aggressive). In 1987, for Scientific American magazine), we made a modification of the original seriffed Lucida with increased contrast of thick to thin strokes, with more "modern" style serifs, and with tighter letterspacing. An editor at MacWorld magazine said it looked "brighter", so we named it Lucida Bright. It was used as the body text in Scientific American for nine years.

$a = b \quad c \neq d$

$p \geq q \quad v \div w$

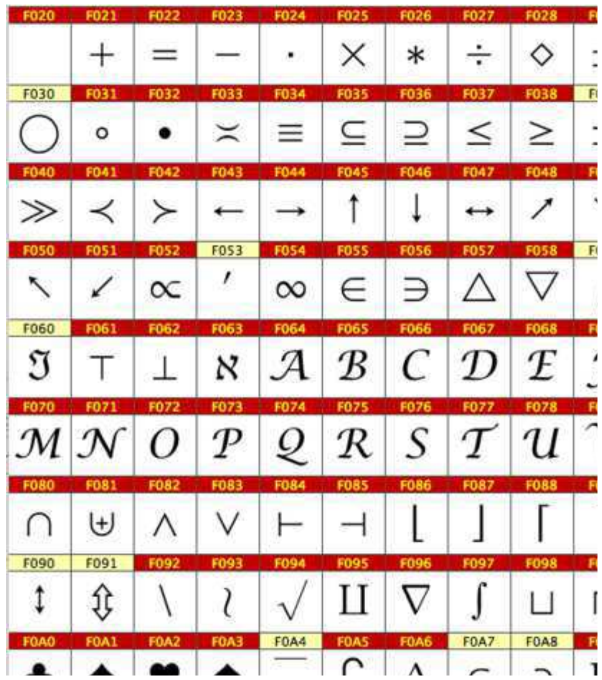
$a \otimes b \quad c \oplus d$

$p \pm q \quad v \ominus w$

$A \leq B \quad C \supseteq D$

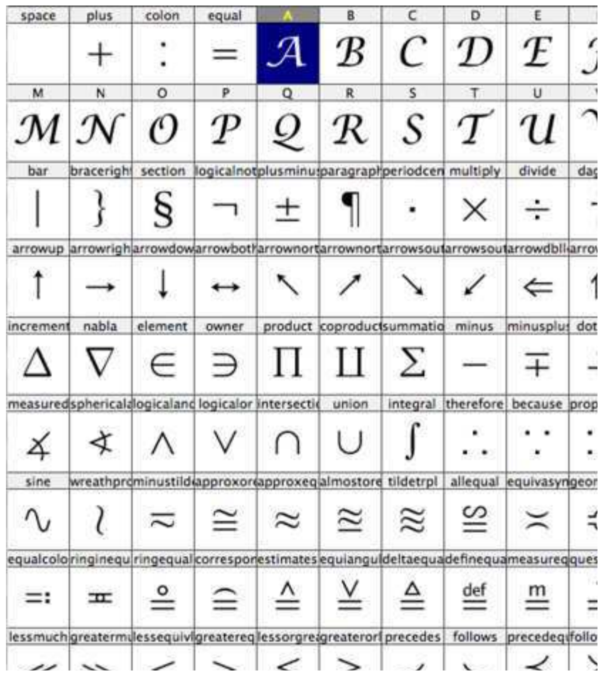
$P \sqsupseteq Q \quad V \cup W$

14. For Lucida Bright, we developed "brightened" math characters. These were released by Microsoft in 1992 in TrueType font format. This prompted requests for TeX-adapted versions in PostScript format. At this point, we came to realize that the production of fully functional, TeX-compatible math fonts required more knowledge of TeX and more effort than we two designers could accomplish on our own.



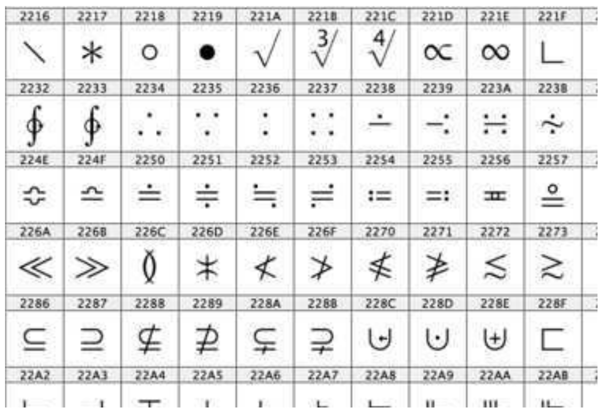
F020	F021	F022	F023	F024	F025	F026	F027	F028
	+	=	−	·	×	∗	÷	◇
F030	F031	F032	F033	F034	F035	F036	F037	F038
◯	∘	•	≍	≡	⊆	⊇	≤	≥
F040	F041	F042	F043	F044	F045	F046	F047	F048
≫	≺	≻	←	→	↑	↓	↔	↗
F050	F051	F052	F053	F054	F055	F056	F057	F058
↖	↙	∝	′	∞	∈	∋	△	▽
F060	F061	F062	F063	F064	F065	F066	F067	F068
ℑ	⊤	⊥	ℵ	𝒜	ℬ	𝒞	𝒟	ℰ
F070	F071	F072	F073	F074	F075	F076	F077	F078
ℳ	𝒩	𝒪	𝒫	𝒬	ℛ	𝒮	𝒯	𝒰
F080	F081	F082	F083	F084	F085	F086	F087	F088
∩	⊎	∧	∨	⊢	⊣	⌊	⌋	⌈
F090	F091	F092	F093	F094	F095	F096	F097	F098
↕	⇳	\	≀	√	∐	∇	∫	⊔
F0A0	F0A1	F0A2	F0A3	F0A4	F0A5	F0A6	F0A7	F0A8
♣	♦	♥	♠					

15. Over the next few years, we worked with Berthold and Blenda Horn, of their firm Y&Y, at turning the Lucida Bright Math fonts into fully functioning PostScript TeX math fonts. It was a tremendous effort made feasible only with the patient and extraordinary collaboration of the Horns.



space	plus	colon	equal	A	B	C	D	E
	+	:	=	𝒜	ℬ	𝒞	𝒟	ℰ
M	N	O	P	Q	R	S	T	U
ℳ	𝒩	𝒪	𝒫	𝒬	ℛ	𝒮	𝒯	𝒰
bar	bracerigh	section	logicalnot	plusminu	paragraph	periodcen	multiply	divide
|	}	§	¬	±	¶	·	×	÷
arrowup	arrowrigh	arrowdow	arrowbot	arrownort	arrownort	arrowsout	arrowsout	arrowdbll
↑	→	↓	↔	↖	↗	↘	↙	⇐
increment	nabla	element	owner	product	coproduct	summatio	minus	minusplus
∆	∇	∈	∋	∏	∐	∑	−	∓
measured	sphericala	logicaland	logicalor	intersecti	union	integral	therefore	because
∡	∢	∧	∨	∩	∪	∫	∴	∵
sine	wreathpr	minustild	approxor	approxeq	almostore	tildetrpl	allequal	equivasyn
∿	≀	≂	≅	≈	≊	≋	≌	≍
equalcolo	ringinequ	ringequal	correspor	estimates	equiangul	deltaequa	definequa	measureq
≕	≖	≗	≘	≙	≚	≜	≝	≞
lessmuch	greaterm	lessequivl	greatereq	lessorgre	greateror	precedes	follows	precedeq

16. During this development, we and Y&Y received requests for more math characters, so we kept adding them, and then came more requests for more characters. If we hadn't declared a halt and frozen the fonts for release, we would probably be adding more characters still. (In truth, we still are, but now to the TUG releases, as we'll see.) To distinguish the Y&Y fonts from the previous Lucida Bright Math, which were in TrueType, not PostScript, and had fewer characters and different encodings, we named the Y&Y releases "Lucida New Math". I suppose we could have called them Lucida Bright Math 2.0, but that would have been confusing, too.

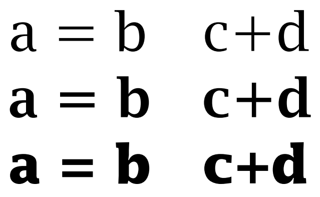


2216	2217	2218	2219	221A	221B	221C	221D	221E	221F
∖	∗	∘	∙	√	∛	∜	∝	∞	∟
2232	2233	2234	2235	2236	2237	2238	2239	223A	223B
∲	∳	∴	∵	∶	∷	∸	∹	∺	∻
224E	224F	2250	2251	2252	2253	2254	2255	2256	2257
≎	≏	≐	≑	≒	≓	≔	≕	≖	≗
226A	226B	226C	226D	226E	226F	2270	2271	2272	2273
≪	≫	≬	≭	≮	≯	≰	≱	≲	≳
2286	2287	2288	2289	228A	228B	228C	228D	228E	228F
⊆	⊇	⊈	⊉	⊊	⊋	⊌	⊍	⊎	⊏
22A2	22A3	22A4	22A5	22A6	22A7	22A8	22A9	22AA	22AB

17. In the years since the Y&Y versions of the Lucida Bright New Math, the Unicode Consortium added math to the Unicode standard, with hundreds of characters beyond the standard TeX repertoire. Eventually, almost twenty years after the Y&Y Lucida versions, Karl Berry asked (following user requests) if we would like to work on OpenType versions in collaboration with TUG, and we began to work on OpenType versions of Lucida Bright and Lucida Math. The task turned out to be several times greater than the original Y&Y development, and also much greater than we anticipated. Partly because we revised some of our design notions during the development process, and partly because we added hundreds of characters.

a = b c ≠ d
a = b c ≠ d
a ⊗ b c ⊕ d
a ⊕ b c ⊗ d
A ≤ B C ⊇ D
A ≤ B C ⊇ D
P ⊒ Q V ∪ W
P ⊒ Q V ∪ W

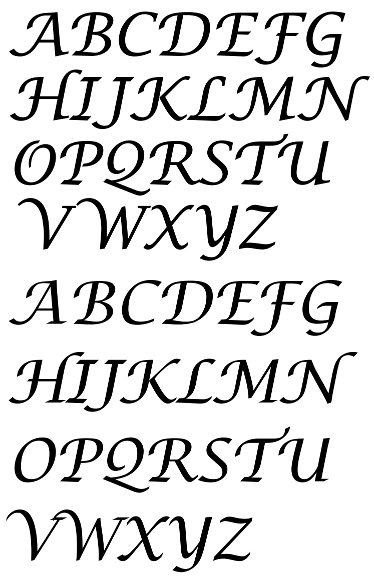
18. In developing the OpenType versions, we decided to abandon our original notion that the math operators should match the figure widths; a comparison of the old and new operator sizes is shown above. Our experience over the years suggested that larger operators were more readable by mathematicians. This is anecdotal evidence; we did not conduct controlled laboratory tests of mathematicians reading equations, which should be done. Nevertheless, we made the math symbols bigger. (Interestingly, one of the most prolific legibility researchers of the 20th century, Miles Tinker, wrote his psychology Ph.D. thesis at Stanford in 1927 on "An Experimental study of legibility, perception, and eye movement in the reading of formulae" ... exactly a half-century before Donald Knuth began work on TEX at Stanford.)



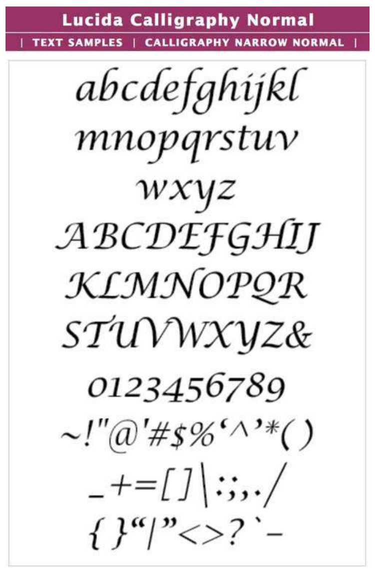
19. We also made a series of bold weight versions of the math symbols. Unlike the original bold weight of Lucida, which had stems twice as thick as the normal weight, the Lucida Bright bold is more accurately a demibold, with stems 1.5 times as thick as the normal. Still very noticeable but not as domineering in text. (The above shows Lucida Bright Math OT, Lucida Bright Math Demi OT, and the original Lucida Bold.)

Not all these bold characters have well-defined semantics in mathematics, but we became accustomed to requests for such things, so we believed that if we designed the characters, mathematicians would find uses for them.

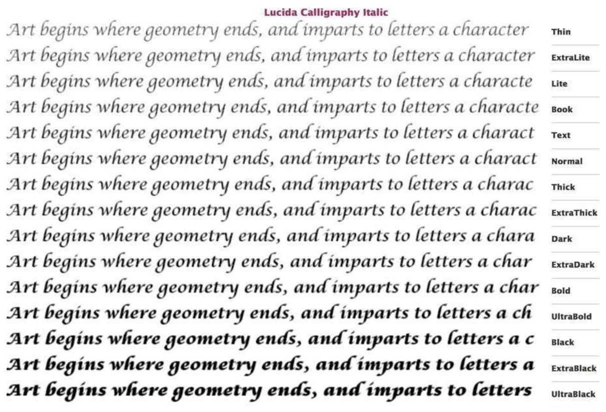
20. In our experience, and evidently in that of other type designers as well, the development of successful, original typefaces and fonts for mathematics requires technical collaboration. We wish to thank those who helped with the technical aspects of Lucida Math font development. For the Adobe PostScript versions of original Lucida Math: Daniel Mills. For the Y&Y PostScript versions, Lucida New Math: Berthold Horn and Blenda Horn. For the TUG OpenType versions, Lucida Math OT: Karl Berry, Khaled Hosny, and Michael Sharpe, plus a cast of testers, bug reporters, commenters, and other advisers: Barbara Beeton, Hans Hagen, Taco Hoekwater, Bogusław Jackowski, Mojca Miklavec, Norbert Preining, Will Robertson, Ulrik Vieth, Bruno Voisin.



21. Work on the Lucida math fonts sometimes resulted in new stand-alone designs, especially scripts. The first such spin-off was "Lucida Calligraphy", a "brighter" version of the original calligraphic capitals in Lucida Math. Above, the original capitals are shown first, with the bright Lucida capitals below.



22. Here is Lucida Calligraphy in the ASCII set. Originally distributed by Microsoft and now by other vendors, including Monotype. It is based on Italian Renaissance chancery cursive, but, like the other Lucida designs, adapted to digital rendering at a range of resolutions and technologies. We see it almost every day somewhere, often used in some surprising way.



Lucida Calligraphy Italic

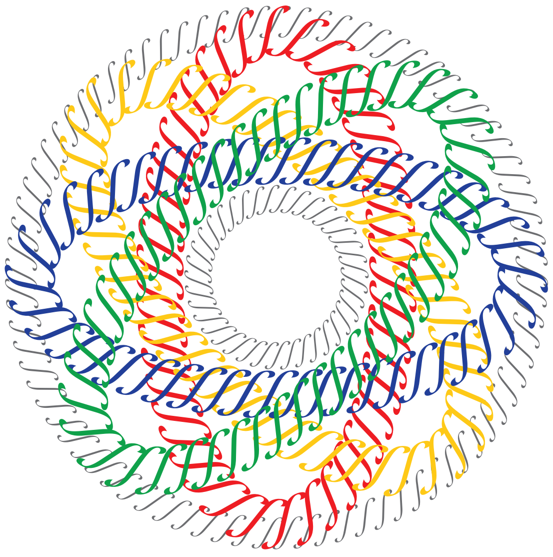
Art begins where geometry ends, and imparts to letters a character	Thin
Art begins where geometry ends, and imparts to letters a character	ExtraLite
Art begins where geometry ends, and imparts to letters a characte	Lite
Art begins where geometry ends, and imparts to letters a characte	Book
Art begins where geometry ends, and imparts to letters a charact	Text
Art begins where geometry ends, and imparts to letters a charact	Normal
Art begins where geometry ends, and imparts to letters a charac	Thick
Art begins where geometry ends, and imparts to letters a charac	ExtraThick
Art begins where geometry ends, and imparts to letters a chara	Dark
Art begins where geometry ends, and imparts to letters a char	ExtraDark
Art begins where geometry ends, and imparts to letters a char	Bold
Art begins where geometry ends, and imparts to letters a ch	UltraBold
Art begins where geometry ends, and imparts to letters a c	Black
Art begins where geometry ends, and imparts to letters a	ExtraBlack
Art begins where geometry ends, and imparts to letters	UltraBlack

23. We have since expanded Lucida Calligraphy through a range of weights. Most of these are in ASCII only for English-speaking users. We are now developing additional characters sets for European languages and orthographies.

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890
&@?!*

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890
&@*?!

24. For the Lucida OpenType Math fonts, we have recently developed an English Roundhand script. It is not a copy of a standard traditional roundhand, but a style with more "zip" (equivalent to "élan") devised by Kris Holmes. This will soon be released in normal and bold weights, first through TUG. (The name hasn't yet been confirmed, but several dozen TUG members at the recent meeting in Toronto signed a petition requesting it be called "Typey McTypeface".)



The mathematician's patterns,
like the painter's or the poet's, must be beautiful;
the ideas, like the colours or the words,
must fit together in a harmonious way.

G. H. Hardy

25. Designing so many hundreds, eventually thousands, of characters for mathematics, starting from scratch has been a thirty year adventure that is not over yet. The English mathematician G.H. Hardy once wrote what could serve as a guide for type design as well as mathematics. "The Mathematician's patterns, like the painter's or the poet's, must be beautiful; the ideas, like the colours or the words, must fit together in a harmonious way." (From *A Mathematician's Apology*.)

26. Kris Holmes could not be at the conference, but she contributed most of what makes the Lucida designs exciting as well as legible, so I want to close with a photo of her when she studied with Hermann Zapf many years ago, looking intently over Zapf's shoulder as he demonstrates how to write with colors; and then again with Zapf, a quarter-century later, when Kris was one of the judges for a Linotype Arabic type design competition.

◇ Charles Bigelow
`lucidafonts.com`

New font offerings: Cochineal, Nimbus15, LibertinusT1Math

Michael Sharpe

Abstract

This document is an expansion of my talk at TUG 2016, detailing the major projects I have worked on during roughly the previous year. Cochineal is a fork of Crimson, an excellent text font family reminiscent of Minion. Nimbus15 is a reworking of the newest versions (2015) of the Nimbus fonts from URW++, which now have added Greek and Cyrillic alphabets. LibertinusT1Math attempts the conversion of Khaled Hosny's LibertinusMath from `otf` to `pfb` with LaTeX support files.

1 Cochineal

Cochineal is an oldstyle text font family containing Roman, Greek and Cyrillic alphabets, derived from Sebastian Kosch's Crimson (2014) font family. Crimson, originally named Crimson Text, has regular, semibold and bold weights, with semibold the least developed. Because of the time and effort it would have taken to bring semibold up to parity with the other weights, Cochineal is made available only in regular and bold weights, and regular and bold italic, in both `otf` and `pfb` formats.

The Crimson fonts in these weights contained about 4500 glyphs, close to 1500 of which were in the regular Roman font. To make glyph coverage uniform across the four styles required making around 1500 new glyphs — a substantial FontForge job of several months duration. The likelihood of bugs, especially in spacing and kerning, in a project of this scale is quite high, as the name might suggest.

(This paragraph is rendered in Cochineal.) The Cochineal package provides fonts in regular, *italic*, **bold**, and ***bold italic***, with a full array of features, including both lining (1234567890) and old-style (1234567890) figures, both in tabular and proportional spacing, superiors $^{\text{Abc123}}$, inferior figures $_{456}$, Small Caps (and **Small Caps**, *Small Caps*, ***Small Caps***), and a swash Q that can be specified globally with the package option `swashQ` or individually with the macro `\Qswash`: Q. In Latin scripts, Cochineal is somewhat reminiscent of Minion Pro, though its italic glyphs tend to be narrower. The Greek (ελληνικά) coverage permits polytonic Greek as well as some ancient forms, and the Cyrillic (кириллица) includes essentially complete `T2A` coverage.

LaTeX support for Cochineal is provided in encodings `OT1`, `T1`, `TS1`, `LY1`, `LGR`, `T2A` and `OT2`. While `LGR` and `OT2` are little-used by authors of `tex` documents in which Greek or Cyrillic is predominant, they seem to be important to Western scholars who need to be able to generate short segments of polytonic and ancient Greek or Cyrillic using a Western keyboard. As `otf` versions of the fonts are provided, they may be used directly in Unicode TeXs by means of the `fontspec` package. Because the `cochineal` package contains a `cochineal.fontspec` file specifying the `otf` file names, just include in your preamble:

```
\usepackage{fontspec}
\setmainfont[Mapping=tex-text]{cochineal}
```

Usage under LaTeX has many options that are spelled out in detail in the package documentation.

For mathematical typesetting with Cochineal, one may use `newtxmath` with option `cochineal`:

```
\usepackage{cochineal}
\usepackage[cochineal,vvarbb]{newtxmath}
\usepackage[cal=boondoxo,frak=boondox]{mathalfa}
```

produces output like:

A Simple *Central Limit Theorem*
Let $X_1, X_2, \cdots$ *be a sequence of* i.i.d. *random variables with mean* 0 *and variance* 1 *on a probability space* $(\Omega, \mathcal{F}, \mathrm{Pr})$. *Let*

$$\mathfrak{N}(y) := \int_{-\infty}^{y} \frac{\mathrm{e}^{-t^2/2}}{\sqrt{2\pi}}\,\mathrm{d}t,$$

$$S_n := \sum_{1}^{n} X_k.$$

Then

$$\mathrm{Pr}\left(\frac{S_n}{\sqrt{n}} \le y\right) \xrightarrow[n\to\infty]{} \mathfrak{N}(y)$$

or, equivalently, for $f \in \mathcal{C}_{\mathrm{b}}(\mathbb{R})$,

$$\mathbb{E} f\big(S_n/\sqrt{n}\big) \xrightarrow[n\to\infty]{} \int_{-\infty}^{\infty} f(t)\frac{\mathrm{e}^{-t^2/2}}{\sqrt{2\pi}}\,\mathrm{d}t.$$

The Cochineal package includes a "theorem" font style, a version of italic with upright punctuation and lining figures, which I think is more suitable than ordinary italic for theorem statements and such. I have abused NFSS by setting `\textsl` to point to the theorem font.

1.1 Production issues

Producing `T2A`, `LGR` and `OT2` encoded support files is somewhat complicated because specialized encoding files must be generated to describe all required ligatures. Those for Western European encodings are fairly easy to obtain using `autoinst`, a wrapper for `otftotfm`, but it turns out that the latter does not respect the spacing parameters in the `otf`, and these must be corrected by passing through a `space-factor` setting. (It seems that this is a common issue when using scripts that call `otftotfm`.) I used the following:

```
autoinst --noupdmap --notitling       \
 --inferiors --superiors --fractions  \
 --target=${tmfc}                     \
 --encoding=LY1,OT1,T1                \
 --extra="--space-factor=1.06635"     \
 --vendor=public --typeface=cochineal \
 *.otf
```

2 Nimbus15

Nimbus15 is derived from the Nimbus fonts issued in 2015 by URW++ by way of Artifex, makers of Ghostscript. They are metric clones of Courier, Helvetica and Times. The 2015 versions appeared in an update to the Ghostscript sources in October 2015. They are included in TEX Live 2016 in PostScript binary format, but lack the associated `.afm` files.

The novelty here is that there are now Greek and Cyrillic glyphs in all the Nimbus fonts. It is indeed regrettable that the license under which the new Nimbus fonts are distributed is incompatible with versions issued prior to 2000, on which the TEX Gyre fonts were based, because these may not now be blended. This limits the utility of Nimbus15 because the TEX Gyre versions are much better except in Greek and Cyrillic. NimbusMono, at least the narrow version described below, may have some reason to exist independently.

Starting from the Artifex distribution, several characters were added throughout: cyrbreve U+F6D4, dotlessj U+0237, and visiblespace U+2423. This was done so the `OT1` and `OT2` encodings would be complete in all cases.

2.1 NimbusSerif ("Times")

`NimbusRomNo9L`, a metric clone of Times, was extended to include Greek (monotonic only) and Cyrillic glyphs. The current distribution from URW++/ Artifex has many errors in spacing and kerning of Greek and Cyrillic glyphs. I expanded the Greek section so that polytonic and some ancient Greek forms are available, added a number of Cyrillic glyphs and tried to correct the spacing and kerning. Since TEX Gyre Termes has much more extensive coverage of Latin glyphs, the only usage for this font that makes sense to me is for standalone Greek and Cyrillic.

2.2 NimbusSans ("Helvetica")

`NimbusSanL`, a metric clone of Helvetica, has been extended to include Greek (monotonic only) and Cyrillic glyphs. I changed the tonos accent from vertical to slanted for consistency with the Courier and Times clones. Analogous to NimbusSerif, given that TEX Gyre Heros has much more extensive coverage of Latin glyphs, the only usage that makes sense to me is for standalone Greek and Cyrillic.

2.3 NimbusMono ("Courier")

In short:

```
NimbusMono-Regular     -> zco-Light
NimbusMono-Bold        -> zco-Bold
NimbusMono-Oblique     -> zco-LightOblique
NimbusMono-BoldOblique -> zco-BoldOblique
```

In addition, a new weight, intermediate between `Light` and **`Bold`**, was created, given the names `zco-Regular` and `zco-Oblique`.

The low asterisk U+204E glyph was added to all the `zco` fonts so that `*` would render correctly.

The glyphs in `Light`, `Regular` and **`Bold`** have stem widths 41, 64 and 100 units respectively. (The stem width of `cmtt10` in this scale is 69, slightly more than zco-Regular, while its advance width is 525, less than zco-Regular at 600.) A few glyphs required modification prior to and following the thickening process.

The Greek glyphs support only monotonic Greek typesetting. Several Greek glyphs were modified from the originals, most importantly alpha (less fish-like), nu (curved, not v-shaped) and Phi (less tall). Thanks are due to Dimitrios Filippou for his important feedback on Greek typographic issues.

Additionally, `zco-Regular` was further modified to a narrow version, `zcoN-Regular`, starting with the FontForge Style/Change Glyph transformations and finishing with manual adjustments to shorten serifs where necessary and make circular outlines narrower. This narrow version, though available only in upright and oblique shapes, seems to me more useful than the overly wide normal Courier. For an example, see the list underneath the section heading above.

3 LibertinusT1Math

The last job mentioned was the conversion of Khaled Hosny's LibertinusMath `otf` to a traditional LATEX setup with `pfb` fonts and accompanying LATEX support files. This was a somewhat complex process, and will require a separate article at a later time.

4 Availability

Packages are available from CTAN and in TEX Live, MiKTEX and other distributions. A list of all my packages on CTAN, including these, is at the url below.

⋄ Michael Sharpe
UCSD
`http://ctan.org/author/id/sharpe`

MFCONFIG: A METAFONT plug-in module for the Freetype rasterizer

Jaeyoung Choi, Sungmin Kim, Hojin Lee and Geunho Jeong

Abstract

One of METAFONT's advantages is its ability to create font variants by changing values of parameters representing font characteristics. This advantage can be applied not only to Latin alphabetic characters, but also to complicated CJK (Chinese-Japanese-Korean) characters. Second, font families like bold, italic, and bold-italic do not need to be created separately for METAFONT, because it can automatically generate a variety of styled fonts via changing parameter values. Therefore, METAFONT can reduce the development time and cost for production of a font family. It is not possible, however, to directly use METAFONT in modern font engines; the output must be changed to an outline font format if it is to be used in a current computing environment.

In this paper, a module named MFCONFIG, enabling direct usage of METAFONT on Linux is proposed. It is a plug-in module for the FONTCONFIG library, and must also be installed with the popular rasterizer Freetype. FONTCONFIG and Freetype are already compatible with other digital font types, both bitmap and outline; MFCONFIG adds METAFONT support. Furthermore, by setting various parameters, the proposed module supports a variety of font styles, all generated from METAFONT.

1 Introduction

Text is an effective way to communicate and record information. With the growing use of smart devices, digital fonts are more commonly used than analog fonts. Although many styles of digital fonts have been created, they still do not meet the requirements of all users, and users cannot change digital font styles freely [10]; for instance, if a user wants to use a thinner outline font, either he/she has to find a thinner styled font, or an in-application function to change the font thickness. As several different features of font style are needed, though, such searching or changing of font style of an existing font is typically not easy. A perfect application satisfying users' diverse requirements regarding font styles does not exist. Also, it is impossible to provide all styles of all fonts in accordance with users' preferences.

Currently, popular digital fonts, either bitmap or outline, have limits on changing font style [8]. However, METAFONT is a structured font definition that allows users to change the font style freely. METAFONT, a font system for TEX, was created by D. E. Knuth [4]. It has functions for drawing characters and parameters to determine the font styles. When the user changes the parameters, the font style is changed automatically. Therefore, a variety of styled fonts can be generated from one METAFONT font. Figure 1 shows a variety of styled fonts created by the changing of the thickness and slant; examples of two thickness and slant styles for the Latin letter "A" and the Chinese character "漢" are shown. If other features such as serif and pen are applied together, a greater variety of styled fonts can be generated.

Most users, however, are unable to use METAFONT on their PCs because current font engines do not support METAFONT. METAFONT fonts are expressed as program source code, completely different from standard digital bitmap and outline fonts. If a user wants to use a specific METAFONT font in a general font engine such as Freetype, then he/she needs to convert the METAFONT font into the needed outline font format.

In the case of Roman characters, the design of "only" several hundred characters is required; moreover, their shapes are generally simpler than those of CJK (Chinese-Japanese-Korean) characters. In the mid-1980s, when METAFONT was introduced, hardware was not fast enough for real-time conversion of the METAFONT fonts into the corresponding bitmap or outline fonts. Moreover, outline fonts are more commonly used than METAFONT.

Current PC hardware, however, has sufficient performance for the real-time execution of METAFONT. If METAFONT could be used directly in a

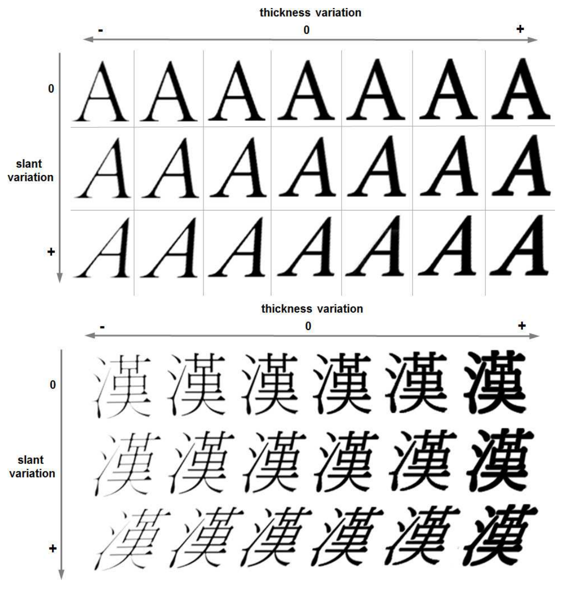


Figure 1: METAFONT style variations

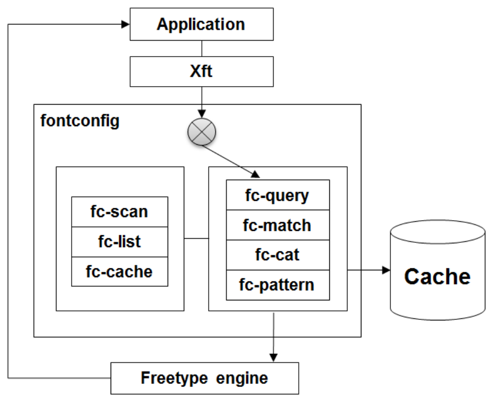


Figure 2: Architecture of FONTCONFIG

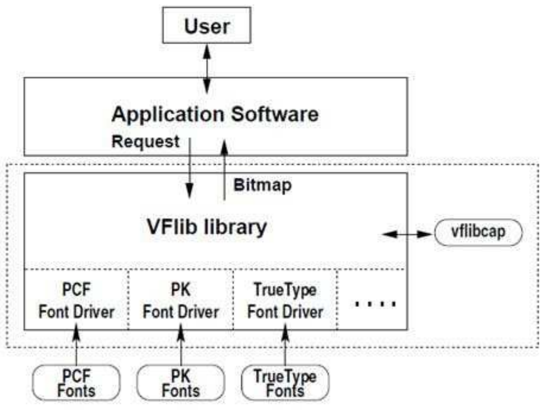


Figure 3: Architecture of VFlib

PC, then users could easily make and use a variety of styled fonts by themselves. As we previously saw, one METAFONT font can generate a variety of styled fonts by changing style parameter values. Therefore, METAFONT can save great amounts of time and repeated effort in terms of font design to make font families of plain, italic, bold, and bold-italic fonts. In particular, in the case of CJK usage, METAFONT could be an effective way to make and display a variety of font styles — because, compared to alphabetic scripts, CJK characters are both complicated in shape and expressed by combinations of radicals.

In this paper, a METAFONT module that enables direct METAFONT usage on Linux is proposed. It is possible to plug this module into FONTCONFIG to provide digital font information to the FreeType engine. When the MFCONFIG module is used, conversion of a METAFONT into corresponding outline fonts becomes unnecessary. It is simple to change font styles by applying new parameter values. Also, this module can interact with most existing FONTCONFIG functions without modification of either FONTCONFIG or Freetype. The MFCONFIG module therefore has good usability and compatibility for the support of METAFONT in the Freetype engine.

2 Existing font systems

FONTCONFIG [7] provides extended font configuration for the Freetype rasterizer, and the Xft (X–FreeType) library [6] has been developed to provide interfaces between applications and Freetype. These font libraries are able to collect font information on the current PC system such as font paths, style information, extra meta information, and so on. Figure 2 shows a font-output sequence that is required for applications using the X Window system under Linux. After an application sends a font request according to name and style to the Xft library, it also delivers the request information to FONTCONFIG. FONTCONFIG uses its internal commands to check the following conditions: (1) whether the requested font is installed, (2) whether the style of the user's request has been applied to the stored font, and (3) whether the requested font has already been stored with the printing format in the cache. (4) If the requested font is not stored in the cache, it needs to be converted into the requested printing format and stored in the cache, and (5) the requested font in the cache is selected and then delivered to Freetype. Lastly, the requested font is printed with the font styles.

FONTCONFIG is a library for Freetype, and capable of supporting general digital font formats that can also be processed in Freetype. The architecture of FONTCONFIG is shown in Figure 2. It can support TrueType, OpenType, Type1, CFF, PFR, and BDF, but it does not support METAFONT. For the direct support of METAFONT in FONTCONFIG, it might be necessary to change the internal implementation of FONTCONFIG; for instance, changing the overall processes in FONTCONFIG from `fc-scan` (for font searching) to `fc-pattern` (for the matching of the styled font pattern). This is not a simple task.

Additionally, the Xft library interface exists between an application and FONTCONFIG, intended for providing font information such as font name and size. It would not be a good approach to modify Xft to support METAFONT, as it would likely reduce Xft's performance.

VFlib [2, 3] is a font driver system for supporting a variety of font types. The system supports virtual fonts like BDF, PCF, and TrueType, as shown in Figure 3. It provides a database of general font information, and a useful API for the supported font

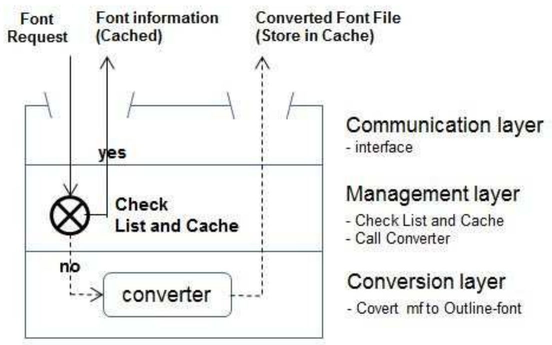


Figure 4: Three layers of the MFCONFIG module

types. VFlib includes separate modules for each font type, so a new module could be added to support METAFONT. But VFlib is a complicated system consisting of many different kinds of font drivers and an information dataset of default font information. In addition, the VFlib interface is required for an application to use the VFlib library. Therefore, to add a METAFONT module to VFlib, additional functions must be implemented for every relevant application, which is not practical. So, VFlib is not suitable to add support for METAFONT.

The proposed MFCONFIG module in this paper combines the following two features: (1) the process for the printing of digital fonts in FONTCONFIG, and (2) the font driver architecture of VFlib. The module can process METAFONT independently, and it can be easily installed or removed since it is implemented as a plug-in module. Also, the steps that are used for its implementation are similar to FONTCONFIG's internal commands, so METAFONT can be used along with the existing digital font formats.

3 Implementing the MFCONFIG module

As shown in Figure 4, the MFCONFIG module consists of the following three layers: communication, management, and conversion. The communication layer provides an interface between FONTCONFIG and MFCONFIG. The management layer checks whether the requested METAFONT is ready in the cache. If not, it sends a request message to the conversion layer to convert the METAFONT font. The conversion layer makes a new outline font file by using the requested METAFONT and the customized style values. The resulting outline font is stored in the cache.

As shown in Figure 5, MFCONFIG can be used as a plugin for FONTCONFIG. First, an application requests a font from the Xft library (step 1). Next, FONTCONFIG sends the font information and the values of the style parameters to MFCONFIG through the interface of the communication layer (step 2). This interface checks if the requested font is a METAFONT font or not.

In the case of METAFONT, *mf-query* analyzes the requested information, and *mf-match* tries to find this METAFONT from *mf-list*. If *mf-list* does not have the information, the requested METAFONT font is not installed. In this case, *mf-query* returns a "not found" flag to FONTCONFIG (step 3). Otherwise, if the METAFONT font is installed and is already stored in the cache, *mf-query* returns a "found" flag to FONTCONFIG (step 3).

One more case: the requested METAFONT font is installed, but the corresponding outline font is not yet in the cache. In this case, *mf-converter* in the conversion layer needs to convert the METAFONT font into the corresponding outline font (step 2a). In this step, the METAFONT font and the styled parameter values from the application are required for the conversion. After conversion, the outline font is stored in the cache (step 2b), and *mf-query* sends the "found" flag to FONTCONFIG (step 3).

After step 3, the remaining steps are the default steps of FONTCONFIG. The font information is sent to the internal programs of FONTCONFIG (step 4) that try to find the corresponding outline font in the cache (step 5); then, this outline font is sent to the Freetype rasterizer (step 6). If MFCONFIG returns the "not found" flag, then FONTCONFIG uses a default font file. Lastly, the Freetype engine renders the outline font that was made from the requested METAFONT with the styled parameter values.

The details of the three layers in MFCONFIG are presented below. First, the communication layer, which is an interface between FONTCONFIG and MFCONFIG, is the starting point of the MFCONFIG module. Therefore, the Freetype engine receives METAFONT font information from FONTCONFIG just as with existing font formats. The main functions of the interface are as follows: (1) delivery of the requested METAFONT information to the management layer, (2) returning results to FONTCONFIG, and (3) storage of the outline-font file from *mf-converter* in the cache memory.

The major programs of the MFCONFIG module operate in the management layer. This layer is in charge of "searching" and "managing". "Searching" is an independent function, finding all installed METAFONT fonts and storing the information in a list. This list is used for checking whether a specific font is installed or not, and for fetching its information quickly. The searching is implemented in *mf-scan* and *mf-list* which, as shown in Figure 6,

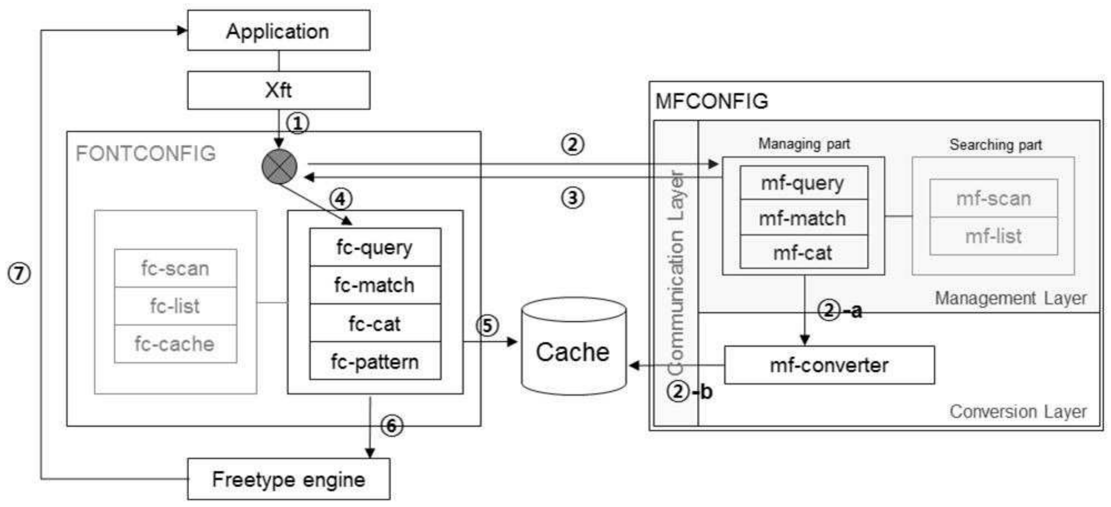


Figure 5: MFCONFIG architecture linked into fontconfig

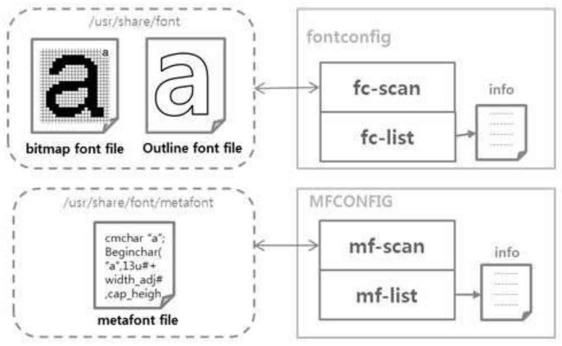


Figure 6: Management layer, cf. FONTCONFIG

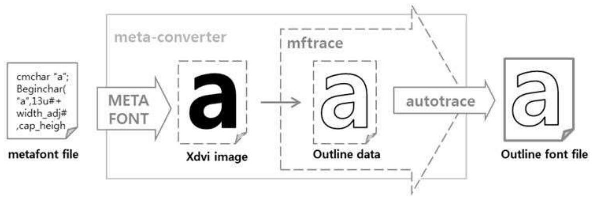


Figure 7: Conversion layer operation

work similarly to FONTCONFIG's *fc-scan* and *fc-list*.

"Management" is a core process of the MFCONFIG module that is responsible for the following actions: (1) checking if the requested METAFONT font is prepared in the list, (2) checking if the corresponding outline font is stored with the requested style in the cache, and (3) if the outline font is not so stored, check whether conversion of the METAFONT font into the corresponding outline font is needed. If the outline font has already been prepared in the cache, then a notification is sent directly from MFCONFIG to FONTCONFIG to use it, and FONTCONFIG sends the cached outline font to Freetype. If the outline font is not stored in the cache, then the conversion layer converts the METAFONT font into the corresponding outline font by applying the style parameters, as shown in Figure 7. The resulting outline font is then stored in the cache, and a notification from the management layer through the communication layer tells FONTCONFIG to use the font.

Thus, the work of the MFCONFIG module is perfectly compatible with the standard FONTCONFIG, and it can provide new functions to support METAFONT. The module handles management of METAFONT fonts and their conversion to corresponding outline fonts in real time. When a different style of an METAFONT font is requested, MFCONFIG can conveniently display the resulting font on the screen by applying the style values to the METAFONT fonts. Therefore MFCONFIG provides good usability for METAFONT. In addition, it is not necessary to generate the font family set of plain, bold, italic, and bold-italic in advance with respect to MFCONFIG, because the font styles can be generated easily.

4 Examining the MFCONFIG module

For performing the experiments of this study, an application for the use of the X Window system in Linux was developed, and the display of a text file was attempted with the use of a variety of font files. The TrueType font family FreeSerif was used in the usual four font styles (*normal, bold, italic, bold-italic*) and Computer Modern was used for METAFONT. The Computer Modern fonts were examined with the four similar styles *normal, thickness, italic,* and *thickness+italic*. The sample text comprises over 2,000 words and over 8,800 characters, including

Styles	Output	Font files
Normal	Computer	FreeSerif.ttf
Bold	**Computer**	FreeSerifBold.ttf
Italic	*Computer*	FreeSerifItalic.ttf
Bold+Italic	***Computer***	FreeSerifBoldItalic.ttf

Table 1: The FreeSerif font family

Style (variable)	Style 1	Style 2	Style 3
Normal	Computer (basic)	Computer (width x2)	Computer (width / 3)
Stroke (hair, stem, curve)	Computer (+20,+10,+10)	Computer (+30,+10,+10)	Computer (+20,+20,+20)
Slant (slant)	Computer (1/4)	Computer (1/2)	Computer (1/3)
Stroke + Slant (slant, Hair, Stem, Curve)	Computer (1/4,+20,+10,+10)	Computer (1/2,+30,+10,+10)	Computer (1/3,+20,+20,+20)

Table 2: Computer Modern fonts in various styles, made by changing style variables

space characters. For performance analysis of the Freetype rasterizer, the time between the requesting of a font with styles from an application and the successful display of text on screen was measured and compared.

Table 1 shows the FreeSerif font family in the four styles, and Table 2 shows 12 styles for the Computer Modern METAFONT. These styles were all made from one original METAFONT font by simple changes of the style parameters. Therefore, the METAFONT font has a good capability of generating various font styles.

For displaying text in an application, the four FreeSerif files from Table 1 and Style 1 of Computer Modern from Table 2 were used. For the CM style, the four parameter values are *hair, stem, curve, and slant*. The three parameters of *hair, stem*, and *curve* are related to the *bold* style, but these parameters are different for lowercase and uppercase. The *slant* parameter is related to the *italic* style. The chosen parameter values for the representation of a bold style are *hair*+20, *stem*+10, and *curve*+10, while *slant* is 0.25 for a representation of the italic style.

Table 3 shows the average time to print both the FreeSerif and Computer Modern contents, and Figures 8 and 9 show the displayed results. In this experiment, with the FreeSerif TrueType fonts, results from 10 ms to 30 ms were obtained, and the

Type	FreeSerif	Computer Modern
(a) Normal	15 ms (10~30)	70 ms (50~80)
(b) Bold	18 ms (10~30)	85 ms (70~100)
(c) Italic	16 ms (10~30)	105 ms (70~110)
(d) Bold+italic	16 ms (10~30)	100 ms (90~120))

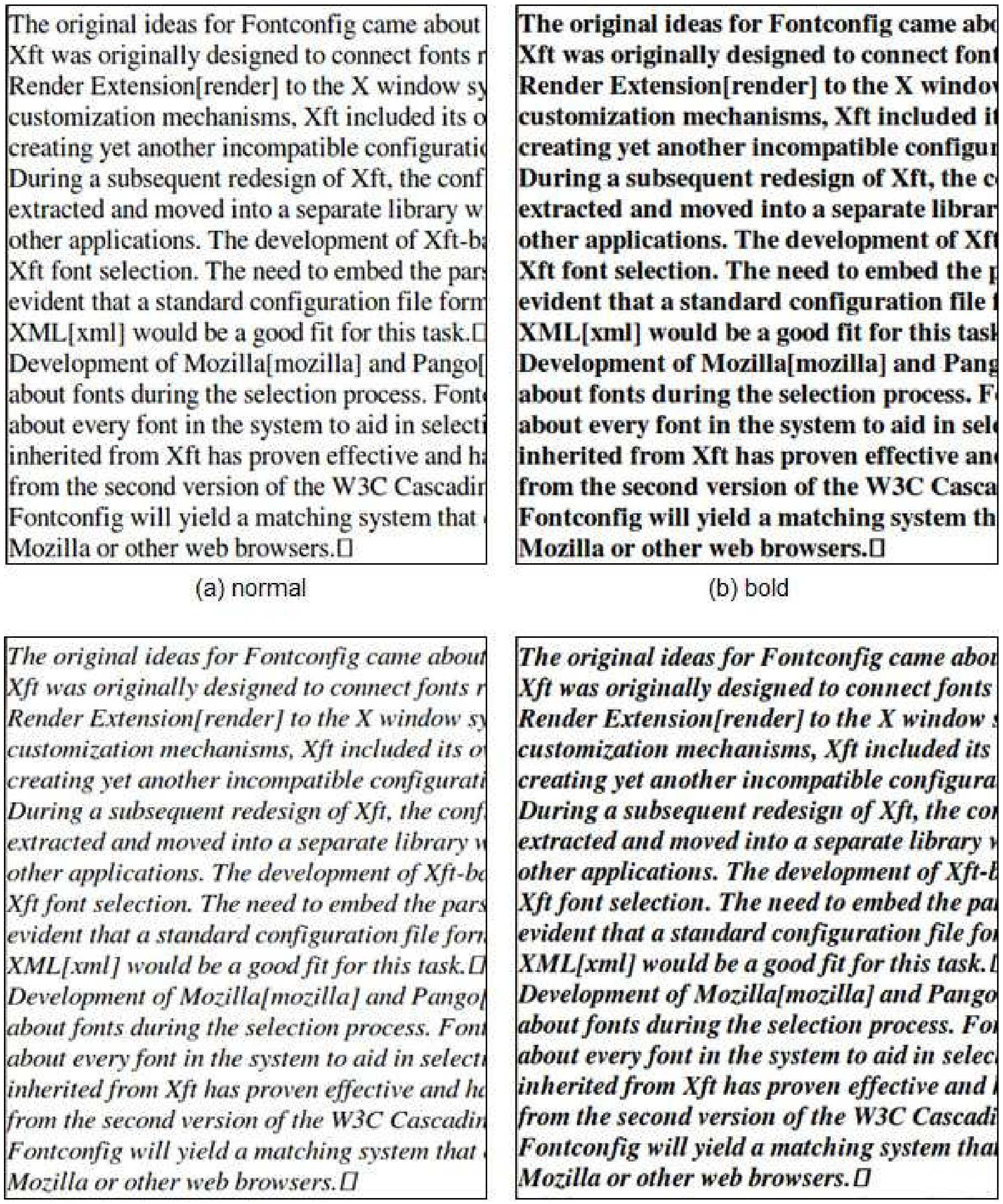
Table 3: Average time for display of TrueType and METAFONT fonts (milliseconds)

average time is 16 ms. Therefore, extra time was required for the conversion of the TrueType fonts. In the case of the Computer Modern METAFONT font, the result is much slower than for FreeSerif, because of the additional time needed for the conversion of the METAFONT font into the corresponding outline font. The obtained results are from 50 ms to 120 ms, and the average time is 90 ms. Even though this is 10 times slower, 90 ms is still a reasonable time for rendering text to a display. Thus, we can conclude that the MFCONFIG module can be used with FONTCONFIG to support METAFONT in (almost) real time on a modern Linux PC.

The MFCONFIG module is a convenient system to provide users with various styled fonts on screen by applying style parameters directly to the METAFONT font. An METAFONT font can be used in just the same way as a TrueType font.

In this paper, we discuss the METAFONT font Computer Modern, which provides alphanumeric values and symbols. It is possible to perform tests with other METAFONT fonts from CTAN (Comprehensive TEX Archive Network) directories that support languages such as Russian and Thai. Unfortunately, difficulty was experienced when complicated CJK fonts are used.

CJK font definitions are very complicated compared to alphabet-based fonts, and they are composed of several thousand phonemes. A number of studies have been conducted to partially implement CJK fonts, such as Hóng-zì [5, 11, 12] and Tsukurimashou [9], including the use of a structural font generator using METAFONT for Korean and Chinese [1], and others. However, CJK fonts created with METAFONT have still not nearly reached the level of quality and practicality reached by commercial offerings. The authors expect that using the MFCONFIG module for the generation of CJK fonts will take more time. It may, however, be possible to solve this problem by optimizing the meta-converter in the conversion layer. Currently, the meta-converter works with *mftrace* and *autotrace* programs, which take a long time to generate outline fonts.



The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configuratio
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-b
Xft font selection. The need to embed the pars
evident that a standard configuration file form
XML[xml] would be a good fit for this task.□
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Font
about every font in the system to aid in selecti
inherited from Xft has proven effective and h
from the second version of the W3C Cascadir
Fontconfig will yield a matching system that
Mozilla or other web browsers.□

(a) normal

**The original ideas for Fontconfig came ab
Xft was originally designed to connect font
Render Extension[render] to the X windo
customization mechanisms, Xft included it
creating yet another incompatible configur
During a subsequent redesign of Xft, the c
extracted and moved into a separate librar
other applications. The development of Xft
Xft font selection. The need to embed the
evident that a standard configuration file
XML[xml] would be a good fit for this tas
Development of Mozilla[mozilla] and Pang
about fonts during the selection process. F
about every font in the system to aid in sel
inherited from Xft has proven effective an
from the second version of the W3C Casca
Fontconfig will yield a matching system th
Mozilla or other web browsers.□**

(b) bold

*The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configurati
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-b
Xft font selection. The need to embed the pars
evident that a standard configuration file forn
XML[xml] would be a good fit for this task.□
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Font
about every font in the system to aid in selecti
inherited from Xft has proven effective and h
from the second version of the W3C Cascadin
Fontconfig will yield a matching system that
Mozilla or other web browsers.□*

(c) italic

***The original ideas for Fontconfig came abo
Xft was originally designed to connect fonts
Render Extension[render] to the X window
customization mechanisms, Xft included its
creating yet another incompatible configura
During a subsequent redesign of Xft, the co
extracted and moved into a separate library
other applications. The development of Xft-
Xft font selection. The need to embed the pa
evident that a standard configuration file fo
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango
about fonts during the selection process. Fo
about every font in the system to aid in selec
inherited from Xft has proven effective and
from the second version of the W3C Cascadi
Fontconfig will yield a matching system tha
Mozilla or other web browsers.□***

(d) bold-italic

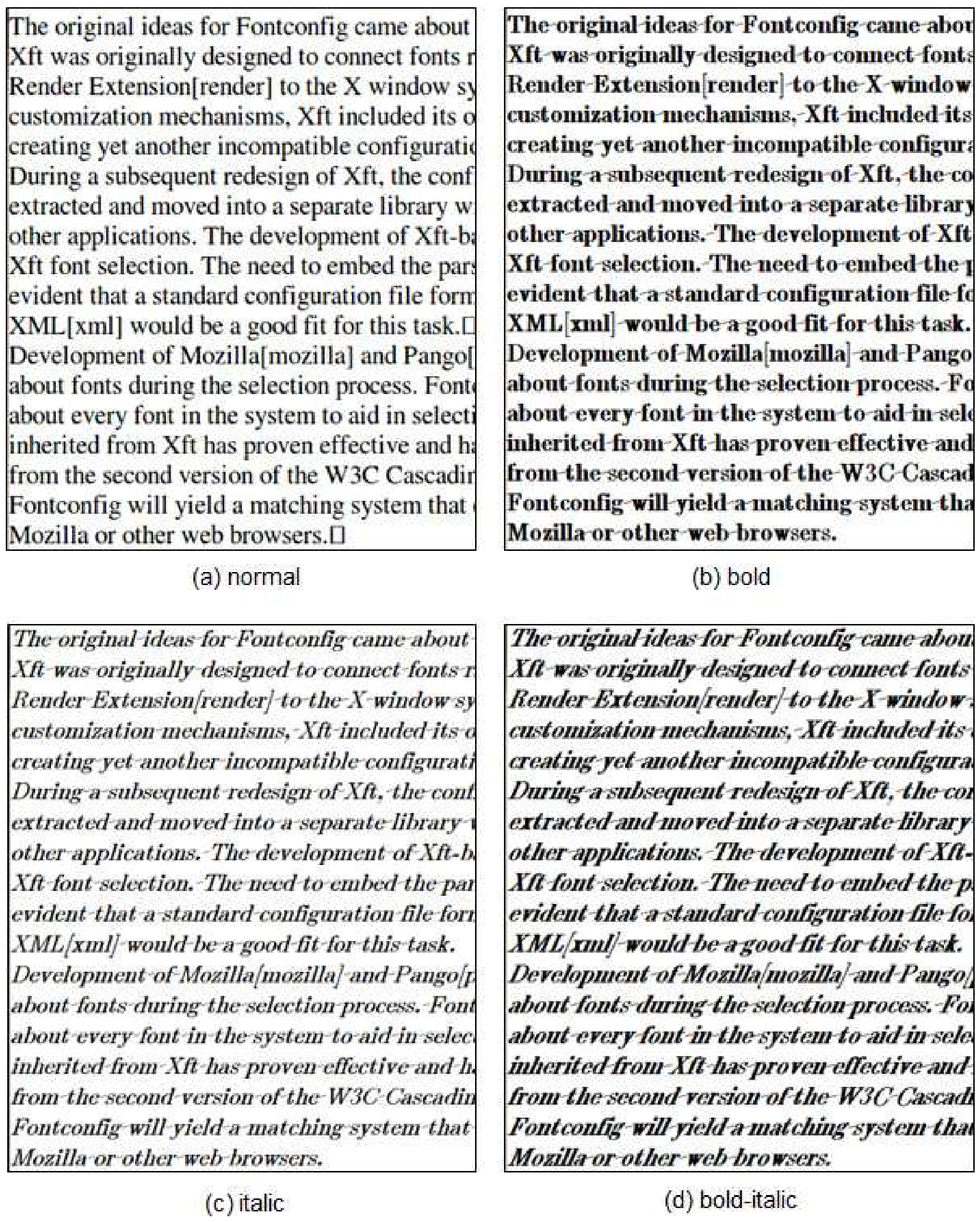
Figure 8: Displayed text in FreeSerif in the usual four styles

5 Conclusion

In this paper, the MFCONFIG module, enabling the direct use of METAFONT on Linux, is proposed. It is installed and used with the popular Freetype rasterizer. MFCONFIG is a plug-in module for the FONTCONFIG library The module supports a variety of the styled fonts that are generated from METAFONT by setting different parameters.

Existing digital font formats — notably the outline fonts of Type 1, TrueType, and OpenType — either do not typically allow users to change their styles, aside from font size scaling. From the experiments of the present study, it has been demonstrated that a variety of fonts can be directly generated on screen by applying different style parameters to a prototype METAFONT font using a Freetype rasterizer that is installed with MFCONFIG. Furthermore, the fonts could be seen within an average time of 90 ms, which is a barely noticeable duration.

The MFCONFIG module targets METAFONT fonts to be used with Freetype, a well-known rasterizer. MFCONFIG can be used effectively with



The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configuratio
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-ba
Xft font selection. The need to embed the pars
evident that a standard configuration file form
XML[xml] would be a good fit for this task.□
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Font
about every font in the system to aid in selecti
inherited from Xft has proven effective and ha
from the second version of the W3C Cascadin
Fontconfig will yield a matching system that
Mozilla or other web browsers.□

(a) normal

**The original ideas for Fontconfig came abou
Xft was originally designed to connect fonts
Render Extension[render] to the X window
customization mechanisms, Xft included its
creating yet another incompatible configura
During a subsequent redesign of Xft, the co
extracted and moved into a separate library
other applications. The development of Xft
Xft font selection. The need to embed the p
evident that a standard configuration file fo
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango
about fonts during the selection process. Fo
about every font in the system to aid in sele
inherited from Xft has proven effective and
from the second version of the W3C Cascad
Fontconfig will yield a matching system tha
Mozilla or other web browsers.**

(b) bold

*The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configurati
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-b
Xft font selection. The need to embed the par
evident that a standard configuration file forr
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango[p
about fonts during the selection process. Font
about every font in the system to aid in selec
inherited from Xft has proven effective and h
from the second version of the W3C Cascadin
Fontconfig will yield a matching system that
Mozilla or other web browsers.*

(c) italic

***The original ideas for Fontconfig came abou
Xft was originally designed to connect fonts
Render Extension[render] to the X window
customization mechanisms, Xft included its
creating yet another incompatible configura
During a subsequent redesign of Xft, the co
extracted and moved into a separate library
other applications. The development of Xft-
Xft font selection. The need to embed the p
evident that a standard configuration file fo
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Fo
about every font in the system to aid in sele
inherited from Xft has proven effective and
from the second version of the W3C Cascadi
Fontconfig will yield a matching system tha
Mozilla or other web browsers.***

(d) bold-italic

Figure 9: Displayed text in Computer Modern in the usual four styles

alphabet-based fonts, which are relatively simple and have a limited number of characters. However, there are only a few METAFONT fonts for various languages. It will likely take a longer time to process CJK METAFONT fonts, which have complicated shapes and more than several thousand phonemes. Further work will focus on these CJK METAFONT fonts to improve performance, and otherwise optimize the MFCONFIG module. In addition, this module will be experimented with as a font driver in the Freetype rasterizer.

Acknowledgements

This work was supported by an Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No.R-20160301-002987, Technology Development Project for Information, Communication, and Broadcast).

References

[1] Gyungjae Gwon, Minju Son, Geunho Jeong, and Jaeyoung Choi. Structural font generator using METAFONT for Korean and Chinese, 2016. In preparation.

[2] H. Kakugawa, M. Nishikimi, N. Takahashi, S. Tomura, and K. Handa. A general purpose font module for multilingual application programs. *Software: Practice and Experience*, 31(15):1487–1508, 2001. `dx.doi.org/10.1002/spe.424`.

[3] Hirotsugu Kakugawa. VFlib: A general font library that supports multiple font formats. *Cahiers GUTenberg*, iss. 28–29:211–222, March 1998. `cahiers.gutenberg.eu.org/cg-bin/article/CG_1998___28-29_211_0.pdf`.

[4] Donald E. Knuth. *Computers and Typesetting, Volume C: The METAFONTbook*. Addison-Wesley, 1986.

[5] Javier Rodríguez Laguna. Hóng-zì: A Chinese METAFONT. *TUGboat*, 26(2):125–128, 2005. `tug.org/TUGboat/tb26-2/laguna.pdf`.

[6] Keith Packard. The Xft font library: Architecture and users guide. *Proceedings of the 5th annual conference on Linux Showcase & Conference*, 2001. `keithp.com/~keithp/talks/xtc2001/paper/`.

[7] Keith Packard, Behdad Esfahbod, et al. Fontconfig. `fontconfig.org`.

[8] Y. Park. Current status of Hangul in the 21st century [in Korean]. ⟨*The T*⟩ *Type and Typography magazine*, vol. 7, August 2012. `www.typographyseoul.com/news/detail/222`.

[9] Matthew Skala. Tsukurimashou: A Japanese-language font meta-family. *TUGboat*, 34(3):269–278, 2013. `tug.org/TUGboat/tb34-3/tb108skala.pdf`.

[10] S. Song. Development of Korean Typography Industry [in Korean]. *Appreciating Korean Language*, 2013. `www.korean.go.kr/nkview/nklife/2013_3/23_0304.pdf`.

[11] Candy L.K. Yiu and Jim Binkley. Qin notation generator. *TUGboat*, 26(2):129–134, 2005. `tug.org/TUGboat/tb26-2/yiu.pdf`.

[12] Candy L.K. Yiu and Wai Wong. Chinese character synthesis using MetaPost. *TUGboat*, 24(1):85–93, 2003. `tug.org/TUGboat/tb24-1/yiu.pdf`.

◇ Jaeyoung Choi
Soongsil University, Seoul, Korea
`choi (at) ssu.ac.kr`

◇ Sungmin Kim
Soongsil University, Seoul, Korea
`sungmin.kim (at) ssu.ac.kr`

◇ Hojin Lee
Soongsil University, Seoul, Korea
`hojini (at) ssu.ac.kr`

◇ Geunho Jeong
Gensol Soft, Seoul, Korea
`ghjeong (at) gensolsoft.com`

Towards an operational (LA)TEX package supporting optical scaling of dynamic mathematical symbols

Abdelouahad Bayar

Abstract

In processing of digital documents containing mathematical formulas, the handling of dynamic mathematical symbols is still a difficult problem. A tool to compose mathematics should support the typing of variable-sized symbols taking care of optical scaling and supporting the quality of metal typesetting. Until now, there is no tool that allows these possibilities in a direct and operational way.

This contribution describes and puts into practice the basic steps to develop a (LA)TEX package directly based on a parameterized PostScript Type 3 font. This package will present to (LA)TEX end-users a tool to compute in the usual way mathematical formulas consisting of dynamic mathematical symbols while taking into account optical scaling. Formatting (LA)TEX documents using this package is achieved without requirements of special environments or external programs.

We find that the concept of using parameterized Type 3 fonts directly with (LA)TEX commands can give an accurate and straightforward way to manage dynamic graphics in documents formatted under (LA)TEX, e.g., logo graphics.

Keywords: (LA)TEX, PostScript Type 3, dynamic mathematical symbols, optical scaling

1 The problem

1.1 Class of mathematical symbols

Mathematical formulas are written using static and/or variable-sized symbols. When using a font at a given size, the dimension and shape of a static symbol remain unchanged in the whole document; α and $+$ are good examples to represent this class. A variable-sized symbol varies in terms of size and sometimes shape from one context to another in the same document. As an example, we can cite the width of the hat symbols indicating angles: $\widehat{A}$ and $\widehat{AOB}$. The managing of variable-sized symbols, which we will sometimes call dynamic mathematical symbols,[1] is still a significant challenge in the area of document processing (see later).

[1] The term "dynamic mathematical symbol" encompasses variable-sized symbols and also symbols defined in dynamic fonts. Dynamic fonts are fonts in which printed character graphics are defined in each instantiation at the time of printing and not in the definition of the font.

1.2 Optical scaling

This is a concept used to handle different sizes of the same font. It is in contrast to linear scaling. To produce, say, size 48 using linear scaling, size 12 is merely magnified four times. This is not the case with optical scaling; rather, the building of the character is done by taking into account the eye of the reader. More details on the optical scaling concept are found in [3, 8, 9].

When we consider humans or trees, for example, we can see obviously that they do not grow according to a linear model. The human eye is naturally attuned to the point of view represented by art. It would be better to talk about *natural scaling* than optical scaling since the first is more general than the second. (By the way, the confusion between "optical scaling" and "optical scale" introduced by Harry Carter in typefounding [7] will not happen.)

1.3 Metal vs. digital typesetting and optical scaling

In old books, especially those in mathematics, mathematical formulas were typeset using optical scaling. We give an example of a formula taken from [11] exhibiting the metal braces. It is clear that these braces are not related with a linear scaling (see Figure 1). Optical scaling was not difficult to achieve in metal typesetting since symbols were processed in their final sizes. However, the support of optical scaling is not easy in automatic systems when computing symbols or fonts, especially in the case of digital typesetting. More information on this point is in [3, 16].

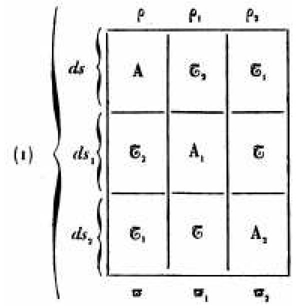


Figure 1: Braces in metal typesetting

1.4 Existing work

Dynamic characters, especially dynamic mathematical symbols, are omnipresent in scientific documents. So, a good application to process scientific documents must include all required means to support dynamic characteristics. In the last four decades, a number of tools have been developed which support dynamism in different ways. We can cite in this case native TEX [12] and LATEX [13] to support mathematical variable-sized symbols. One important tool to indicate is CurExt [14], a (LA)TEX package. With this package, it is possible to typeset mathematical formulas consisting of variable-sized mathematical symbols, particularly Arabic ones. In the same way, CurExt allows one to write Arabic text taking into account the kashida concept. We have to note that the kashida makes manifest an important phenomenon of dynamism in Arabic text typesetting. Detailed information about the kashida and justification of Arabic texts can be found in [10]. It is very important to consider the work accomplished in [2, 3]. This consisted of the design of the "math-fly" font, a PostScript Type 3 font, to supply dynamic mathematical symbols taking into account optical scaling. The particular property of this work, in comparison with the preceding, is that it is used neither under nor with (LA)TEX.

CurExt as a package to extend TEX's capabilities in handling variable-sized symbols has some difficulties in processing mathematical formulas containing more than two (matched) dynamic symbols [14]. So, it does not offer adequate support to manage the general case of mathematical formulas.

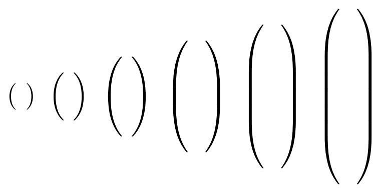


Figure 2: Stretches of parentheses in TEX

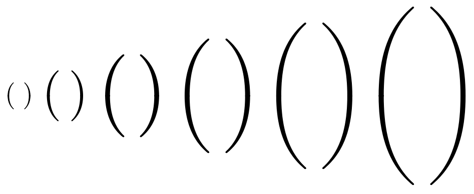


Figure 3: Stretches of parentheses via our dynamic PostScript

As for TEX, readers here likely already know that it supports the composition of mathematical formulas with multiple variable-sized symbols. Optical scaling is not very well supported, however, since the thickness cannot be changed after a certain point (see Figure 2), while the present package can do so continuously (Figure 3). Furthermore, due to the non-dynamic properties of Metafont, dynamic variable-sized symbols at big sizes do not change in

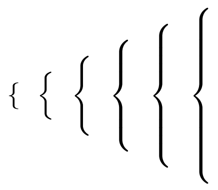


Figure 4: Stretches of left braces in TEX

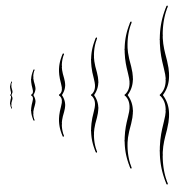


Figure 5: Stretches of left braces via our dynamic PostScript

shape (Figures 2 and 4), but merely have straight parts extended. The result does not look like the traditional typesetting shown in Figure 1. Figure 5 shows the results for braces with the present package, with the shape changing in a natural way, in addition to the increasing thickness.

Regarding optical scaling, we observe that any one of these tools is completely operational. *The problem of typesetting mathematical formulas with good quality respecting optical scaling is still challenging.*

In the following, the paper follows this plan: the second section presents what is required for handling dynamic mathematical symbols, taking into account optical scaling. In the third section, the practical and operational way to design the system is given. The next section is dedicated to describing and comparing the implementation of the package under different TEX tools. The paper ends with conclusions and perspectives.

2 Requirements to handle dynamic mathematical symbols taking care of optical scaling

To realize a convenient tool to compose mathematical formulas based on dynamic mathematical symbols with respect to optical scaling, it is required to subdivide the study into two parts. The first part concerns defining a *font of symbols* whereas the second refers to the *way to use this font* to produce mathematical formulas. In all cases, we must not neglect the fact that the tool has to assist the (end-)user in producing good mathematical formulas in a direct and straightforward way.

We know that fonts are classified in two groups: static fonts and dynamic fonts. We give briefly and accurately the difference between the two classes. In static fonts, shapes (the graphic to print) of characters are generated and finalized before printing. However, in dynamic fonts, characters take their printing characteristics at printing time. More details and examples for comparison are found in [1]. A suitable font to deal with variable-sized symbols must be dynamic; moreover, it must have the following features:

- The language to implement programs encoding dynamic symbols must provide more flexibility in parameterizing symbols. Also, it must have the ability to receive values from outside the font to instantiate parameters and so generate the shape to print.
- The interaction between the document processing system and the font language must be well defined and directly used in the document processing task.

(LA)TEX, as a text formatting system, provides natively an interface to fonts encoded in the Metafont language. This is achieved principally via tfm files. Nevertheless, Metafont does not allow manipulating dynamism at printing time. (LA)TEX can also use other kinds of fonts like PostScript Type 1, TrueType, or the hybrid of these two, OpenType. These fonts are referenced by TEX as if they were virtually Metafont fonts. So the use of these fonts does not add to either a means to supply real dynamism. It is also very interesting to cite XƎTEX and XƎLATEX. These are extensions to TEX and LATEX in order to work directly with OpenType fonts (also Type 1 and TrueType) without use of any intermediate mapping files. Even with this capacity, XƎTEX and XƎLATEX do not support a complete dynamism due to the limited interaction interface between the TEX engine and the font. Furthermore, OpenType supports only a semi-dynamism or a discrete dynamism.

PostScript Type 3 fonts have some particular features:

- They use the full PostScript language. This means that the specification of fonts can use all operators and constructors existing in the PostScript language, especially local and global variables. Variables are the means to communicate new characteristics to the procedure encoding dynamic symbols.
- The concept of caching the character bitmaps (used in Type 1) can be deactivated via replacing `setcachedevice` with `setcharwidth`. This implies that each time a given character is to be printed, its bitmap will be fully computed. Consequently, the model supporting variability (optical scaling) can take new values and states.

Using PostScript Type 3 does lose some important abilities like fast printing, improvement via hints, and handling by Adobe Type Manager (ATM). But the support of dynamic mathematical symbols taking care of optical scaling may outweigh the disadvantages. Also, nowadays, the computers inside printers are so fast and so large that the time required to move the paper inside printers dominates the printing speed. Moreover, the printer industry regularly makes significant improvements in resolution. Then, the efficiency benefits such as caching and hints are gone.

PostScript Type 3 fonts can be used by TEX in the same way as Type 1. In this case, we cannot take advantage of the benefit of dynamic specification in PostScript. In [2], the authors stated that a parameterized Type 3 font could not be fully used by formatters (editors) such as TEX or others without modifications to the way they call formula symbols. For this reason, they chose to check their font in the Grif project. In our case, the PostScript Type 3 font will be inserted directly into the (LA)TEX source of the document to be formatted by means of the `\special` macro. Of course, the way to process dynamic mathematical symbols in mathematical formulas will be reviewed. The details of the concept are given in the following.

3 The design of a practical and operational system

3.1 General package layout

The development of a package able to use directly a PostScript Type 3 font is based on the existing possibilities of interaction between (LA)TEX and PostScript. This is done using the command `\special` via the `dvips` driver to translate dvi files to PostScript ones [15]. More precisely, we use these methods to include literal PostScript in TEX documents to work with the PostScript Type 3 font. A summarized design of the package is given below.

- The PostScript Type 3 font supporting dynamic mathematical symbols and all useful procedures is defined in the package as a literal header, a '`!`' `\special`. This is mandatory since the font will be used later when including other PostScript codes to show PostScript dynamic symbols. Our Type 3 font is named "`dynMath`". The command is:
  `\special{! ...dynMath specification...}`.

The font implements the mathematical symbols which will support curvilinear stretching depending on the values of two global variables

h and w.[2] In the font, the stretching model allows symbols to stretch in height (depth) and width depending on the values of h and w while keeping the same thickness. We note that the scaling is not linear. It is a semi-optical scaling since the thickness is not affected. It is done in macros we will define, such as `\meLeft` for example (see next).

- The principal macro for handling mathematical formulas and thus dealing with inclusion of the PostScript dynamic mathematical symbols is defined in the package. Its skeleton is:
  `\def\meLeft#1#2\meRight#3{...}`.
  `#1`: the left delimiter,
  `#2`: the formula to delimit, and
  `#3`: the right delimiter.

  This macro manages the dynamic mathematical symbols which are delimiters. Other dynamic symbols (e.g., the radical sign), are defined in separate macros or in some cases existing macros will be redefined. About the delimiters, we chose to finalize at the moment the implementation of only two symbols, namely parentheses and braces. In reality, these two groups of symbols are an *adequate representation of curved symbols*. Parentheses have simple curved shapes, whereas the braces have curved shapes with inflections. We notice that variable-sized symbols that are simple combinations of lines are simple to implement in the font.

- In the `\meLeft` macro:
  1. The dimensions (width, height and depth) of the formula are computed. Let h_f, w_f and d_f be these dimensions respectively.
  2. Depending on h_f, w_f, d_f and the left delimiter symbol, `\meLeft` determines the stretching amounts of h and w. Then, the corresponding size `fs` at which the font `dynMath` will be used to output the left symbol is calculated.
  3. The dimensions of the left symbol `symHeight`, `symWidth`, and `symDepth` taking into account the PostScript font `fs` are determined.
  4. In an `\hbox` of dimensions `symHeight`, `symWidth`, `symDepth`, the left symbol is written using literal PostScript:
     ```
     ... \special{" ...
     /fs ... store
     /h ... store
     /w ... store
     /dynMath findfont fs scalefont setfont
     ⟨code of the symbol⟩ show
     }
     ```
  5. The formula is written.
  6. The steps from the second to the fourth are applied for the right delimiter. Frequently, `symHeight`, `symWidth` and `symDepth` remain unchanged.

[2] We have chosen to use simple names like h and w to make the description of the equations easy in this paper. In actual code, meaningful names will be used; for example, h and w will be replaced by `verticalStretch` and `horizontalStretch` respectively.

3.2 The design of the `dynMath` font

The font `dynMath` is based on the font `cmex10.mf`. The simple difference is that a symbol appears in `dynMath` only once as opposed to the multiple versions in `cmex10.mf`. For example, the left parenthesis is encoded in cells numbered 0, 16, 18, 32. The stretchable parenthesis is built upon the characters numbered 48, 66 and 64. However, in `dynMath`, only one parameterized parenthesis is located in the font at the order 0. For some particular values of the parameters, we get the parenthesis with expected characteristics. As said before, only (,), { and } with code numbers 0, 1, 8 and 9 respectively are encoded at the moment.

We used the existing font `cmr10.mf` to get the nuclei of left and right parentheses which we parameterized applying a mathematical model. Notice that we did not use `cmex10.mf`. This is because the small parenthesis in `cmex10.mf` is bigger than the normal parenthesis in text. For the left and right braces, we saw that none of the Metafont fonts supplied with TeX distributions provide braces that look like the metal ones. So, we designed them. The command applied to generate the basic encoding of the parenthesis through `cmr10.mf`:

```
mpost '&mfplain \mode=localfont; \
  mag=100.375; input cmr10.mf'
```

To explain the global concepts for designing the font, we use the parenthesis as an example. The same process is applied to the other symbols. Without loss of generality, we show only the top part of the left parenthesis (with respect to the mathematical axis).

Applying MetaPost to `cmr10.mf`, we get the encoding of the left parenthesis. Its top left part is shown in Figure 6. It is defined based on two Bézier curves linked by two line segments at the top and bottom. To get a parenthesis symbol that is able to stretch when needed, the two Bézier curves are multi-decomposed using the generalized algorithm of refinement [4].

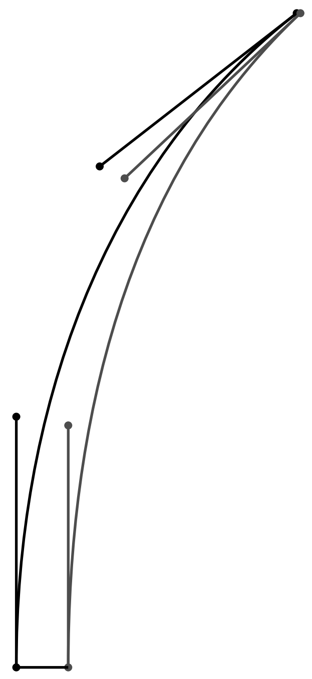


Figure 6: Top part of left parenthesis — initial encoding at size 500

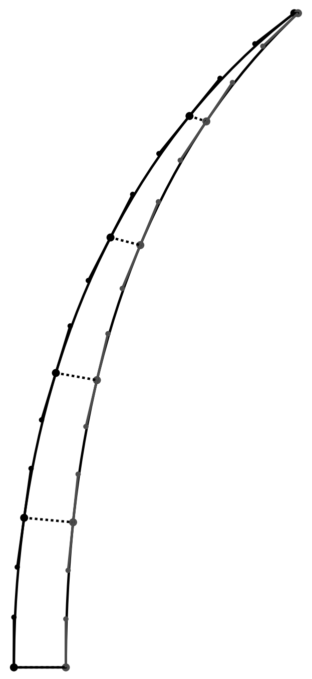


Figure 7: Decomposed top part of left parenthesis (parameterized curves) at size 500 without stretching

To explain in detail, we consider a refinement of the fifth order. In Figure 7, the curves are decomposed according to the decomposition parameters $1/5$, $1/4$, $1/3$ and $1/2$. Every Bézier curve of the encoding appears as a concatenation of five sub-curves. The latter are then parameterized taking into account the variables h and w such that when the stretching amounts are equal to zero the shape is identical to the initial one illustrated in Figure 6.

Figure 8 shows an example of stretching at size 500. h and w take the values 250 and 83.33 PostScript points respectively ($83.33 \approx 250/3$). We notice that the initial and stretched versions of the half symbol differ in height and width but they have the same thickness. To emphasize this fact, let us consider the line segments joining the extreme control points of the left and right sub-curves. Each segment in Figure 7 and its corresponding segment in Figure 8 are parallel and have the same length. It is obvious that the scaling is not linear. It does not support a true optical scaling either since the thickness remains unchanged. In the PostScript font, only a semi-optical scaling is defined. The optical scaling is completed in the TEX package. The way to parameterize the control points in order to support this semi-optical scaling requires obedience to a strict mathematical model. (We do not present the mathematical concepts here because this is outside the objective of this paper. Nevertheless, we cite the basics of the development.) The curves are parameterized based on the mathematical model which insures that stretched and initial curves have the same geometric and similarity characteristics. In our PostScript font, the left part of the parentheses has undergone a Bézier refinement of order 15, whereas in the example, the fifth order was enough because the amounts of stretching are not big.

3.3 Optical scaling support

In this section, we present how the optical scaling is supported by the package. The best way to describe the concepts is via the delimiters, particularly balanced ones such as parentheses and braces. This cannot be done without enumerating the principal characteristics of a mathematical formula.

We consider an abstract mathematical formula to present the characteristics. It consists of a box with some height, depth and width (which is not relevant to this paper). Figures 9 and 10 show the two cases of mathematical formulas, namely when the formula is high or deep, respectively. Moreover, they introduce some characteristic variables of formulas:

- f_h: height of formula from the baseline.
- f_d: depth of formula from the baseline.

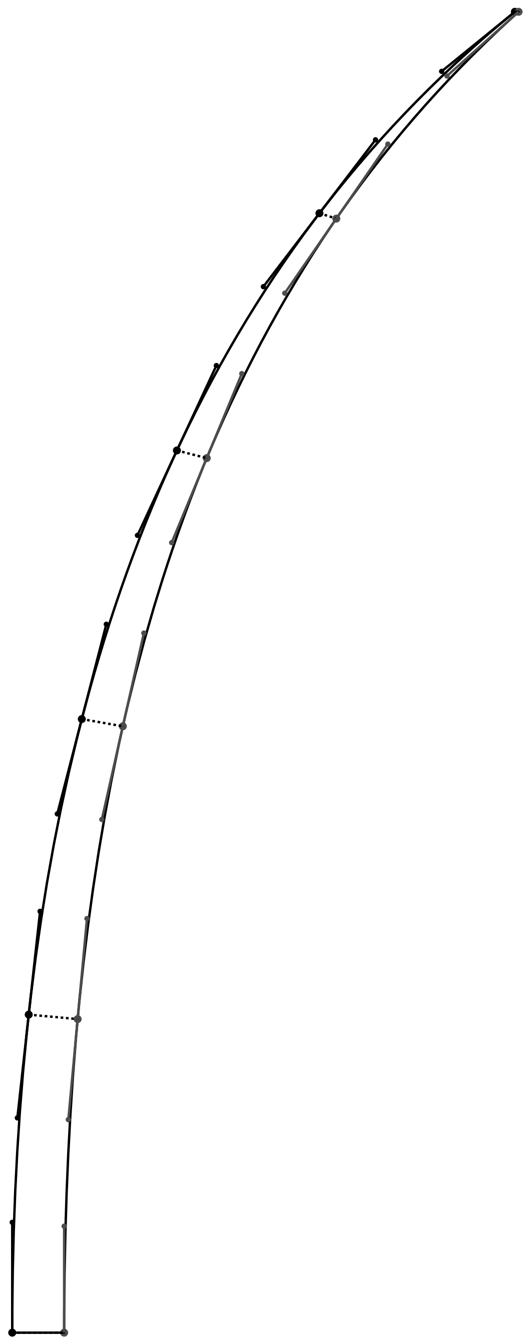


Figure 8: Decomposed top part of left parenthesis (parameterized curves) at size 500, stretched 250 vertically and 83.33 horizontally

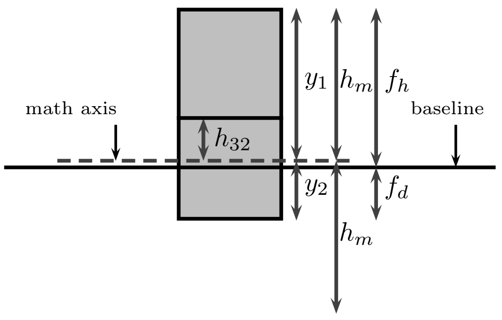


Figure 9: Abstract high mathematical formula

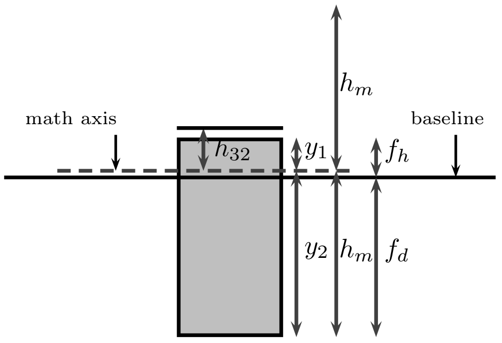


Figure 10: Abstract deep mathematical formula

- y_1: mathematical height of the formula, measured from the mathematical axis to the top of the formula.
- y_2: mathematical depth of the formula, measured from the mathematical axis to the bottom of the formula.
- h_m: mathematical balanced height (depth) of the (balanced) formula. We have that $h_m = \max(y_1, y_2)$.
- h_{32}: the mathematical height of parenthesis at size 32 (corresponding to the height h^p_{32} in the PostScript `dynMath`; of course the value manipulated in the package considers the relation between pt and bp). h_{32} is a reference in processing the optical scaling (see later).

We note that a TEX variable v_n is the value in TEX units corresponding to v^p_n in PostScript units. For example, h_{32} is the height in pt corresponding to h^p_{32} being the height in PostScript of the dynamic symbol at size 32. We have formally $v_n = 1.00375 \times v^p_n$.

One part of optical scaling, as previously stated, is supported by the PostScript Type 3 font whereas the other part is a direct job of the TEX package. After a study of the fonts generated from `cmr10.mf` and `cmex10.mf` via MetaPost, we noticed that the standalone parentheses symbols (the symbol consisting of one character) obtained from `cmex10.mf` are in some way a linear scaling of the standalone parentheses supplied by cmr10.mf. The dimensions of the big parenthesis in cmex10.mf is approximately three times the size of the ones relative to the parenthesis in cmr10.mf. Consequently, the optical scaling is handled in two different ways depending on the value of h_m. The first case deals with the values of h_m less than or equal to h_{32} whereas the second takes care of values greater strictly than h_{32}.

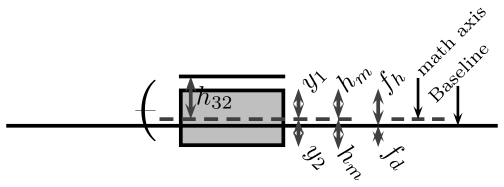


Figure 11: Abstract mathematical formula with $h_m \leq h_{32}$ and non-aligned math axis

When the mathematical height h_m is less than or equal to h_{32}, the optical scaling is treated simply as a linear scaling. Figure 11 illustrates this case. Let *fs* be the PostScript font size in which the height of half of the left parenthesis (taken as an example) equals h_m. Then the font dynMath is set to *fs* and the delimiter is written in a \special.

We can see that when the delimiter is introduced, the mathematical axis of the formula and that of the delimiter are not aligned. This is normal because the "TeX size" in which the formula (10, 11, ...) is written will usually be different from the delimiter size. In the \special macro, a shifting operation is accomplished before writing the PostScript delimiter, as shown here in Figure 12:

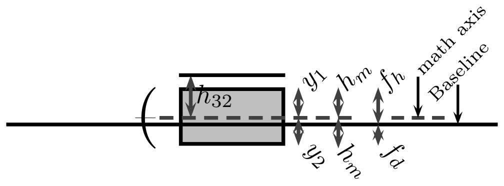


Figure 12: Abstract mathematical formula with $h_m \leq h_{32}$ and aligned math axis

In the case where h_m is strictly greater than h_{32}, the PostScript font size is managed differently. The mathematical model adopted to formalize stretching of mathematical symbols supplied to us has a maximal vertical stretching of 32700 bp. As the size 32 is a threshold to handle the symbol stretching, we consider the maximal amount allowed in stretching the half of a parenthesis to be h^p_{max}. The superscript p relates to the PostScript context. We have:

$$h^p_{max} = \frac{32700 \times 32}{1000} = 1190.4 \qquad (1)$$

Let h_{max} be the equivalent amount in TeX points; then we have

$$h_{max} = 1.00375 \times h^p_{max}\text{pt} = 1194.864\text{pt} \qquad (2)$$

In the following, we give needed dimensions only in TeX points since all calculations are done by TeX. The first step in processing is to determine the size that will be used to compute the delimiter. The PostScript font dynMath provides the possibility to extend symbols without affecting the thickness. This is what remains to support optical scaling. Let e be the thickness variable (in TeX units). e is calculated as a function of the mathematical height of the formula to be delimited. This is given in Equation 3:

$$e\,(h_m) = c_1 h_m + c_0 \qquad (3)$$

where c_1 and c_0 are constants satisfying the following requirements:

- $e\,(h_{32}) = e_{32}$
- $e\,(h_{max}) = \lambda \times e_{32}$
- e_{32}: thickness of the dynamic symbol at size 32.
- λ: a scaling constant factor.[3] We notice that λ is not a global value for the font. It depends on the dynamic symbol.

For λ, e_{1000} and e_{32} known, we can determine the size *fs* for which the thickness of the symbol is e. Using *fs*, we can produce the corresponding math height of the symbol h_{fs}. The formulas giving these variables are in Equations 4 and 5.

$$fs = \frac{1003.75}{e_{1000}} e \qquad (4)$$

$$h_{fs} = \frac{h_{1000}}{1003.75} fs \qquad (5)$$

[3] For parenthesis and brace, fulfillment is reached with $\lambda = 3.236pt$. 3.236 equals 2×1.618. 1.618 is the golden ratio.

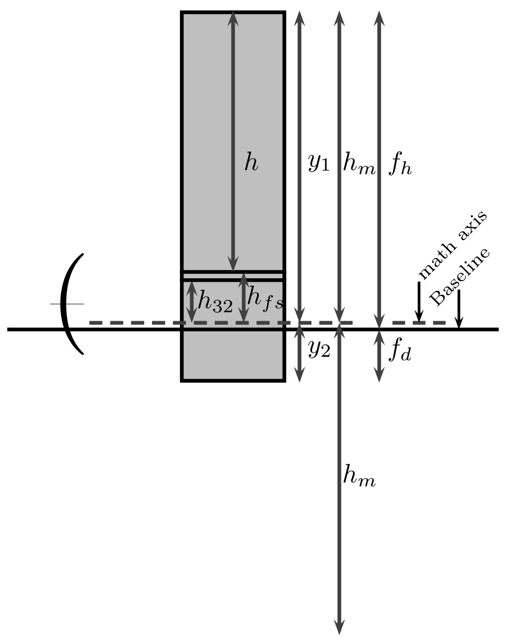


Figure 13: Abstract mathematical formula with $h_m > h_{32}$, non-aligned math axis and non-stretched parenthesis

As an important remark, we can see easily that $h_{32} < h_{fs} < h_m$. Once h_{fs} is processed, the amount of vertical stretching h can be determined letting $h = h_m - h_{fs}$. (See Figure 13 above.)

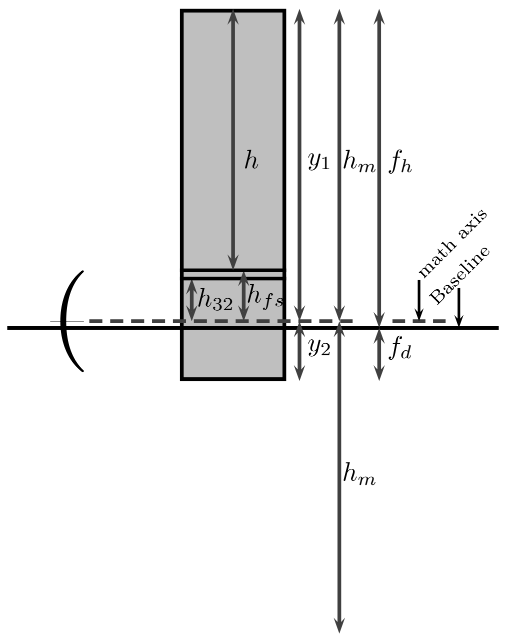


Figure 14: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and non-stretched parenthesis

The dynamic symbol is written, especially the left parenthesis as shown in Figure 13, in `\special` macros using the font fs^p ($fs^p = 0.9962 \times fs$) as the size. We remark that for the case where $h \leq h_{32}$ that the math axis of the formula and that of the parenthesis do not coincide (see Figure 13). So, a shift transformation is applied before writing the symbol in PostScript (see Figure 14).

The horizontal stretching amount w may take values depending on h. In the case of the left parenthesis, w describes the interval $[0, {}^{h}\!/_{3}]$. In Figure 15, the parenthesis at the fs size is stretched by h vertically and by $w = {}^{h}\!/_{3}$ horizontally.

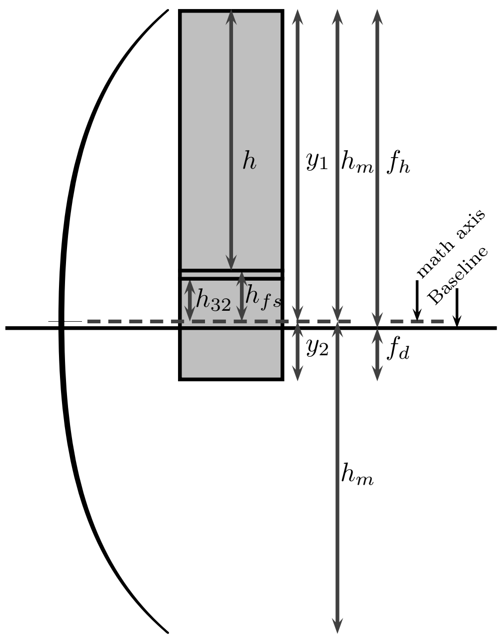


Figure 15: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and stretched parenthesis

4 Implementation

As we said, the system to typeset mathematical formulas based on dynamic variable-sized symbols is composed into a PostScript Type 3 font and a set of TEX macros. Until now, the package is presented as a simple TEX source file. It contains only TEX macros that are recognized also by LATEX. So the package operates with both TEX and LATEX. At the same time, it is a good nucleus from which to develop a LATEX package. One of the important steps in the process of executing the macros managing variable-sized symbols is the catching of the current mathematical style. It is a mandatory task in order to get the right dimensions of the mathematical formula and determine the true sizes of the dynamic mathematical symbol. First, we have developed a package that can operate in all TeX programs. The determination of the current math style imposes the use of complete recursion, i.e., recursion in macro definitions and execution. So the package is very slow and more demanding of memory. To solve this problem, we resorted to the use of Lua(LA)TeX since they supply `\mathstyle` (`\luatexmathstyle`) which permits recognition of the mathematical style on the fly, greatly reducing the time of processing and the use of memory. Of course, since a PostScript Type 3 font is used in conjunction with the TeX package, then source documents are formatted via `dviluatex` (`dvilualatex`) and `dvips` commands.

5 Conclusions

We have developed a mini-package for TeX offering support of variable-sized mathematical symbols with respect to optical scaling. As an illustration, we implemented only two symbols: the parenthesis and the brace. But the support of these two symbols demonstrates the feasibility as well as the possibility to produce scientific documents with the quality of metal typesetting. In the future, we will finish, at both the font and package levels, the support of all the rest of the dynamic mathematical symbols. At the same time, options relating to the printing quality will be added to the final package. A more important task will concern the optical scaling with consideration of an artistic view point. Indeed, the relationship between the values of horizontal, vertical stretching and thickness will be studied taking into account artistic satisfaction.

References

[1] Jacques André, B. Borghi, “Dynamic Fonts”, *PostScript Language Journal*, Vol. 2, no. 3, pp. 4–6, 1990. `http://jacques-andre.fr/japublis/fontesdyn.pdf`

[2] Jacques André, Irène Vatton, *Contextual Typesetting of Mathematical Symbols Taking Care of Optical Scaling*, Technical report No. 1972, INRIA, October 1993.

[3] Jacques André, Irène Vatton, “Dynamic Optical Scaling and Variable-sized Characters”, *Electronic Publishing*, Vol. 7, No. 4, pp. 231–250, December 1994. `http://jacques-andre.fr/japublis/opticalscaling.pdf`

[4] Brian A. Barsky, *Arbitrary Subdivision of Bézier Curves*, Technical Report UCB.CSD 85/265, Computer Science Division, University of California, 1985. `http://www.eecs.berkeley.edu/Pubs/TechRpts/1986/CSD-86-265.pdf`

[5] M.J.E. Benatia, M. Elyaakoubi, A. Lazrek, “Arabic Text Justification”, TUG 2006 conference, Marrakesh, Morocco. *TUGboat* 27:2, pp. 137–146, 2006. `http://tug.org/TUGboat/tb27-2/tb87benatia.pdf`

[6] Daniel M. Berry, “Stretching Letter and Slanted-Baseline Formatting for Arabic, Hebrew and Persian with ditroff/ffortid and Dynamic PostScript Fonts”, *Software—Practice and Experience*, vol. 29, no. 15, pp. 1417–1457, 1999. `https://cs.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/keshide.paper.pdf`

[7] Harry Carter. “The Optical Scale in Typefounding”, *Typography*, No. 4, pp. 2–6, Autumn, 1937. `https://issuu.com/letterror/docs/harry_carter_optical_scale_in_typefounding`

[8] Circuitous Root, “From the Optical Scale to Optical Scaling”, 2016. `http://www.circuitousroot.com/artifice/letters/press/typemaking/mats/optical/index.html`

[9] Circuitous Root, “Clubs and Cults Revisiting the Concept of ‘Typeface’ and the Optical Scale in Typefounding”, 2016. `http://www.circuitousroot.com/artifice/letters/press/typemaking/making-matrices/terms/logical-grouping/clubs-and-cults/index.html`

[10] Mohamed Elyaakoubi, Azzeddine Lazrek, “Justify just or just justify”, Journal of Electronic Publishing, Volume 13, Number 1, 2010. `http://dx.doi.org/10.3998/3336451.0013.105`

[11] G. Lamé, “Leçons sur les coordonnées curvilignes et leurs diverses applications”, Imprimerie de Mallet Bachelier, Paris – Rue du Jardinet 12, 1859. `https://archive.org/details/bub_gb_jfgxNexuunYC`

[12] D.E. Knuth, *The TeXbook, Computers and Typesetting*, Vol. A, Reading, MA: Addison-Wesley, 1984.

[13] Leslie Lamport, *LaTeX — A Document Preparation System*, Reading, MA: Addison Wesley, 1985.

[14] Azzeddine Lazrek, “CurExt, Typesetting variable-sized curved symbols”, EuroTeX 2003 conference, Brest, France. *TUGboat* 24:3, pp. 323–327, 2003. `http://tug.org/TUGboat/tb24-3/lazrek.pdf`

[15] Tomas Rokicki, “Dvips: A DVI-to-PostScript translator”, version 5.996, 2016. `http://tug.org/dvips`

[16] Richard Southall, “Character description techniques in type manufacture”, in Raster Imaging and Digital Typography II, eds., Robert A. Morris and Jacques André, pp. 16–27, Cambridge, UK, October 1991.

◇ Abdelouahad Bayar
Cadi Ayyad University
Ecole Supérieure de Technologie de Safi
[High college of technology of Safi]
Morocco
`a.bayar (at) uca.ma`

A LaTeX reference manual

Jim Hefferon

Abstract

The LaTeX Reference Manual summarizes the features of LaTeX 2$_\varepsilon$. It can be a valuable resource for authors using LaTeX and deserves to be better known.

1 Introduction

The LaTeX Reference Manual, `latexrefman`, aims to provide a freely available document summarizing the commands and environments of LaTeX 2$_\varepsilon$. It is unofficial, not associated with the LaTeX project.

You can see a current copy, in a variety of formats, either at the project home page[1] or on CTAN[2].

This work brings together an array of LaTeX sources and documentation and organizes that material into a reference format. Its base language is English; presently, there are French and Spanish translations.

This project deserves to be better known, among both potential users and potential contributors.

2 For potential users

`latexrefman` is a resource that can be valuable to LaTeX authors while they are writing.

2.1 Why more documentation?

There are many works on LaTeX: tutorial and advanced, online and on paper, in many languages and at many levels of sophistication.

The chief distinction of `latexrefman` is that it is a reference manual. If, for instance, you can't remember the specifics of the syntax of a command then you can go directly to that command's entry. There will be a complete description, including all of the salient technical points as well as an overview, and hyperlinks to related entries.

An example of the difference between a reference and other works is that here each entry assumes whatever level of reader sophistication is needed to cover the topic. For instance, an entry might specify that the argument of a command is typeset in LR mode, which a tutorial is likely not to state.

In addition, `latexrefman` is organized by command. For instance, the major LaTeX environments each get a separate entry.

`latexrefman` now covers the commonly-used commands and environments. The entries work to make fine distinctions clear. They also give the values of various parameters in the standard LaTeX classes. It is online so it can be easily accessed and so that searches and hyperlinks are also easy.

In short, the document is written with a focus on being useful to a working LaTeX author who is typing, who runs across an issue, and who wants a convenient way to see all the information needed to resolve the issue, presented in one place.

[1] `http://home.gna.org/latexrefman/`

[2] `https://ctan.org/pkg/latex2e-help-texinfo`

2.2 Sources

The information in this manual is available from other LaTeX sources and documentation, but is scattered. Besides the ultimate reference of the LaTeX 2$_\varepsilon$ source code, and books such as [1] and [2], there are also many reputable online sources including the *Comprehensive Symbols List*[3] and the *Users Guide for amsmath*[4]. The work of this project is to bring these diverse sources together and synthesize the information into a reference form.

This project has an advantage over reference manuals for other subjects: the existence of online forums. The TeX family has a long history and is also blessed with a helpful community so there are a number of long-lived forums, including the Usenet group `comp.text.tex`,[5] the mailing list `texhax`,[6] the TeX-LaTeX Stack Exchange,[7] and the Reddit subgroup `/r/LaTeX`.[8] This great body of material not only provides authors of a reference with answers that may be hard to find elsewhere but, just as importantly, provides those authors with questions showing what gives users trouble. If online research reveals a question about a topic that has puzzled LaTeX users over a long time then we can increase the value of the reference by including explanation or examples specifically addressing that question.

2.3 Coverage

At present, `latexrefman` covers the most-used commands and environments of core LaTeX 2$_\varepsilon$.

We do not plan to ever cover a broad, let alone complete, range of packages from CTAN. Instead, the plan is to exhaustively cover all of core LaTeX 2$_\varepsilon$. That goal remains rather distant; thus, additional project contributors would be most welcome.

[3] `https://ctan.org/pkg/comprehensive/`

[4] `ftp://ftp.ams.org/pub/tex/doc/amsmath/amsldoc.pdf`

[5] `https://groups.google.com/forum/#!forum/comp.text.tex`

[6] `http://lists.tug.org/texhax`

[7] `http://tex.stackexchange.com/`

[8] `http://www.reddit.com/r/latex`

3 For potential contributors

This is a project where a person looking for a way to give back to the community can make a real contribution without a lot of initial effort. You can start off small by finding some improvement on an existing entry, or a missing one, and submitting a suggestion or a patch.

You could submit that improvement via the mailing list. You can also download the document source. You can reach both at the project's home page.[9]

The document source is in Texinfo.[10] To give a feel for this, here are parts of the source of the entry on LaTeX's *quotation* and *quote* environments.

It begins with the syntax of the two:

```
Synopsis:

@example
\begin@{quotation@}
@var{text}
\end@{quotation@}
@end example

or

@example
\begin@{quote@}
@var{text}
\end@{quote@}
@end example
```

The most obvious difference from writing in LaTeX is that the at sign `@` takes the place of backslash as the escape character, including the escaping used in `@example` and `@end` but also including escaping the braces as `@{` and `@}`. There is (intentionally) little in the way of macros so writing is relatively straight-ahead — a person used to LaTeX does not need to ramp up much to start working with the source.

Here is a little more, later in the same entry:

```
To compare the two: in the
@code{quotation} environment, paragraphs
are indented by 1.5@dmn{em} and the space
between paragraphs is small,
@code{0pt plus 1pt}.  In the @code{quote}
environment, paragraphs are not indented
and there is vertical space between paragraphs
(it is the rubber length @code{\parsep}).
```

We see here that, as discussed earlier, there is a focus on the working LaTeX author. The body addresses the question of when to choose one environment or the other.

In addition, you can see that the reference manual strives to give precise default values, as actually defined, rather than vague circumlotions. These numbers come from the LaTeX 2$_\varepsilon$ source files. So this is an example of a way that a person can make a useful contribution back to the community: spend a little time tracking down values that are not yet specified.

Finally, that entry closes with an example that is short but is also cut-and-pasteable. That is, this example is designed to be one that would get our hypothesized LaTeX author started.

```
@example
\begin@{quotation@}
\it Four score and seven years ago
  ... shall not perish from the earth.
\hspace@{1em plus 1fill@}---Abraham Lincoln
\end@{quotation@}
@end example
```

There are other things in the source file not shown here, notably cross-reference information. In addition, the source is in a Subversion repository. But both are easy to get used to.

3.1 History of contributions

This project has been around, in various forms, for a long time. George Greenwade started it as help files for VMS. It was updated for LaTeX 2.09 by Stephen Gilmore and for LaTeX 2$_\varepsilon$ by Torsten Martinsen. Today, active contributors are Vincent Belaïche, Karl Berry, and Jim Hefferon. Vincent also maintains the French translation and has made some updates to the Spanish translation, but reports that Spanish needs a new maintainer. Translations to more languages would be most welcome.

4 Summary

The LaTeX Reference Manual aims to provide a freely available document summarizing the features of LaTeX 2$_\varepsilon$. In its current state it can be a useful resource for LaTeX authors. Give it a try!

It plans eventually to cover all of the commands of core LaTeX 2$_\varepsilon$. Contributors are very welcome.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, second edition, 1986.

[2] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion (Tools and Techniques for Computer Typesetting).* Addison-Wesley, second edition, 2004.

◇ Jim Hefferon
Saint Michael's College
jhefferon (at) smcvt dot edu

[9] http://home.gna.org/latexrefman

[10] https://www.gnu.org/software/texinfo/

Astrological charts with `horoscop` and `starfont`

Matthew Skala

Abstract

TeX should create beautiful documents for all fields of human endeavour, and in this talk, I describe one frequently under-served by typesetting systems: astrology. Astrology has its own tradition of written knowledge and symbolic notation, analogous to that of mathematics but arguably even older; and like mathematics, astrology presents unique challenges for typesetting. Writers of astrological software often focus their attention primarily on the calculations, leaving any kind of graphical presentation as an afterthought. In this talk I present the `horoscop` and `starfont` LaTeX packages, meant for creating visually appealing astrological documents using TeX. No prior knowledge of astrology, and not too much of TeX, will be assumed.

`starfont` and `horoscop`

Human beings have looked for meaning in the sky since early prehistoric times. Some of our oldest written materials record the phases of the moon, and seasonal variations in the rising and setting of the sun. Astrological goals such as eclipse prediction motivated much of early mathematics; and mathematical developments in turn made possible more complicated astrological investigations. As mathematics developed a written symbolic notation, so did astrology. It is only recently that the two disciplines were considered distinct from each other at all.

So if TeX is the best tool for typesetting mathematics, then shouldn't it also be useful for typesetting astrology? The question is especially important because there are very few other good tools available. Historically, authors of astrological documents would draw their charts by hand, or use hand-set type. The availability of computers has made astrological computations much easier and more precise; but graphical output from astrological software is often disappointing.

I wrote the LaTeX packages called `starfont` and `horoscop`, starting around 2003, to bring high-quality astrological typesetting to the TeX world. Both are available from CTAN.

The `starfont` package provides the two fonts named StarFont Sans and StarFont Serif, designed by Anthony I.P. Owen. These fonts include signs of the zodiac like ♈♉♍ and planet glyphs like ☿♀♂, as well as other astrological and alchemical symbols. Other LaTeX packages offer some of these characters,

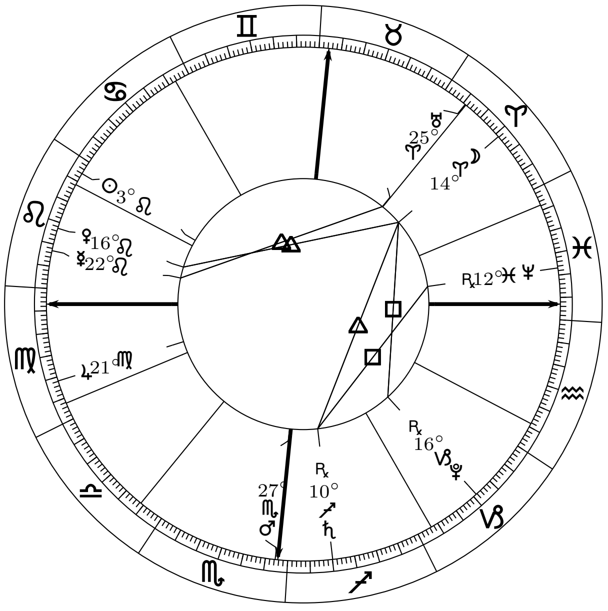


Figure 1: Mundane horoscope for the opening of TUG 2016: 8:45AM, July 25, 2016, Toronto.

but often in ways unsuitable for high-quality astrological typesetting. For example, `wasysym`'s Leo ♌ and North Node ☊ are indistinguishable.

The `horoscop` package's main function is generating wheel charts as in Figure 1. This kind of chart represents the sky at a specific time and place in a schematic form that emphasizes the information most relevant to astrological interpretation. The package can take manually-specified coordinates for celestial bodies or interface to external software via `\write18` to calculate their positions.

There are some challenges behind the scenes in typesetting such a chart. Most of the plotting is done in polar coordinates, requiring trigonometric calculations in TeX code. Labels plotted on the chart should not collide even if the objects they represent are near each other in the sky, so the package recalculates positions iteratively, using spring tension. A less visible challenge concerns rounding coordinates to lower precision, because the details of the rounding rules are significant in interpretation.

The `horoscop` package provides a range of ready-made chart designs, and also a framework for users to define their own. It aims to bring TeX's astrology to the same level as TeX's mathematics.

◇ Matthew Skala
Copenhagen
Denmark
`mskala (at) ansuz.sooke.bc.ca`
`http://ansuz.sooke.bc.ca/`

Remaking ACM LaTeX styles

Boris Veytsman

Abstract

The Association for Computing Machinery is one of the largest publishers of computation texts in the world. It publishes more than fifty journals and many more conference proceedings every year. It was among the early adopters of TeX.

Unfortunately, over the years ACM styles accumulated many patches and haphazard changes. They diverged to the point when supporting became an impossible task. This warranted a complete refactoring.

This talk discusses the experience of rewriting ACM styles and the lessons learned.

1 Introduction

Five years ago I was asked to update BibTeX styles for the Association of Computing Machinery (ACM). I did not know at that time that this commission would start a very interesting line of work.

The ACM [2] is one of the largest publishers in the computing and information science in the world. It produces dozens and dozens of journals and conference proceedings. Thus I considered the work on this assignment to be a great honor and a large responsibility.

It befits the ACM mission and traditions that it is one of the early adopters of TeX. There are timestamps in the ACM style files going as far back as the middle of 1980s, i.e., even predating TeX3. As any computer specialist knows only too well, code this old requires much care and attention lest it become a crazy quilt of patches upon patches (the integrity of TeX itself over the years is an important exception rather than the general rule). This is especially true when the code is maintained by generations of programmers stressed by deadlines and production requirements.

In the case of the ACM files, both the LaTeX and BibTeX code and the output display the result of many temporary *ad hoc* decisions and show overlapping fingerprints of editors and coders, often with incompatible philosophies and approaches. As one frustrated TeXpert wrote me (name withheld by request),

> ...3 packages copied in with a comment (good!) that they are needed but without taking out `\endinput` that was in the code from the package copied in (bad :-) so after the first nothing else is ever used...

> ...and it seems there is a redefinition of startsection inside that is broken—last night 30 min before my deadline I found 3 sections dangling at the bottom of columns...

> ...and the footnotes are horror and the fonts too and...

> ...looks worse than your average Word document ...

1. Class files:
   (a) `acm_proc_article-sp.cls`
   (b) `acmlarge.cls`
   (c) `acmsiggraph.cls`
   (d) `acmsmall-ec13.cls`
   (e) `acmsmall.cls`
   (f) `acmtog.cls`
   (g) `acmtrans2m.cls`
   (h) `sig-alternate-05-2015.cls`
   (i) `sig-alternate.cls`
   (j) `sigchi-ext.cls`
   (k) `sigchi.cls`
   (l) `sigplanconf.cls`
2. BibTeX styles:
   (a) `ACM-Reference-Format-Journals.bst`
   (b) `SIGCHI-Reference-Format.bst`
   (c) `acmsiggraph.bst`
   (d) `acm-abbrv.bst`
   (e) `acm-alpha.bst`
   (f) `acm-plain.bst`
   (g) `acm-unsrt.bst`

Figure 1: Legacy code base (2015)

These problems were exacerbated by the amount of copy and paste in the TeX code. Many times over the years whenever the need arose, the original code was cloned, changed in subtle (or not so subtle) ways, and a new class file was released. At the end of 2015 I found that I was dealing with as many as 12 class files and 7 BibTeX styles (Figure 1). Thus any update to the system required dozens of tantalizingly similar but slightly different changes in these files. This was not sustainable.

Another problem with the old styles was that interfaces to the elements like tables or figures were set long before the common standards were adopted. As the result, they looked quite strange for a LaTeX user. The unusual ways to do usual things were confusing to the authors and caused errors.

Thus, the decision of the senior staff of ACM to make a radical refactoring of the styles was excellent news. Both the typographic design and the coding were going to change. This was an opportunity to write the styles from scratch.

2 Organization of work

With many stakeholders, it took some effort to organize the writing of the styles and templates. The tasks were distributed as follows. The ACM editors updated the design and fonts selection. I wrote the LaTeX and BibTeX code. The company Aptara [1], which does typesetting for the ACM, developed word processor templates for the authors who do not use TeX, as well as tools for the extraction of metadata.

Since many conference committees (SIGs) wanted to be involved in the process, LaTeX and BibTeX code was put in a Github repository (`https://github.com/borisveytsman/acmart`). Github-based development turned out to be quite efficient for our purposes: the testers and SIG representatives could quickly assess the changes, submit bug reports and even contribute the code. Github seems to be a mature environment for free software development.

Sometimes it was difficult to accommodate the wishes of all the stakeholders, but we tried to keep in the spirit of compromise and consensus.

3 Design features

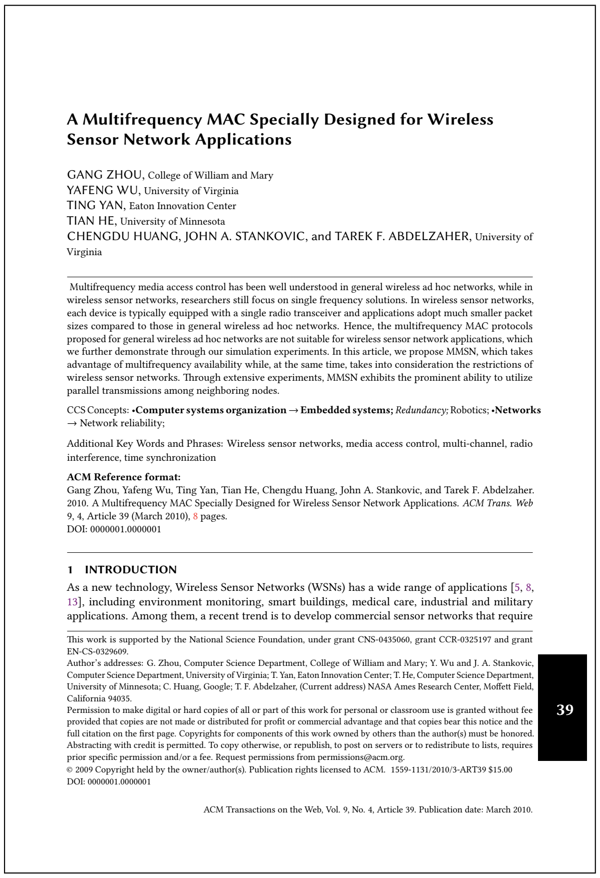
Instead of many class files (Figure 1) we use one class, *acmart*, with options corresponding to the output version. I sincerely hope this decision (one document class with options rather than several classes) will prevent the proliferation of copy-and-paste that plagued the old styles.

As suggested by the name, *acmart* is based on the famous *amsart* class [4], so all AMS-LaTeX advanced math typesetting features are available by default. You can use environments like `cases`, `gather` or `multline`, commands like `\dfrac` and `\tfrac` or `\text` in math mode, as well as AMS-style theorem definitions (the class itself defines several theorem-like constructs and theorem styles).

There are three journal options: *acmsmall* for small trim size journals, *acmlarge* for large trim size journals and *acmtog* for *Transactions on Graphics*, which traditionally uses two-column format. There are five proceedings options: *sigconf* for most conference proceedings, *siggraph*, *sigplan* and *sigchi* for specific proceedings with distinct formatting, and *sigchi-a* for the special SIGCHI Extended Abstract. The latter is quite unusual: it has wide margins with marginal figures and tables. Another option, *manuscript*, is for a generic manuscript.

In Figures 2, 3, 4 and 5 some examples of the output are shown. Additional samples can be found in the documentation on CTAN (`http://ctan.org/pkg/acmart`) or in your TeX distribution.

Another important decision was to eliminate use of proprietary fonts. The Libertine fonts [6] with



A Multifrequency MAC Specially Designed for Wireless Sensor Network Applications

GANG ZHOU, College of William and Mary
YAFENG WU, University of Virginia
TING YAN, Eaton Innovation Center
TIAN HE, University of Minnesota
CHENGDU HUANG, JOHN A. STANKOVIC, and TAREK F. ABDELZAHER, University of Virginia

Multifrequency media access control has been well understood in general wireless ad hoc networks, while in wireless sensor networks, researchers still focus on single frequency solutions. In wireless sensor networks, each device is typically equipped with a single radio transceiver and applications adopt much smaller packet sizes compared to those in general wireless ad hoc networks. Hence, the multifrequency MAC protocols proposed for general wireless ad hoc networks are not suitable for wireless sensor network applications, which we further demonstrate through our simulation experiments. In this article, we propose MMSN, which takes advantage of multifrequency availability while, at the same time, takes into consideration the restrictions of wireless sensor networks. Through extensive experiments, MMSN exhibits the prominent ability to utilize parallel transmissions among neighboring nodes.

CCS Concepts: •**Computer systems organization → Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Additional Key Words and Phrases: Wireless sensor networks, media access control, multi-channel, radio interference, time synchronization

ACM Reference format:
Gang Zhou, Yafeng Wu, Ting Yan, Tian He, Chengdu Huang, John A. Stankovic, and Tarek F. Abdelzaher. 2010. A Multifrequency MAC Specially Designed for Wireless Sensor Network Applications. *ACM Trans. Web* 9, 4, Article 39 (March 2010), 8 pages.
DOI: 0000001.0000001

1 INTRODUCTION

As a new technology, Wireless Sensor Networks (WSNs) has a wide range of applications [5, 8, 13], including environment monitoring, smart buildings, medical care, industrial and military applications. Among them, a recent trend is to develop commercial sensor networks that require

This work is supported by the National Science Foundation, under grant CNS-0435060, grant CCR-0325197 and grant EN-CS-0329609.
Author's addresses: G. Zhou, Computer Science Department, College of William and Mary; Y. Wu and J. A. Stankovic, Computer Science Department, University of Virginia; T. Yan, Eaton Innovation Center; T. He, Computer Science Department, University of Minnesota; C. Huang, Google; T. F. Abdelzaher, (Current address) NASA Ames Research Center, Moffett Field, California 94035.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2009 Copyright held by the owner/author(s). Publication rights licensed to ACM. 1559-1131/2010/3-ART39 $15.00
DOI: 0000001.0000001

39

ACM Transactions on the Web, Vol. 9, No. 4, Article 39. Publication date: March 2010.

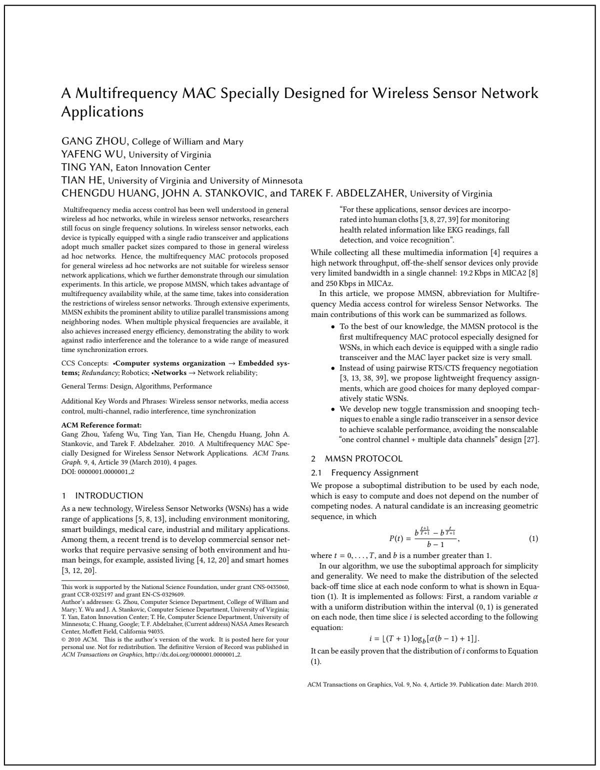
Figure 2: Journal output, small trim size: *acmsmall*

newtxmath [5] give the pages a clean and crisp look. The footnotes are no longer cramped. In general, we tried to add a little air to the pages, while keeping in mind that the authors are constrained by page count limits.

One of the main principles of the design is the integrity of the interface. While the typesetting of the journals and proceedings is quite different, the interface is the same. The author should be able merely to change *acmsmall* to *sigconf* option in the `\documentclass` command in order to typeset the manuscript in a different category. The only exception are the marginal figures and tables for the *sigchi-a* option, which have no corresponding material in the other formats.

Another principle is the logical markup with most visual decisions made by LaTeX. This can be demonstrated by the way authors' information is encoded. In the old design the authors should manually set the number of authors and align their addresses on the page using tabular-like commands. The new design does this automatically.

Since the TeX file is used both for typesetting and for automatic extraction of metadata by Aptara



A Multifrequency MAC Specially Designed for Wireless Sensor Network Applications

GANG ZHOU, College of William and Mary
YAFENG WU, University of Virginia
TING YAN, Eaton Innovation Center
TIAN HE, University of Virginia and University of Minnesota
CHENGDU HUANG, JOHN A. STANKOVIC, and TAREK F. ABDELZAHER, University of Virginia

Multifrequency media access control has been well understood in general wireless ad hoc networks, while in wireless sensor networks, researchers still focus on single frequency solutions. In wireless sensor networks, each device is typically equipped with a single radio transceiver and applications adopt much smaller packet sizes compared to those in general wireless ad hoc networks. Hence, the multifrequency MAC protocols proposed for general wireless ad hoc networks are not suitable for wireless sensor network applications, which we further demonstrate through our simulation experiments. In this article, we propose MMSN, which takes advantage of multifrequency availability while, at the same time, takes into consideration the restrictions of wireless sensor networks. Through extensive experiments, MMSN exhibits the prominent ability to utilize parallel transmissions among neighboring nodes. When multiple physical frequencies are available, it also achieves increased energy efficiency, demonstrating the ability to work against radio interference and the tolerance to a wide range of measured time synchronization errors.

CCS Concepts: •**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, media access control, multi-channel, radio interference, time synchronization

ACM Reference format:
Gang Zhou, Yafeng Wu, Ting Yan, Tian He, Chengdu Huang, John A. Stankovic, and Tarek F. Abdelzaher. 2010. A Multifrequency MAC Specially Designed for Wireless Sensor Network Applications. *ACM Trans. Graph.* 9, 4, Article 39 (March 2010), 4 pages.
DOI: 0000001.0000001_2

1 INTRODUCTION

As a new technology, Wireless Sensor Networks (WSNs) has a wide range of applications [5, 8, 13], including environment monitoring, smart buildings, medical care, industrial and military applications. Among them, a recent trend is to develop commercial sensor networks that require pervasive sensing of both environment and human beings, for example, assisted living [4, 12, 20] and smart homes [3, 12, 20].

> "For these applications, sensor devices are incorporated into human cloths [3, 8, 27, 39] for monitoring health related information like EKG readings, fall detection, and voice recognition".

While collecting all these multimedia information [4] requires a high network throughput, off-the-shelf sensor devices only provide very limited bandwidth in a single channel: 19.2 Kbps in MICA2 [8] and 250 Kbps in MICAz.

In this article, we propose MMSN, abbreviation for Multifrequency Media access control for wireless Sensor Networks. The main contributions of this work can be summarized as follows.

- To the best of our knowledge, the MMSN protocol is the first multifrequency MAC protocol especially designed for WSNs, in which each device is equipped with a single radio transceiver and the MAC layer packet size is very small.
- Instead of using pairwise RTS/CTS frequency negotiation [3, 13, 38, 39], we propose lightweight frequency assignments, which are good choices for many deployed comparatively static WSNs.
- We develop new toggle transmission and snooping techniques to enable a single radio transceiver in a sensor device to achieve scalable performance, avoiding the nonscalable "one control channel + multiple data channels" design [27].

2 MMSN PROTOCOL

2.1 Frequency Assignment

We propose a suboptimal distribution to be used by each node, which is easy to compute and does not depend on the number of competing nodes. A natural candidate is an increasing geometric sequence, in which

$$P(t) = \frac{b^{\frac{t+1}{T+1}} - b^{\frac{t}{T+1}}}{b-1}, \tag{1}$$

where $t = 0, \ldots, T$, and b is a number greater than 1.

In our algorithm, we use the suboptimal approach for simplicity and generality. We need to make the distribution of the selected back-off time slice at each node conform to what is shown in Equation (1). It is implemented as follows: First, a random variable α with a uniform distribution within the interval $(0, 1)$ is generated on each node, then time slice i is selected according to the following equation:

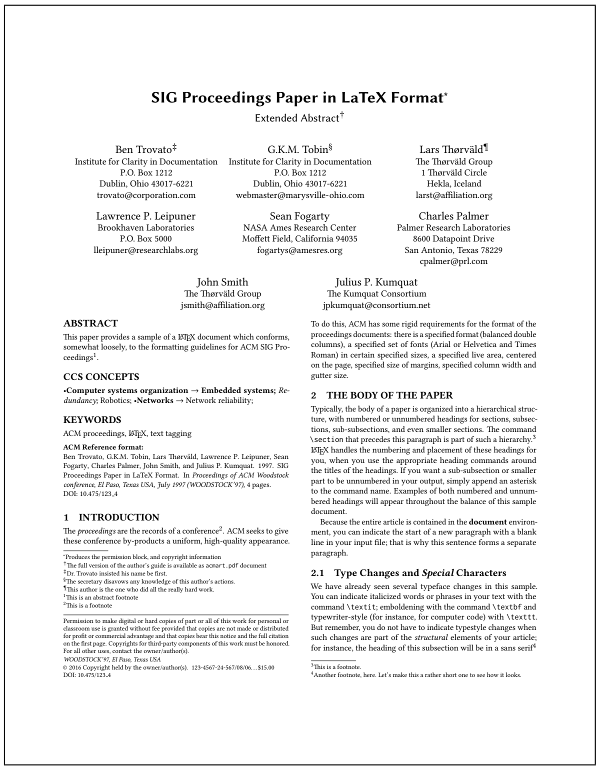
$$i = \lfloor (T+1)\log_b[\alpha(b-1)+1] \rfloor.$$

It can be easily proven that the distribution of i conforms to Equation (1).

This work is supported by the National Science Foundation, under grant CNS-0435060, grant CCR-0325197 and grant EN-CS-0329609.
Author's addresses: G. Zhou, Computer Science Department, College of William and Mary; Y. Wu and J. A. Stankovic, Computer Science Department, University of Virginia; T. Yan, Eaton Innovation Center; T. He, Computer Science Department, University of Minnesota; C. Huang, Google; T. F. Abdelzaher, (Current address) NASA Ames Research Center, Moffett Field, California 94035.
© 2010 ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, http://dx.doi.org/0000001.0000001_2.

ACM Transactions on Graphics, Vol. 9, No. 4, Article 39. Publication date: March 2010.

Figure 3: Journal output, two columns: *acmtog*



SIG Proceedings Paper in LaTeX Format[*]

Extended Abstract[†]

Ben Trovato[‡]
Institute for Clarity in Documentation
P.O. Box 1212
Dublin, Ohio 43017-6221
trovato@corporation.com

G.K.M. Tobin[§]
Institute for Clarity in Documentation
P.O. Box 1212
Dublin, Ohio 43017-6221
webmaster@marysville-ohio.com

Lars Thørväld[¶]
The Thørväld Group
1 Thørväld Circle
Hekla, Iceland
larst@affiliation.org

Lawrence P. Leipuner
Brookhaven Laboratories
P.O. Box 5000
lleipuner@researchlabs.org

Sean Fogarty
NASA Ames Research Center
Moffett Field, California 94035
fogartys@amesres.org

Charles Palmer
Palmer Research Laboratories
8600 Datapoint Drive
San Antonio, Texas 78229
cpalmer@prl.com

John Smith
The Thørväld Group
jsmith@affiliation.org

Julius P. Kumquat
The Kumquat Consortium
jpkumquat@consortium.net

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings[1].

CCS CONCEPTS

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference format:
Ben Trovato, G.K.M. Tobin, Lars Thørväld, Lawrence P. Leipuner, Sean Fogarty, Charles Palmer, John Smith, and Julius P. Kumquat. 1997. SIG Proceedings Paper in LaTeX Format. In *Proceedings of ACM Woodstock conference, El Paso, Texas USA, July 1997 (WOODSTOCK'97)*, 4 pages.
DOI: 10.475/123_4

1 INTRODUCTION

The *proceedings* are the records of a conference[2]. ACM seeks to give these conference by-products a uniform, high-quality appearance. To do this, ACM has some rigid requirements for the format of the proceedings documents: there is a specified format (balanced double columns), a specified set of fonts (Arial or Helvetica and Times Roman) in certain specified sizes, a specified live area, centered on the page, specified size of margins, specified column width and gutter size.

2 THE BODY OF THE PAPER

Typically, the body of a paper is organized into a hierarchical structure, with numbered or unnumbered headings for sections, subsections, sub-subsections, and even smaller sections. The command `\section` that precedes this paragraph is part of such a hierarchy.[3] LaTeX handles the numbering and placement of these headings for you, when you use the appropriate heading commands around the titles of the headings. If you want a sub-subsection or smaller part to be unnumbered in your output, simply append an asterisk to the command name. Examples of both numbered and unnumbered headings will appear throughout the balance of this sample document.

Because the entire article is contained in the **document** environment, you can indicate the start of a new paragraph with a blank line in your input file; that is why this sentence forms a separate paragraph.

2.1 Type Changes and *Special* Characters

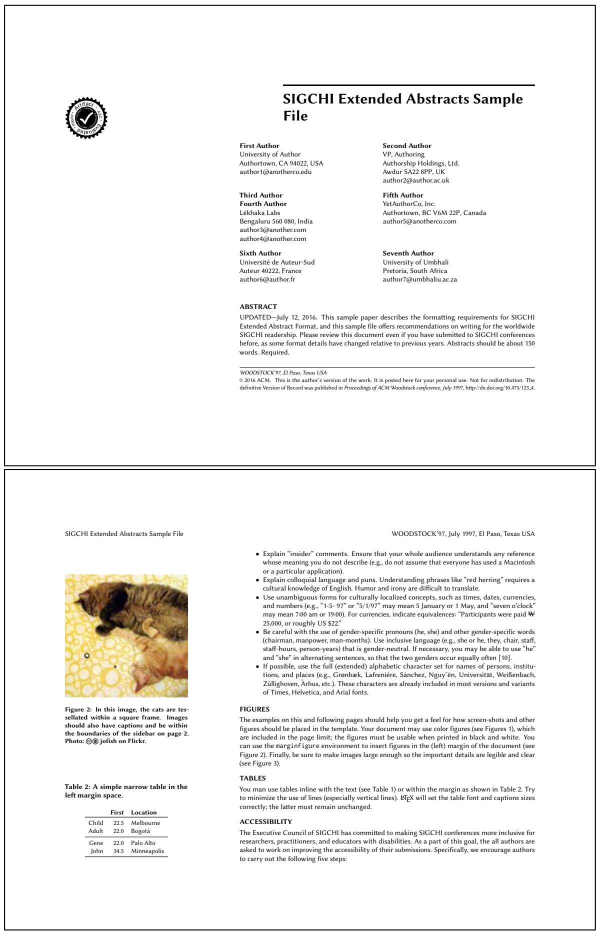
We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the heading of this subsection will be in a sans serif[4]

[*]Produces the permission block, and copyright information
[†]The full version of the author's guide is available as `acmart.pdf` document
[‡]Dr. Trovato insisted his name be first.
[§]The secretary disavows any knowledge of this author's actions.
[¶]This author is the one who did all the really hard work.
[1]This is an abstract footnote
[2]This is a footnote

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
WOODSTOCK'97, El Paso, Texas USA
© 2016 Copyright held by the owner/author(s). 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

[3]This is a footnote.
[4]Another footnote, here. Let's make this a rather short one to see how it looks.

Figure 4: Proceedings output: *sigconf*



SIGCHI Extended Abstracts Sample File

First Author
University of Author
Authortown, CA 94022, USA
author1@anotherco.edu

Second Author
VP, Authoring
Authorship Holdings, Ltd.
Awdur SA22 8PP, UK
author2@author.ac.uk

Third Author
Fourth Author
Lēkhaka Labs
Bengaluru 560 080, India
author3@another.com
author4@another.com

Fifth Author
YetAuthorCo, Inc.
Authortown, BC V6M 22P, Canada
author5@anotherco.com

Sixth Author
Université de Auteur-Sud
Auteur 40222, France
author6@author.fr

Seventh Author
University of Umbhali
Pretoria, South Africa
author7@umbhaliu.ac.za

ABSTRACT

UPDATED—July 12, 2016. This sample paper describes the formatting requirements for SIGCHI Extended Abstract Format, and this sample file offers recommendations on writing for the worldwide SIGCHI readership. Please review this document even if you have submitted to SIGCHI conferences before, as some format details have changed relative to previous years. Abstracts should be about 150 words. Required.

WOODSTOCK'97, El Paso, Texas USA
© 2016 ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of ACM Woodstock conference, July 1997*, http://dx.doi.org/10.475/123_4.

SIGCHI Extended Abstracts Sample File — WOODSTOCK'97, July 1997, El Paso, Texas USA

- Explain "insider" comments. Ensure that your whole audience understands any reference whose meaning you do not describe (e.g., do not assume that everyone has used a Macintosh or a particular application).
- Explain colloquial language and puns. Understanding phrases like "red herring" requires a cultural knowledge of English. Humor and irony are difficult to translate.
- Use unambiguous forms for culturally localized concepts, such as times, dates, currencies, and numbers (e.g., "1-5- 97" or "5/1/97" may mean 5 January or 1 May, and "seven o'clock" may mean 7:00 am or 19:00). For currencies, indicate equivalences: "Participants were paid ₩ 25,000, or roughly US $22."
- Be careful with the use of gender-specific pronouns (he, she) and other gender-specific words (chairman, manpower, man-months). Use inclusive language (e.g., she or he, they, chair, staff, staff-hours, person-years) that is gender-neutral. If necessary, you may be able to use "he" and "she" in alternating sentences, so that the two genders occur equally often [10].
- If possible, use the full (extended) alphabetic character set for names of persons, institutions, and places (e.g., Grønbæk, Lafreniére, Sánchez, Nguy˜ên, Universität, Weißenbach, Züllighoven, Århus, etc.). These characters are already included in most versions and variants of Times, Helvetica, and Arial fonts.

Figure 2: In this image, the cats are tessellated within a square frame. Images should also have captions and be within the boundaries of the sidebar on page 2. Photo: 🅭🅯 jofish on Flickr.

FIGURES

The examples on this and following pages should help you get a feel for how screen-shots and other figures should be placed in the template. Your document may use color figures (see Figures 1), which are included in the page limit; the figures must be usable when printed in black and white. You can use the `marginfigure` environment to insert figures in the (left) margin of the document (see Figure 2). Finally, be sure to make images large enough so the important details are legible and clear (see Figure 3).

Table 2: A simple narrow table in the left margin space.

	First	Location
Child	22.5	Melbourne
Adult	22.0	Bogotá
Gene	22.0	Palo Alto
John	34.5	Minneapolis

TABLES

You man use tables inline with the text (see Table 1) or within the margin as shown in Table 2. Try to minimize the use of lines (especially vertical lines). LaTeX will set the table font and captions sizes correctly; the latter must remain unchanged.

ACCESSIBILITY

The Executive Council of SIGCHI has committed to making SIGCHI conferences more inclusive for researchers, practitioners, and educators with disabilities. As a part of this goal, the all authors are asked to work on improving the accessibility of their submissions. Specifically, we encourage authors to carry out the following five steps:

Figure 5: SIGCHI Extended abstract: *sigchi-a*

```
\author{Ben Trovato}
\authornote{Dr.~Trovato insisted his
  name be first.}
\orcid{1234-5678-9012}
\email{trovato@example.edu}
\author{A. U. Thor}
\email{author@example.edu}
\affiliation{%
  \institution{Institute for Clarity
    in Documentation}
  \streetaddress{P.O. Box 1212}
  \city{Dublin}
  \state{Ohio}
  \postcode{43017-6221}
  \country{USA}}
```

Figure 6: Example of author information commands

tools, the commands are highly structured. For example, the authors' information is typed using the commands like `\streetaddress` or `\city` (Figure 6). There are special commands for grant sponsors and grant numbers, etc.

The class offers a number of useful features like canned copyright statements (vetted by the ACM lawyers), review mode with line numbers printed, anonymous mode with the information about the authors, affiliations, grants and acknowledgments suppressed (for a blind review), etc. This anonymous mode is just one of the options for conditional typesetting: the authors could also have different versions for the online and hard-copy; for example, the online version could include supplementary materials. There are provisions to include CCS "concepts": hierarchical keywords generated by the ACM website.

The class uses standard $\LaTeX\,2_\varepsilon$ interfaces to common elements, such as figures and tables, as much as possible. The only area with ACM-specific commands is the front matter: unfortunately all publishers use their own systems to indicate the authors and their affiliations, and ACM is no exception here.

4 Bibliography

Historically some ACM publications used author-year citations, while other used numbered cites. Even the author-year ones were not uniform: some used *natbib*, while some used their own interface. There were pervasive differences in bibliography formatting. This led to a large number of "official" ACM BibTeX styles (see Figure 1).

The new *acmart* package uses only one BibTeX style, which is *natbib*-compatible and defaults to numeric citations. Fortunately, the *natbib* package [3] allows the user to choose either author-year or numbered citations, thus allowing SIGs to customize their bibliographies. Even when the citation style is numeric, commands like `\citeyear` and `\citeauthor` are allowed.

Another interesting feature of the citation style is that the bibliographic output is highly structured for use by the cross-referencing software. This is done transparently to the user, creating the entries like the one shown on Figure 7.

```
\bibitem[\protect\citeauthoryear{Akyildiz,
    Melodia, and Chowdhury}{Akyildiz
    et~al\mbox{.}}{2007}]%
  {Akyildiz-02}
\bibfield{author}{\bibinfo{person}{I.~F.
    Akyildiz},
  \bibinfo{person}{T. Melodia}, {and}
  \bibinfo{person}{K.~R. Chowdhury}.}
\bibinfo{year}{2007}).
\newblock \showarticletitle{A Survey on
  Wireless  Multimedia Sensor Networks}.
\newblock \bibinfo{journal}{{\em Computer
    Netw.\/}}
\bibinfo{volume}{{51}, 4},
\bibinfo{pages}{921--960}.
```

Figure 7: Bibliography entry made by the new ACM `bst` file

5 Conclusions and acknowledgments

This large work of creating the new ACM styles would not be possible without the help of many people. I would like to express my gratitude to:

- ACM editors: Craig Rodkin, Bernard Rous.
- Aptara: Neeraj Saxena, Sehar Tahir.
- Testers, users and SIG representatives: Chris Guccio, Wayne Graves, Matthew Fluet, Jofish Kaye, Frank Mittelbach, John Owens, Tobias Pape, David A. Shamma, Stephen Spencer.
- Authors of the early versions of ACM TeX and BibTeX styles.

The new ACM styles are available on CTAN and the ACM web site, as well as in the major TeX distributions like TeX Live and MikTeX.

As mentioned above, development is hosted at Github, `https://github.com/borisveytsman/acmart`. The Github interface is the best way to send me bug reports or feature suggestions.

References

[1] Aptara. `http://www.aptaracorp.com`.

[2] Association for Computing Machinery. `http://www.acm.org`.

[3] Patrick W. Daly. *Natural Sciences Citations and References (Author-Year and Numerical Schemes)*, 2010. `http://ctan.org/pkg/natbib`.

[4] Michael Downes and Barbara Beeton. *The amsart, amsproc, and amsbook document classes*. American Mathematical Society, 2015. `http://ctan.org/pkg/amsart`.

[5] Michael Sharpe. *New TX font package*, 2016. `http://ctan.org/pkg/newtx`.

[6] Bob Tennent. *LaTeX Support for Linux Libertine and Biolinum Fonts*, 2014. `http://ctan.org/pkg/libertine`.

◇ Boris Veytsman
Systems Biology School and Computational Materials Science Center
MS 6A2
George Mason University
Fairfax, VA 22030 USA
borisv (at) lk dot net
http://borisv.lk.net

Advances in PythonTEX with an introduction to fvextra

Geoffrey M. Poore

Abstract

The PythonTEX package allows Python and several other programming languages to be embedded within LATEX documents. It also typesets code with syntax highlighting provided by the Pygments library for Python. Code typesetting has been improved with the new fvextra package, which builds on fancyvrb by allowing long lines of code to be broken and by providing several additional features. Code execution has been improved by a new set of commands and environments that perform variable substitution or string interpolation. This makes it easier to mix LATEX with Python or other languages while avoiding errors due to expansion, tokenization, and catcodes.

1 Limitations with code typesetting and execution

In 2012, I released the first version of the PythonTEX package [13] with the goal of making it simpler to write mathematical and scientific LATEX documents. PythonTEX allows the LATEX source of a document to contain both a mathematical result and the Python code that calculated it, or both a plot and the Python code that generated it. Originally, PythonTEX only allowed Python code in a LATEX document to be executed, with the output included in the document. It is now possible to execute Ruby, Octave, Sage, Bash, and Rust code as well. From the beginning, PythonTEX has also allowed general code typesetting with syntax highlighting.

PythonTEX's code typesetting has been functional but relatively basic. It uses the Pygments library [15] for Python to perform syntax highlighting. (Pygments is also used for syntax highlighting by the minted [14] package, which I maintain, as well as the verbments [18], texments [3], and pygmentex [5] packages.) Pygments supports over 300 languages and can perform highlighting that would not be practical in a pure LATEX solution such as the listings package [2]. Yet Pygments is not without drawbacks. It uses the fancyvrb package [16] to perform the actual code typesetting. fancyvrb lacks many of the advanced features found in listings, such as the ability to break long lines of code. Unfortunately, attempting to use listings instead of fancyvrb brings its own set of issues; among other things, listings lacks built-in support for UTF-8 and other multi-byte encodings under the pdfTEX engine.

This paper introduces fvextra [12], a new package I have created to address these limitations in code typesetting. fvextra extends and patches fancyvrb. It provides most of the features that fancyvrb lacks compared to listings, including line breaking with fine-grained control over break locations. The fvextra package also provides additional features, such as the ability to highlight specific lines or line ranges based on line numbers. The most recent versions of PythonTEX and minted require fvextra and fully support all new features.

Another longstanding drawback in PythonTEX relates to code execution rather than typesetting. Documents that use PythonTEX are valid LATEX documents; there is no preprocessing step to produce LATEX source. A PDF or other output is created by running LATEX (code is saved to a temporary file), then running the `pythontex` executable (code is executed), and finally running LATEX again (code output is brought into the final PDF). The advantage of this approach is that it is possible to create macros that mix LATEX with Python or other languages. Since LATEX handles all code before it is executed, code can be assembled using macros. In a preprocessor approach such as that used by Sweave [4], knitr [17], and Pweave [9], this is generally not possible because LATEX only receives a copy of the document in which code has been replaced by its output.

The disadvantage of PythonTEX not being a preprocessor is that LATEX does indeed process everything. For example, it is not possible to use a PythonTEX command to insert Python output in the midst of a `verbatim` environment; the command would appear literally. Similarly, PythonTEX commands can cause errors within `tikzpicture` environments or in other situations in which characters do not have their normal meanings (catcodes) or in which other special processing is applied.

This paper introduces a new solution for these scenarios. New commands and environments perform variable substitution or string interpolation. These effectively allow the preprocessor approach to be applied to the argument of a command or the contents of an environment. It is now simpler to mix LATEX with Python or another language while avoiding errors due to expansion, tokenization, and catcodes.

2 A brief overview of PythonTEX

Before describing new PythonTEX features, I will briefly summarize PythonTEX usage to provide context. General PythonTEX usage has been explored in greater detail previously in *TUGboat* [6] and elsewhere [10, 11].

Using PythonTEX involves loading the package in the preamble:

```
\usepackage{pythontex}
```

and modifying the compile process. As mentioned above, PythonTEX requires a three-step compile. For a document doc.tex, this might look like

```
pdflatex doc.tex
pythontex doc.tex
pdflatex doc.tex
```

The pythontex executable is typically installed along with the package when PythonTEX is installed with a TEX distribution's package manager. The second and third steps of the compile process are only necessary when code needs to be executed or highlighted. PythonTEX caches all results to maximize performance, and the pythontex executable will not actually do anything unless it detects changes.

2.1 Code typesetting

PythonTEX provides a \pygment command and a pygments environment for general code typesetting. These are similar to the \mintinline command and minted environment provided by the minted package. Colorizing is automatic (but grayscaled here for the printed *TUGboat*). For example,

```
Inline: \pygment{python}{var = "string"}
```

results in

Inline: `var = "string"`

The code may be delimited by a pair of curly braces, as shown, or by a single pair of identical characters (like \verb). Similarly,

```
\begin{pygments}{python}
def func(var):
    return var**2
\end{pygments}
```

produces

```
def func(var):
    return var**2
```

Code typesetting may be customized using fancyvrb's \fvset, which applies document-wide options, or the \setpygmentsfv command, which restricts options to \pygment and pygments.

There are also language-specific commands and environments that do not need the language to be specified. For example, \pyv and pyverbatim could be substituted in the examples above if "{python}" were deleted. Typesetting may be customized with \fvset or \setpythontexfv.

2.2 Code execution

There is a \pyc command and a pycode environment that may be used to execute Python code. Similar commands and environments exist for Ruby, Octave, Sage, Bash, and Rust. By default, anything that is printed or written to stdout will automatically be included in the document, just as if it had been saved in an external file and then brought in via \input. For example,

```
\begin{pycode}
print("Python says hello to \\tug!")
\end{pycode}
```

produces

Python says hello to TUG!

Notice that by default printed text is interpreted as normal LATEX input, not as verbatim.

There is also a \py command for conveniently inserting string representations of Python expressions in a document. It would be possible to use the \pyc command for this purpose. For example, to insert the value of 2^8 into the document, this would suffice:

```
\pyc{print(2**8)}
```

However, \py is more convenient:

```
\py{2**8}
```

It is also possible to use \py to insert the value of a previously defined variable. For instance, if \pyc{x = 2**8} had been used previously to set the value of x, then \py{x} would produce 256.

2.3 Code typesetting and execution

There is a \pyb command and a pyblock environment that both typeset and execute code. Anything printed or written to stdout by the code is not automatically inserted in the document, since it might not be desirable to have typeset code immediately next to its output. Instead, anything printed by the most recent command or environment may be inserted using the \printpythontex command.

3 An introduction to fvextra

The fancyvrb package was first publicly released in 1998 at version 2.5. A few bugs were fixed and a few features added later that year in version 2.6. Since then, the documentation lists two bug fixes, with version 2.8 released in 2010. The stability of fancyvrb speaks to its success as a fancy verbatim package.

I released the first version of the fvextra package at the end of June 2016. The package focuses on adding features to fancyvrb that improve code typesetting, especially when used with syntax highlighting provided by Pygments. fvextra also implements a few patches to the fancyvrb internals and makes a few changes to default fancyvrb behavior. All patches and changes to defaults are detailed in the fvextra documentation. At the end of July 2016, I released PythonTEX version 0.15 and minted 2.4. These require the fvextra package and support all features described below.

3.1 Single quotation marks

By default, LaTeX verbatim commands and environments convert the backtick (`` ` ``) and single typewriter quotation mark (') into the left and right curly single quotation marks (‘’). This behavior carries over into fancyvrb. In typeset code, these characters should be represented literally. This is typically accomplished by manually loading the upquote package [1].

That approach has two drawbacks. First, not using upquote by default means that it is easily forgotten. I have had the experience myself of reviewing the final proofs of a paper, only to realize that I had forgotten to load upquote. Second, when upquote is loaded, obtaining the curly quotation marks is inconvenient if they are ever legitimately desired in a verbatim context.

The fvextra package requires upquote, so that the default behavior is correct for typesetting code. It also defines a new `curlyquotes` option that restores the default LaTeX behavior. For example, using fancyvrb's `Verbatim` environment,

```
\begin{Verbatim}[curlyquotes]
`Single quoted text'
\end{Verbatim}
```

yields

```
‘Single quoted text’
```

This eliminates one of the most common mistakes in code typesetting while still providing convenient access to the normal LaTeX behavior.

3.2 Math in verbatim

The fancyvrb package allows typeset mathematics to be embedded within verbatim material. Pygments builds on this with its `mathescape` option, which enables typeset math within code comments. That can be useful when typesetting code that implements mathematical or scientific algorithms.

A close examination of fancyvrb's typeset math within verbatim reveals that the result differs from normal math mode. Spaces are significant and appear literally, rather than vanishing. The `\text` command provided by the amstext package [7] and loaded as part of amsmath [8] uses the verbatim font rather than the normal document font. The single quotation mark (') causes an error rather than becoming a prime (that is, being converted into `^\prime`). fvextra modifies typeset math within verbatim so that all of these behave as expected. For example,

```
\begin{Verbatim}[commandchars=\\\{\},
                 mathescape]
Verbatim $x^2 + f_\text{sub}(x) = g''(x)$
\end{Verbatim}
```

now correctly produces

`Verbatim` $x^2 + f_\text{sub}(x) = g''(x)$

The `commandchars` option used in this example is defined by fancyvrb and allows macros within otherwise verbatim material. The `mathescape` option is a new feature added by fvextra that serves as a shortcut for giving the dollar sign, underscore, and caret their normal math-related meanings. When fvextra's `mathescape` is used with code highlighted by Pygments, it reduces to Pygments' `mathescape`, only producing typeset math in comments.

3.3 Tabs and tab expansion

By default, fancyvrb converts tabs into a fixed number of spaces, which may be controlled with the `tabsize` option. It also offers tab expansion to tab stops with the `obeytabs` option.

Tab expansion involves a clever recursive algorithm. Each tab character causes everything that precedes it in the current line of text to be saved in a box, and the width of the box is compared to the tab stop size to determine the needed width for the current tab. (For those who would like more details, this is defined in the `\FV@@ObeyTabs` and `\FV@TrueTab` macros in `fancyvrb.sty`.)

The tab expansion algorithm works excellently in normal verbatim contexts. Unfortunately, it also fails spectacularly (and silently) when tabs occur within macro arguments, which is common in Pygments output. In a multiline string or comment that is indented with tabs, `obeytabs` typically causes all lines except the first and the last to vanish, with no errors or warnings.

fvextra patches tab expansion so that it will never cause lines to vanish. Tab expansion for tabs that are only preceded by spaces or tabs is always correct, even for tabs that are in macro arguments. This covers the most common case of tabs used for indentation. Unfortunately, tab expansion is not guaranteed to be correct for tabs within macro arguments that are preceded by non-tab, non-space characters. The limitations of the new tab expansion algorithm are discussed in detail in the documentation.

Tabs are also improved in fvextra with the addition of the `tab` and `tabcolor` options. For example,

```
\begin{Verbatim}[showtabs,
                 tab=\rightarrowfill,
                 tabcolor=orange]
        A tab-indented line of text
\end{Verbatim}
```

produces

```
⟶A tab-indented line of text
```

The original fancyvrb treatment of visible tabs was modified so that variable-width symbols such as `\rightarrowfill` expand to fill the full tab width.

3.4 Line highlighting

When writing about code, it can be useful to highlight a specific line or range of lines based on line numbers. As far as I know, fvextra is the first LaTeX package to implement this with its `highlightlines` and `highlightcolor` options. For example,

```
\begin{Verbatim}[numbers=left,
                 highlightlines={1, 3-4}]
First line
Second line
Third line
Fourth line
Fifth line
\end{Verbatim}
```

results in

```
1  First line
2  Second line
3  Third line
4  Fourth line
5  Fifth line
```

By default, a `\colorbox` that uses `highlightcolor` is inserted around specified lines. Additional customization is possible when desired. fvextra defines macros that are applied to the first, last, and inner lines in a range, as well as to isolated highlighted lines and to unhighlighted lines. These macros may be redefined to produce fancier highlighting.

3.5 Line breaking

The ability to automatically break long lines of code is perhaps the most important feature present in listings but missing in fancyvrb.

The fvextra package adds an option `breaklines` that enables line breaking. Line breaking is turned off by default, to ensure that the standard behavior of fancyvrb is unchanged by fvextra. Perhaps it will also encourage users to consider a smaller font size, inserting hard line breaks, or otherwise modifying code as an alternative to automatic line breaking.

By default, `breaklines` indents continuation lines to the same indentation level as the start of the line (adjustable via option `breakautoindent`), and then adds a small amount of extra indentation to make room for a line continuation symbol on the left (adjustable via `breaksymbolindentleft`). For instance,

```
A line that would eventually end up in the
 ↪ margin without breaklines
      An indented line that would be too
       ↪ long without breaklines
```

Many options are provided for customizing break indentation and break symbols. One possibility is to use custom break symbols on both the left and the right. Break symbols could be defined:

```
\newcommand{\symleft}{%
  \ensuremath{\Rightarrow}}
\newcommand{\symright}{\raisebox{-1ex}{%
  \rotatebox{30}{\ensuremath{\Leftarrow}}}}
```

Then the following options could be added:

```
breaklines,
breaksymbolleft=\symleft,
breaksymbolright=\symright
```

An example using these settings is shown below.

```
A line that would eventually end up in      ⇐
 ⇒ the margin without breaklines
```

By default, line breaks occur only at spaces when the `breaklines` option is used. Breaks may also be allowed anywhere (between non-space characters) by turning on the additional option `breakanywhere`. In many cases, however, simply breaking anywhere will not be acceptable. Two more options, `breakbefore` and `breakafter`, allow specific characters to be specified as break locations. For instance, setting

```
breakafter={+-=,}
```

allows breaks after any of the specified characters (`+-=,`). This could be useful for allowing breaks when spaces are not present while avoiding breaks within variable names. Special LaTeX characters such as the percent sign and number sign must be backslash-escaped when passed to `breakbefore` and `breakafter`. When a given character is specified as a potential break location, by default breaks will not be inserted between identical characters; rather, runs of identical characters are grouped. This may be modified with the `breakbeforegroup` and `breakaftergroup` options.

All of the breaking options discussed so far apply equally well to both normal verbatim text and highlighted computer code output by Pygments. fvextra also provides two line breaking options which are specific to Pygments output, and thus intended for the PythonTeX and minted packages. The `breakbytoken` option prevents line breaks from occurring within Pygments tokens, such as strings, comments, keywords, and operators. A complete list of Pygments tokens is available at `pygments.org/docs/tokens`. Breaks are still allowed at spaces outside tokens. The `breakbytoken` option could be used in a case like

```
var = "string 1" + "string 2" + "string 3"
```

to prevent breaks from occurring inside the strings, while still allowing breaks at spaces elsewhere.

There is also a `breakbytokenanywhere` option that prevents breaks within tokens, but allows breaks between immediately adjacent tokens. This could be used in a case like

```
var = "string 1"+"string 2"+"string 3"
```

to prevent breaks within the strings while still allowing breaks before and after the plus signs.

4 Variable substitution and string interpolation

As mentioned in the introduction, one of the advantages of PythonTeX is that it allows macro programming that mixes LaTeX with Python or another language. For instance, I could define a command that swaps the first and last characters in a string:

```
\newcommand{\swapfirstlast}[1]{%
  \pyc{s = "#1"}%
  \py{s[-1] + s[1:-1] + s[0]}}
```

This stores the argument of the command as a Python string, and then uses the character indices to swap the first and last characters. Invoking

```
\swapfirstlast{0123456789}
```

yields

9123456780

Such convenience is only possible because PythonTeX does not function as a preprocessor; LaTeX handles all text before code is seen by Python (or another language) for evaluation, and then the result of evaluation is brought in during the next compile. In this case, LaTeX macros are used to assemble Python code that is subsequently evaluated.

The downside of this approach is that it makes it more difficult to evaluate Python code in a verbatim or other special context. As should be expected,

```
\begin{Verbatim}
x = \py{2**16}
\end{Verbatim}
```

simply produces the literal text

```
x = \py{2**16}
```

Though that makes it convenient to write about PythonTeX, it certainly does make it more difficult to insert Python output in some situations.

In a case like this, it is possible to assemble all of the text as a Python template, and then print it:

```
\begin{pycode}
s = """
\\begin{{Verbatim}}
x = {x}
\\end{{Verbatim}}
"""
print(s.format(x=2**16))
\end{pycode}
```

That does give the desired result:

```
x = 65536
```

Unfortunately, a certain amount of complexity is required even for this simple case. The backslash must be escaped unless a raw string is used. Curly braces, which are of course everywhere in LaTeX, must be doubled to appear literally when used with Python's string formatting.

To simplify these cases, PythonTeX now includes a `\pys` command and `pysub` environment that perform variable substitution or string interpolation. Equivalent commands and environments exist for Ruby, Octave, Sage, Bash, and Rust. Using the `pysub` environment, the last example becomes

```
\begin{pysub}
\begin{Verbatim}
x = !{2**16}
\end{Verbatim}
\end{pysub}
```

The content of the environment is passed verbatim to Python. Substitution fields take the form `!{`⟨*expression*⟩`}`. After ⟨*expression*⟩ is evaluated, a string representation of the result is returned to LaTeX. If ⟨*expression*⟩ is simply a variable name, then it is replaced with a string representation of the variable value.

The form `!{`⟨*expression*⟩`}` was chosen because the exclamation point is one of the few ASCII punctuation characters without a special LaTeX meaning. Using more common string interpolation syntax from other languages seemed unwise; `$`⟨*variable*⟩, `${`⟨*expression*⟩`}`, and `#{`⟨*expression*⟩`}` are constructs which commonly appear in LaTeX. Likewise, using Python's string formatting syntax of `{`⟨*variable*⟩`}` would be problematic, since it would require all literal curly braces to be escaped by doubling.

The exact rules for delimiting and escaping `!{`⟨*expression*⟩`}` differ somewhat from standard LaTeX syntax. If a literal exclamation point followed by an opening curly brace is desired, then the exclamation point is escaped by doubling (`!!`). A literal exclamation point only needs to be escaped when followed immediately by an opening curly brace. Curly braces never need to be escaped, since they only delimit a substitution field when they immediately follow an unescaped exclamation point.

If ⟨*expression*⟩ is delimited by a single pair of curly braces, `!{`⟨*expression*⟩`}`, then it may contain paired curly braces up to five levels deep. If the first or last character in ⟨*expression*⟩ would be a curly brace, then it must be separated from the delimiting braces by a space; leading and trailing spaces are stripped before ⟨*expression*⟩ is evaluated.

While ⟨*expression*⟩ will typically contain paired curly braces, there may be times when it does not. In these cases, it may be delimited by a sequence of curly braces up to six levels deep. Then ⟨*expression*⟩ must not contain an opening or closing sequence of the same depth as the delimiters, but may contain any combination of shorter sequences. For example, in

```
!{{{⟨expression⟩}}}
```

the ⟨*expression*⟩ could contain any combination of `{`, `}`, `{{`, or `}}`, paired or unpaired. It could not contain `{{{` or `}}}`, however; that would require delimiters of greater depth.

The `\pys` command is directly analogous to the `pysub` environment and follows the same rules. For instance,

```
\pys{\verb|x = !{2**32}|}
```

yields

```
x = 4294967296
```

Like the other PythonTEX commands, `\pys` takes an argument delimited by curly braces or by a single matched character, like `\verb`.

Both of the examples of `\pys` and `pysub` above involve verbatim. There are other situations in which they are useful. For example, in the `tikzpicture` environment provided by the tikz package, the `\py` command will typically conflict with tikz processing and result in an error. This may sometimes be avoided by using `\py` to output an entire line of tikz code, including the terminating semicolon, rather than just a snippet of text. The `pysub` environment provides a simpler alternative that avoids any guesswork regarding potential conflicts.

5 Conclusion

With the new fvextra package, it is now possible to typeset code using Pygments syntax highlighting without sacrificing advanced code typesetting features, such as line breaking. This should make PythonTEX (and minted) significantly better options for code typesetting in the future.

PythonTEX's code execution capabilities have also been improved by the new `\pys` command and `pysub` environment, and other commands and environments for variable substitution or string interpolation. These remove many of the remaining obstacles to document programming mixing LATEX with Python, or with any of the other languages supported by PythonTEX, including Ruby, Octave, Sage, Bash, and Rust.

References

[1] Michael A. Covington, Frank Mittelbach, and Markus G. Kuhn. *upquote — upright-quote and grave-accent glyphs in verbatim.* `ctan.org/pkg/upquote`, 2012.

[2] Carsten Heinz, Brooks Moses, and Jobst Hoffmann. *The listings package.* `ctan.org/pkg/listings`, 2013.

[3] Marek Kubica. *The texments package.* `ctan.org/pkg/texments`, 2008.

[4] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in computational statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. ISBN 3-7908-1517-9.

[5] José Romildo Malaquias. *Testing the PygmenTEX package.* `ctan.org/pkg/pygmentex`, 2014.

[6] Andrew Mertz and William Slough. A gentle introduction to PythonTEX. *TUGboat*, 34(3):302–312, 2013. `tug.org/TUGboat/tb34-3/tb108mertz.pdf`.

[7] Frank Mittelbach and Rainer Schöpf. *The amstext package.* `ctan.org/pkg/amstext`, 2000.

[8] Frank Mittelbach, Rainer Schöpf, Michael Downes, and David M. Jones. *The amsmath package.* `ctan.org/pkg/amsmath`, 2016.

[9] Matti Pastell. *Pweave — reports from data with Python.* `mpastell.com/pweave`, 2010.

[10] Geoffrey M. Poore. Reproducible documents with PythonTEX. In Stéfan van der Walt, Jarrod Millman, and Katy Huff, editors, *Proc. of the 12th Python in Science Conference*, pages 78–84, 2013.

[11] Geoffrey M. Poore. PythonTEX: Reproducible documents with LATEX, Python, and more. *Comp. Science & Discovery*, 8(1):014010, 2015.

[12] Geoffrey M. Poore. *The fvextra package.* `github.com/gpoore/fvextra`, 2016.

[13] Geoffrey M. Poore. *The pythontex package.* `github.com/gpoore/pythontex`, 2016.

[14] Geoffrey M. Poore and Konrad Rudolph. *The minted package: Highlighted source code in LATEX.* `github.com/gpoore/minted`, 2016.

[15] The Pocoo Team. *Pygments: Python syntax highlighter.* `pygments.org`, 2016.

[16] Timothy Van Zandt, Denis Girou, Sebastian Rahtz, and Herbert Voß. *The 'fancyvrb' package: Fancy verbatims in LATEX.* `ctan.org/pkg/fancyvrb`, 2010.

[17] Yihui Xie. *Dynamic Documents with R and knitr.* Chapman and Hall/CRC, Boca Raton, FL, 2013. ISBN 978-1482203530, `yihui.name/knitr`.

[18] Dejan Živković. *The verbments package: Pretty printing source code in LATEX.* `ctan.org/pkg/verbments`, 2011.

◇ Geoffrey M. Poore
1050 Union University Dr.
Jackson, TN 38305
`gpoore (at) gmail dot com`
`https://github.com/gpoore/`

Development of an e-textbook using LaTeX and PSTricks

David M. Tulett

Abstract

For a course on decision modeling (linear, integer, and goal programming, networks, and decision trees) I created an e-textbook using LaTeX and PSTricks. I will discuss why I chose these programs, how I used them, and why I decided to make the final product "open access". The full PDF can be downloaded from `http://stor.mun.ca/handle/123456789/37463`.

1 Introduction

A wide-sweeping curriculum change in the undergraduate business programs at Memorial University necessitated the development of many new courses, including Business 2400, Decision Modeling. The introduction of the new courses was staggered, beginning in 2010, with Business 2400 first being offered in Fall Term, 2011. Here is the course description from the 2011–2012 university calendar:

> **2400 Decision Modeling** provides an introduction to: spreadsheet modeling; linear optimization and the related topics of integer, assignment, and transportation models; and decision analysis including payoff matrices, decision trees, and Bayesian revision. All topics will be taught within the context of business applications. [4]

This new course replaced the former Business 4401 and it differed in the following ways:

1. Business 2400 comes one year earlier in the curriculum than Business 4401 did. Since 4401 (and 4500 Finance) acted to "weed out" students from the program, it was felt that 2400 should come earlier than 4401 did.
2. Business 2400 continued the de-emphasis of the learning of algorithms that began when Business 4401 was created 15 years earlier. The only algorithms taught in Business 2400 are: solving two-variable optimization problems graphically; the graphical method for solving the minimal spanning tree problem; and the rollback procedure for solving decision trees.
3. As the name implies, there is now an increased emphasis on the modeling of problems. In most cases this means defining a set of variables, and then stating an objective function to be maximized or minimized subject to a set of constraints.
4. For the solution of formulated models, only Excel is used. All students at Memorial receive a copy of Microsoft Office, which includes Excel, hence there is no new software to download. Also, learning Excel has to be done anyway, so the only new things would be some specific mathematical functions and the use of the Solver.

The author is grateful for funding from a Teaching Fellowship to attend the 2016 TUG conference.

When the course was first offered in the Fall Term of 2011, we adopted a textbook for it: *Managerial Decision Modeling with Spreadsheets* by Render *et al.* [5] At the time, the book sold in the campus bookstore for about \$156. (All monetary figures in this article are in Canadian currency.) Alternatively, it could be obtained as an e-book for about \$70, but this comes with only a six-month licence.

Even though we had adopted a textbook, before the course had even begun I had started to write a document with the course title *Decision Modeling*. Principally, this was because the course was about to be offered in Winter 2012, by what was then known as "distance education" (as of 2016, this is now called *online learning*). For the students in this course, my document would substitute for the lectures that they would miss by not being on-campus. Also, I felt that I could improve upon the textbook's coverage of some topics, such as decision analysis (the use of decision trees). Five years later, the *Decision Modeling* document has become an open access stand-alone e-textbook, and the rest of this article describes how this happened.

In section 2, I describe the beginning of the writing: why LaTeX was chosen, merging of material from earlier courses, and development of new material. In section 3, I describe what I found useful when I was presented with options for completing various tasks in writing the document. Finally in section 4, I describe why this document became open access.

2 Initial development

I have been using LaTeX since the early 1990s, so I am very familiar with it and its graphical cousin, PSTricks. I am also very familiar with Microsoft Word, it being the standard for word processing where I work. Indeed, I use Word anytime that I need to collaborate with a co-worker, be it for administrative purposes or for writing a journal article. Perhaps if the writing of *Decision Modeling* had been a collaborative effort by multiple professors the document would have had to be written using Word. However, I was doing it on my own, so the choice was mine.

From previous courses I had amassed a large amount of material, mostly written in LaTeX, which could, with modification, be embedded within the new document, partially fulfilling its requirements. For this reason alone it would have made sense to create the *Decision Modeling* document in LaTeX, but there are two other main reasons.

1. LaTeX looks better than Word. This is especially true when creating mathematical expressions.
2. Because things like section numbers are never entered by the user in LaTeX, it becomes much easier to move things about, letting LaTeX figure out how everything (sections, figures, tables, footnotes) is to be renumbered.

All this being said, there is one important development in LaTeX that was needed to make the finished product useful for distribution as an e-document, and that is the creation of *pdfLaTeX*. Back in the 1990s, a `.tex` file was compiled to create a `.dvi` file. For anyone with a TeX system, all one had to do was print the `.dvi` file. However, members of the general public could not be expected to have this ability, and even when stand-alone free `.dvi` readers came out, it was still a barrier to have to expect people not interested in LaTeX to download the reader. Once the ability to easily compile a `.tex` file to a PDF became available, it was a major step forward. While Microsoft Office is a standard piece of software where I work, once a document has been converted to PDF its original source becomes irrelevant, be it Word, WordPerfect, Libre Office Writer, or LaTeX.

Hence, for all these reasons, it made sense to create this document using LaTeX for conversion to PDF. As I have mentioned, Excel is used as the prominent method of solution. For any parts of the document which used Excel, nearly everything had to be written from scratch. In other places I was able to import previously created material, but even here substantial alterations had to be made. The alterations were of two main types:

1. In cutting-and-pasting from several source documents, I was greatly aided by LaTeX's logical design, which frees the user from worrying about the numbering of sections, figures, tables, and footnotes. However, the text had to be extensively edited for phrases such as "as we saw in the previous chapter", which would probably now be an anachronism. Also, there needed to be consistency about notation, avoiding, for example, x_1 in one place while using X_1 in another. More subtle than these things, though, would be the existence of ideas which presume a knowledge which might not apply to readers of the new document.
2. Some of the older documents were based on earlier technology, such as the use of LaTeX's native picture environment (rather than the far more sophisticated PSTricks). Over time, these things have been updated.

At the outset of the Fall of 2011, the *Decision Modeling* document consisted of the following chapters, with much content to be completed:

1. **Introduction** Back in 2011, this was a very short chapter, consisting of a look at the paradigm of problem identification, modeling, solution, and implementation; a review of some key Excel functions; and a brief look at professional associations.
2. **Elementary Modeling** This chapter essentially replicated one used in Business 4401, so this required little work.
3. **Applications of Linear Models** Many of the examples in this chapter had been used in previous courses, but they had never been solved in Excel.
4. **Sensitivity Analysis** Most of the concepts had been seen before, but the pictures to illustrate these concepts had been created in the native LaTeX picture environment rather than PSTricks. Also, all computer output needed to be done in Excel.
5. **Network Models** These models are for the assignment, transportation, transshipment, minimal spanning tree, maximal flow, and shortest path problems. The minimal spanning tree problem has a very easy graphically-based algorithm for its solution. For the other five problems, this course stresses the use of Excel, in contrast to previous courses in which these problems had purpose-built algorithms. Because of this, this chapter needed to be written almost from scratch. Nothing had been written on this chapter as of September 2011.
6. **Integer Models** Much of the theory had been described in material written for previous courses, but none of the models had been solved in Excel.
7. **Goal Programming and Nonlinear Models** Same problem as the previous chapter: theory, but no Excel.
8. **Decision Analysis I** Mostly done already, but some Excel work required.
9. **Decision Analysis II** Mostly done already, but some Excel work required.

In addition to the work that needed to be done to complete all these chapters, another requirement was to create end-of-chapter problems for student completion, and to make the solutions for these problems.

3 Making the document

The Fall Term proceeded with the Render *et al.* textbook, on which all course assignments were based. As a supplement, students in my sections could download a PDF then called *Business 2400 Decision Modeling Course Manual* (a document title spread over three lines) which had no Chapter 5, and with the problems noted above in the other eight chapters. An immediate first priority was to have something written for Chapter 5. One of these models is the Assignment Problem, for which a small example is presented here, along with its mathematical model and its spreadsheet model.

3.1 Assignment Problem example: Assigning 3 jobs to 3 machines

Suppose that we have three jobs, and three machines on which these jobs will be done. Each machine will do just one of the three jobs. All three machines are capable of doing each job, but with differences in performance. We can think of these differences in terms of cost (which could be time rather than dollars). Suppose that the costs to assign each job (row) to each machine (column) are as follows:

		Machine		
		1	2	3
	1	30	20	18
Job	2	17	40	21
	3	25	32	28

By inspection, the minimal cost solution is to assign job 1 to machine 2, assign job 2 to machine 1, and assign job 3 to machine 3, for a total cost of $20 + 17 + 28 = 65$. To solve this as a mathematical model, we define the meaning of $X_{i,j}$ for all pairs (i, j):

$$X_{i,j} = \left\{ \begin{array}{ll} 1 & \text{if job } i \text{ is assigned to machine } j \\ 0 & \text{otherwise} \end{array} \right\}$$

$$i = 1, 2, 3 \qquad j = 1, 2, 3$$

The reason for using the numbers 1 and 0 becomes clear when we write the model. For example, if job 2 is assigned to machine 3 (i.e. $X_{2,3} = 1$), then the cost is $21(1) = 21$. If job 2 is *not* assigned to machine 3 (i.e. $X_{2,3} = 0$), then the cost is $21(0) = 0$. Hence, whether or not job 2 is assigned to machine 3, we incur a cost of $21X_{2,3}$.

Hence the objective function is:

$$\begin{aligned} \text{minimize} \quad & 30X_{1,1} + 20X_{1,2} + 18X_{1,3} + \\ & 17X_{2,1} + 40X_{2,2} + 21X_{2,3} + \\ & 25X_{3,1} + 32X_{3,2} + 28X_{3,3} \end{aligned}$$

Every job must be assigned to a machine, hence for each job i one of $X_{i,j}$'s will be 1 (and the other two will be 0), hence the sum will be 1:

$$\begin{aligned} X_{1,1} + X_{1,2} + X_{1,3} &= 1 \\ X_{2,1} + X_{2,2} + X_{2,3} &= 1 \\ X_{3,1} + X_{3,2} + X_{3,3} &= 1 \end{aligned}$$

Every machine must have a job assigned to it, hence for each machine j one of $X_{i,j}$'s will be 1 (and the other two will be 0), hence the sum will be 1:

$$\begin{aligned} X_{1,1} + X_{2,1} + X_{3,1} &= 1 \\ X_{1,2} + X_{2,2} + X_{3,2} &= 1 \\ X_{1,3} + X_{2,3} + X_{3,3} &= 1 \end{aligned}$$

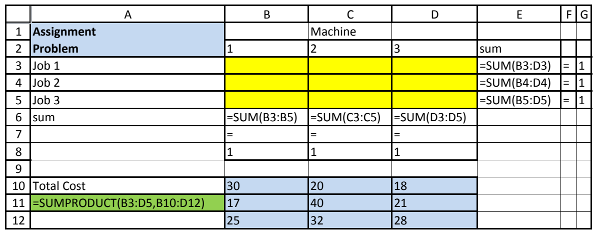
Finally, each variable must be 0 or 1.

$$\text{all } X_{i,j} \in \{0, 1\} .$$

In one sense this is a specialized type of linear programming problem, but it seems to violate one of the assumptions of linear programming — that all variables be continuous, rather than integer. However, it turns out that the assignment problem is naturally integer. By this, we mean that the solution will only contain 0/1 variables, even when these have not been specifically required. Hence, any software for general linear programming will solve an assignment problem.

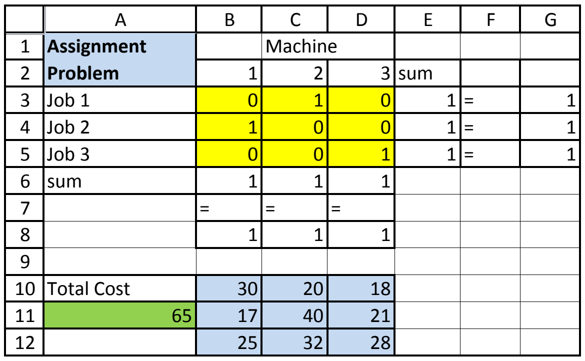
We will use the Solver in Excel to solve this type of problem. Indeed, the rectangular array paradigm of Excel is very useful for this type of problem, where the cost data is in this format in the first place.

Since the cost data are in a 3 by 3 array, we can also use a 3 by 3 array for the values of the variables. Note that the SUMPRODUCT function is happy with this; here it's an array times an array on a cell-by-cell basis, not the dot product of one row with another row. In this example it's B3 times B10 plus C3 times C10 and so on up to D3 times D12. This type of product is not the same as matrix multiplication. Here is the setup in formula mode on the spreadsheet, before entering the Solver (the = signs in row 7 and column F are created by typing `'=`).



	A	B	C	D	E	F	G
1	**Assignment**		Machine				
2	**Problem**	1	2	3	sum		
3	Job 1				=SUM(B3:D3)	=	1
4	Job 2				=SUM(B4:D4)	=	1
5	Job 3				=SUM(B5:D5)	=	1
6	sum	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)			
7		=	=	=			
8		1	1	1			
9							
10	Total Cost	30	20	18			
11	=SUMPRODUCT(B3:D5,B10:D12)	17	40	21			
12		25	32	28			

In the Solver we ask it to minimize A11 by changing variable cells B3:D5, subject to the three constraints B6:D6 = B8:D8, and the three constraints E3:E5 = G3:G5. We click on the "Make unconstrained variables non-negative" box, and ask for the problem to be solved using the "Simplex LP". Solving the model we obtain:



	A	B	C	D	E	F	G
1	**Assignment**		Machine				
2	**Problem**	1	2	3	sum		
3	Job 1	0	1	0	1	=	1
4	Job 2	1	0	0	1	=	1
5	Job 3	0	0	1	1	=	1
6	sum	1	1	1			
7		=	=	=			
8		1	1	1			
9							
10	Total Cost	30	20	18			
11	65	17	40	21			
12		25	32	28			

As we saw earlier, we see from the Solver output that the minimal cost solution is to assign job 1 to machine 2, assign job 2 to machine 1, and job 3 to machine 3, with a total cost of 65.

3.2 Typesetting the above in LaTeX

The above description uses the `tabular` environment for the cost data, and the `array` environment for the objective function and equations. When it comes to doing the Excel output, it would have been possible to use the `tabular` environment to mimic a spreadsheet, including the use of colour, but this would have been slow and cumbersome. Instead, I made the file in Excel, highlighted the range that I wanted printed, went to File/Print, set the Printer to Adobe pdf, made the Settings "Print Selection" and "Fit All Columns on One Page", and under Page Setup, then Sheet, then Print, clicked the boxes for "Gridlines" and "Row and column headings". I named this file `Assignment2.pdf`, then using Adobe Acrobat, this file was cropped and saved. It was imported into the book's `.tex` file as:

```
\begin{center}
\includegraphics[scale=0.9]{Assignment2.pdf}
\end{center}
```

This file needed to be re-scaled to 90% of its original size in order not to spill too much into the margin. This re-scaling is a visual trial-and-error process. It wasn't needed for the final numerical workbook image:

```
\begin{center}
\includegraphics{Assignment3.pdf}
\end{center}
```

When importing Excel files in this manner, the ribbon with its words (File, Home, Insert, etc.) and all the icons does not appear. Normally, I think that this is to be preferred — it emphasizes that the most important part of a workbook is what lies in the rows and the columns. If including the ribbon is desired, we can use the Print Screen command and then import this file into Adobe Acrobat. In my document, I do this only once at the outset, to show what the ribbon looks like.

With this section on the assignment problem, plus other sections for the other network models, I finally had a first draft of Chapter 5 completed before the outset of the Winter Term, 2012. Next came the inclusion of dozens of PDFs created by cropping files created in Excel for the many problems covered throughout the document. A major piece of work remained until 2015, the creation of the graphical images needed for Chapter 4 (Sensitivity Analysis). The earlier work on which this chapter was based included graphics made in the native LaTeX `picture` environment. While this environment was better than nothing, it was very inadequate. Everything was in black-and-white, with very limited ability to draw curved lines. Even straight lines were limited to horizontal and vertical lines, and lines whose rise over the run could be expressed as $a : b$, where a and b are integers from 1 to 6 inclusive.

The way to improve the graphics is to use better software, such as PSTricks, which like LaTeX can be freely obtained from TUG. A good introduction is provided in Chapter 5 of *The LaTeX Graphics Companion* by Goossens *et al.* [3] A complete description of PSTricks is contained in *PSTricks: Graphics and PostScript for TeX and LaTeX* by Voß. [8]

At the outset of Chapter 4, a problem is stated and solved. Here follows the problem description, model formulation, and graphical solution.

3.3 A two-variable example

3.3.1 Problem description

Wood Products Limited buys fine hardwoods from around the world from which they make specialized products for the quality furniture market. Two of their products are two types of spindles.

A type 1 spindle requires 6 cuts, then 4 minutes of polishing, followed by 6.5 minutes of varnishing. A type 2 spindle requires 15 cuts, then 4 minutes of polishing, followed by 4.75 minutes of painting. There is one cutting machine which can operate up to 135 cuts per hour. There is one polishing machine; allowing for maintenance it can operate up to 54 minutes per hour. Both the varnish and paint shops can only handle one spindle at a time. Because of a periodic need for high volume ventilation, the varnish and paint shops cannot be operated continuously. These shops are available for production 58.5 and 57 minutes per hour, respectively.

For each type 1 spindle produced, the company obtains a contribution to profit of \$3. For each type 2 spindle produced, the contribution to profit is \$4. How many spindles of each type should be produced each hour so that the total contribution to profit is maximized?

3.3.2 Model

We define:
X_1 — number of type 1 spindles produced per hour
X_2 — number of type 2 spindles produced per hour.

For reference, each constraint is identified by a word description on the left-hand side, and by a number in brackets on the right-hand side.

$$\begin{array}{rrcrcrl} \text{maximize} & 3X_1 & + & 4X_2 & & & \\ \text{subject to} & & & & & & \\ \text{Cutting} & 6X_1 & + & 15X_2 & \leq & 135 & (1) \\ \text{Polishing} & 4X_1 & + & 4X_2 & \leq & 54 & (2) \\ \text{Varnishing} & 6.5X_1 & & & \leq & 58.5 & (3) \\ \text{Painting} & & & 4.75X_2 & \leq & 57 & (4) \\ & X_1 & , & X_2 & \geq & 0 & \end{array}$$

3.3.3 Graphical solution

Because of the two 4's in the polishing constraint, this constraint will be on a diagonal. Since it's $\leq$, the arrow indicating feasibility will point south-west. So, since $54/4 = 13.5$, having a 14 by 14 grid must contain the optimal solution. Using these boundaries, we obtain the values in Figure 1. The graph (displaying both numerical labels and the names of the constraints) is shown in Figure 2.

We see that constraints (1) and (2), i.e. the cutting and polishing constraints, are binding. The equations we need to solve are:

$$\begin{array}{rcl} 6X_1 + 15X_2 & = & 135 \\ 4X_1 + 4X_2 & = & 54 \end{array}$$

Multiplying the second equation by $6/4 = 1.5$ we obtain:

$$\begin{array}{rcl} 6X_1 + 15X_2 & = & 135 \\ 6X_1 + 6X_2 & = & 81 \end{array}$$

Subtracting the bottom from the top gives $9X_2 = 54$, and hence $X_2 = 6$. Therefore $4X_1 + 4(6) = 54$, hence $4X_1 = 30$, and therefore $X_1 = 7.5$. Putting $X_1 = 7.5$ and $X_2^* = 6.0$ into the objective function we obtain $\text{OFV}^* = 3(7.5)+4(6) = 46.5$. The 7.5 type 1 spindles per hour simply means that we must produce 15 of them every two hours, hence the fractional solution is not of concern.

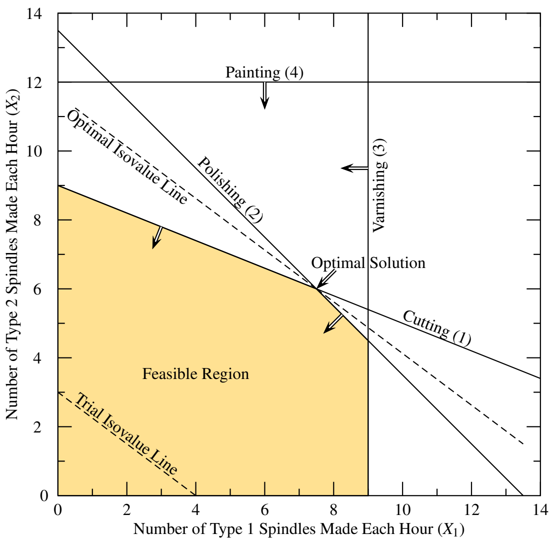


Figure 2: Spindle problem, graphical.

3.4 Doing the above in PSTricks

Everything in the preceding section is easy until we come to the graph. What we need is to be able to: draw straight lines at any angle; write text next to these lines at the same angle; draw arrows at right angles to these lines indicating which side is true for the inequality; fill in the polygon which represents the region in which all inequalities are true. PSTricks gives all these things, though a bit of geometry/trigonometry is needed to make it all work properly. Here are some of the special features of this picture, showing the relevant PSTricks code:

1. We have some lines which are neither horizontal nor vertical. If we take the Cutting constraint as an example, we have identified that the boundary of the inequality, which is the line $9X_1 + 15X_2 = 135$, passes through $(0, 9)$ (on the vertical axis), and $(14, 3.4)$ (on the right-hand side boundary). I found it useful to plot points as integers, so all algebraically determined points were multiplied by 100, making these coordinates (0,900) and (1400,340), but then to use a scaling command in PSTricks to make the graph for the page properly. For this graph, I

							First Point	Second Point
Cutting	$6X_1$	+	$15X_2$	$\leq$	135	(1)	$(0, 9)$	$(14, 3.4)$
Polishing	$4X_1$	+	$4X_2$	$\leq$	54	(2)	$(0, 13.5)$	$(13.5, 0)$
Varnishing	$6.5X_1$			$\leq$	58.5	(3)	$X_1 = 9$	vertical
Painting			$4.75X_2$	$\leq$	57	(4)	$X_2 = 12$	horizontal

Figure 1: Spindle problem, numerical.

used a quarter scale, using the PSTricks command `\psset{unit=0.25pt}`, and then created the line using the PSTricks command

```
\psline(0,900)(1400,340) %(Cutting)
```

2. The set of points for which all inequalities are true is called the *feasible region*. To identify this region, it is highlighted in colour with the built-in paint program in PSTricks. The problem is, we need to give PSTricks the set of coordinates which gives the vertices ("corners") of the feasible region. This requires doing some successive solving of two equations in two unknowns to determine these points. They are the origin at (0,0); then where the vertical axis meets the boundary of the cutting constraint, which is (0,9); then where boundary of the cutting constraint meets the boundary of the polishing constraint which is (7.5,6); then polishing and varnishing at (9,4.5), then varnishing and the horizontal axis at (9,0). I defined my own colour using the PSTricks `\definecolor` command, which I named marygold. (I know that the flower is named *marigold*, but I named this colour after my wife, whose name is Mary.) Therefore the code includes

   ```
   \definecolor{marygold}{cmyk}
       {0,0.1,0.5,0}
   ```

   and

   ```
   \pspolygon
     [fillstyle=solid,fillcolor=marygold]
     (0,0)(0,900)(750,600)(900,450)(900,0)
   ```

3. We want to write a label next to each constraint, so that the label is parallel with the constraint. This requires recalling trigonometry from high school. For the cutting constraint, the inequality is $6X_1 + 15X_2 \leq 135$, hence the rise over the run of the boundary is -6 over 15, or -0.4. To find the angle that the boundary makes with the horizontal axis we need to find the arctangent of this number. PSTricks uses degrees, but Excel's ATAN function uses radians, hence we need to use the DEGREES function as well. The Excel command is therefore `=DEGREES(ATAN(-6/15))` which is rounded to -21.801.

   A related issue is the determination of the coordinates for the centre of the expression "Cutting (1)". For me, this was accomplished by trial-and-error. It isn't just a matter of making the words close but not too close to the line; the words need to be placed so that they don't interfere with other lines or words.

Each diagram done in PSTricks requires a lot of work, with many iterations to get everything right. Each iteration requires `filename.tex` $\Longrightarrow$ `dvips` $\Longrightarrow$ `ps2pdf`. Because of all the work for each diagram, I created each diagram with its own file. When I was satisfied with the final PDF, I cropped it in Adobe Acrobat, and then used an `\includegraphics` command in the main file.

3.5 Continuous improvement

Until recently, the file, now called `DecisionModeling.pdf`, was updated at least once every four months. At the outset, references were made to Excel 2010, then Excel 2013, and very recently Excel 2016. That being said, I tried to make all references to Excel to be as version-free as possible, showing the rows and columns without the ribbon.

Other improvements include: adding new examples; adding more problems to the section in each chapter on "Problems for Student Completion"; and making pedagogical improvements to the writing when students needed more explanation about a topic. Any new work is prone to typographical or other kinds of errors, so each revision provides an opportunity to make the necessary corrections.

4 Making the document open access

4.1 Introduction

My university uses a student shell called D2L ("Desire to Learn") which is used for storing documents, submitting assignments, and storing grades. It is used always for online learning courses, and at the professor's option for on-campus courses as well. Access to the course-specific part of D2L is restricted to those with valid course registrations. Hence, a document can be released to the students by posting it to D2L, without making the document publicly accessible.

It was obvious from the outset that this document should be made available to the students free-of-charge. It was, after all, a work-in-progress, rather than a finished product. Also, I think there would be a conflict-of-interest question in a professor adopting his own self-made publication and then collecting royalties from students. Nevertheless, during this period of development, the document contained the words "© David M. Tulett", because I didn't want anyone else to claim my work as their own, and I didn't know what else I could have written in this regard. I did, however, put a note in the course outline to the effect that any student who wanted to have the document printed in a copy shop had my permission to do so.

Once the document had been completed, I had to think about whether I should try to publish it as a commercial venture or let it be freely accessible to anyone who wanted to read it. I chose the latter, for the following reasons:

1. The content of the book was written specifically for the needs of Business 2400 at Memorial University. Other universities might consider a topic such as queueing theory, or simulation, or the study of algorithms, to be essential. Because of this, textbook publishers want manuscripts which are inclusive of all topics in the field, which is why book lengths often reach a thousand pages. To pursue this, I would have to complete much more material.
2. I am unhappy with what has happened to textbook prices. The textbook mentioned earlier now costs about \$190. There are plenty of other textbooks costing well over \$200. Most students find these prices hard to afford.
3. The existence of free material on the web helps learning.
4. Many prefer to help the public good rather than seek royalties. The TeX community is a good example of this. I can use LaTeX and PSTricks without paying a fee because of the generosity of those who donated their time and effort to create and maintain these programs. In the software world, this is called "Open Source". There are all sorts of things freely available: alternative suites to Microsoft Office; games such as chess; programming languages, and so on.

4.2 Open Access and Creative Commons licences

In a conversation with Jeannie Bail [1], a librarian at the Queen Elizabeth II Library at Memorial University, she informed me that for written material the equivalent of open source is "Open Access". She went on to say that I should read about a major organization acting for the public good in this area, called *Creative Commons*, `http://creativecommons.org`. [2] They offer free licences which are described on their website. The most restrictive licence allows anyone to freely download the material but places the following restrictions (quoted verbatim from the Creative Commons website):

1. Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
2. NonCommercial — You may not use the material for commercial purposes.
3. NoDerivatives — If you remix, transform, or build upon the material, you may not distribute the modified material.

The above is a quick summary of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International Licence. The full text is at `http://creativecommons.org/licenses/by-nc-nd/4.0/`.

A reader of `DecisionModeling.pdf` is informed of this licence by the following graphic:

BY NC ND

This image appears on page i, which follows the title page. Also, the word *copyright* and the copyright symbol © now no longer appear in the document.

4.3 Public availability

Memorial University encourages open access, as it helps to promote learning, and makes a website available for this purpose. For anyone who simply wants to explore what's there, the address is `http://stor.mun.ca/`. On the website the purpose of stor is described:

> stor is a learning object repository that aims to promote an atmosphere of sharing where learning objects can be searched, reused, repurposed and contributed. A learning object is a digital, open educational resource that is created to assist in a learning event. [6]

The edition of September 1 2015 was placed online at this site in March 2016. More recently `DecisionModeling.pdf` has been amended, referencing Excel 2016 instead of Excel 2013, and uploaded to stor in June 2016. [7] Here is a direct link to the current `DecisionModeling.pdf`: `http://stor.mun.ca/handle/123456789/37463`.

With the document now in a public repository, I can inform someone of its existence by sending an email message, with a link to the document. Before this, I would have had to attach the file, which at 8 MB is rather large. I now need to make fellow educators in the field of Decision Modeling aware of the document's existence. I thank the TeX community for helping to make this possible.

References

[1] Jeannie Bail, Librarian at the Queen Elizabeth II Library, Memorial University, St. John's, NL, Canada. Personal communication, 2015.

[2] Creative Commons, `creativecommons.org`, accessed June 2016.

[3] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voß, *The LaTeX Graphics Companion*, 2nd edition, Addison-Wesley, 2008.

[4] Memorial University, 2011–2012 *Calendar*. `http://www.mun.ca/regoff/calendar/2011_2012/sectionNo=BUSI-0288`.

[5] Barry Render, Ralph Stair, Nagraj Balakrishnan, and Brian Smith, *Managerial Decision Modeling with Spreadsheets*, 2nd Canadian Edition, Pearson Canada, Toronto, 2010. ISBN 978-0-13-208013-2, HD30.25.M35, 2010. `http://wps.pearsoned.ca/ca_ph_render_mdm_2/`.

[6] `http://stor.mun.ca/`, accessed June 2016.

[7] David Tulett. `DecisionModeling.pdf`, June 2016. Available at: `http://stor.mun.ca/handle/123456789/37463`.

[8] Herbert Voß, *PSTricks: Graphics and PostScript for TeX and LaTeX*, UIT, Cambridge, England, 2011.

◇ David M. Tulett
Faculty of Business Administration
Memorial University
St. John's, NL, Canada, A1B 3X5
dtulett (at) mun dot ca

An Emacs-based writing workflow inspired by TeX and WEB, targeting the Web

Christian Gagné

Abstract

I present here a practical method developed with colleagues working in the humanities, whereby they produce content using macro-less notations (such as Markdown), which I integrate and publish on the Web by using macro-rich notations such as Emacs Org-mode and TeX. Over time, this method's general applicability has caused me to entertain a notational pipe dream: a minimalistic substitution syntax, suitable for content work in any field (technical or otherwise), which would yield both TeX on the one hand, and on the other hand HTML and XML styled with TeX-equivalent CSS.

1 Introduction

When using Emacs, there are at least four different things called 'macros' — and I have come to use all four in my work as a content integrator: Emacs Lisp macros, keyboard macros recorded in a non-Lisp expression language, Org-mode lexical macros and TeX macros written with AUC-TeX. The Org lexical macros turn out to be especially useful for one who is used to TeX and needs to produce HTML: they bring some of the power of writing in `WEB` to the Web.

The following concerns a methodology developed over the course of my work as a research and teaching assistant in a literature department, with influence from my concurrent work as a teaching assistant in computer science, hence the mix of developer-friendly and user-friendly solutions.

Some background will be relevant in order to explain my methodological choices: during my philosophy studies (which have both continental and analytic components), I was lured into a Medieval Studies research unit and became their "IT guy". At that point I was already a confirmed TeX user and my new medieval interest involved taming the XML beast. I happily obliged; however, the road ended back at a distinctly TeX-flavored abode.

My first job was to teach my literary colleagues to write in a format other than Word or LibreOffice. Given that I love the interplay of theory and practice, I organized workshops for them as I pursued my gradual discovery of theoretical computer science, philosophy of language and of what Edward Tufte calls *analytic design* [11].

It was also in the same school year that I began using Emacs in earnest, printing out reference cards and practicing proper buffer movement. What led me to Emacs was first AUC-TeX, then Org. With AUC-TeX, I first realized how much the dynamics of writing matter. Here I was teaching my colleagues to write Markdown and TEI XML by hand, just as I had begun to discover the joys of writing with macros! However, when I tried to introduce this into their workflow, it seemed to them one step too far: they said they preferred writing XML by hand, and were intimidated by transformations and macro expansions of any kind. Such is one of the reasons behind the hybrid workflow we adopted for our research unit's Web site [6].

While I studied the many alternatives for implementing the TEI guidelines and transforming TEI into other formats, I came across the name Sebastian Rahtz many times. His texts, code examples and generally remarkable implication in such matters set important precedents and helped me bridge the gap between the TeX and XML mindsets. As such, I am very thankful for the pioneering work of Sebastian Rahtz and others in the field of semantic transcription. Below, I will refer to an example of the way I interpreted the *Text Encoding Initiative Guidelines* [2]. I have gotten much mileage out of the Guidelines, but in an unorthodox way.

As it turns out, the way I have used TEI ties in beautifully with some recent wide-ranging efforts in the Web design community to integrate solid information architecture into the use of markup, in order to make it both meaningful and aesthetically pleasing. In other words, I believe that some of those very same 'semantic markup' techniques used in an academic setting will be relevant for Web content work in any field, be it technical, scientific or creative.

2 Flexible delimiters for Web content work

Org's markup syntax is very rich and parses into intricate structures. One of my favorite stated principles from the syntax specification [9] is that 'the paragraph is the unit of measurement'. An order of magnitude higher, many nesting constructs exist. For example, lines beginning with stars do make trees, properly speaking:

```
* Section title example in Org

** This is a subheading --
   and a node in the tree
```

Markdown and Org share this ability to create implicit tree structures through the use of headings. This is applicable in many situations. However, for my scenario I needed to name my content blocks using native HTML5 vocabulary and I could not commit to regular section titles that would convert to `h1` ... `h6` elements. The reason was that I had to make content blocks that would nest arbitrarily: they had to be *closed under inclusion*, one might say (though Web developers rarely speak this way). Web people call this working *content-out* instead of *canvas-in*, after the expressions consecrated by information architects' discussions at Web sites such as *A List Apart* (`alistapart.com`) and *Boxes and Arrows* (`boxesandarrows.com`). The following example is typical of what was needed on our Web site:

```
#+begin_section
#+attr_html: :id about-blocks
             :class mt_sectitle
#+begin_header
How block-delimited lines become paragraphs,
which are at the syntactic level
of mt_concept(elements) in Org parlance
#+end_header

The parsing rules dictate that, because Org
is very much a mt_emphasis(line-based) format
(in its surface guise), paragraphs are created
inside the special blocks as expected.

As for the lexical macros, they are expanded
at the very beginning of the export process.
#+end_section
```

What do we have in this Org example? Two nested special blocks which will be converted to the appropriate HTML5 or LaTeX constructs, as appropriate, and also some Org lexical macros similar to those of the C preprocessor, m4 or `WEB`. Here the lexical macros are shown in their fontified form, a feature added in recent Org versions. Under the fontification hood, there are actually three pairs of braces around the whole macro invocation! The block is much more readable when the braces are removed by fontification, and the macro objects are also syntax-highlighted.

Org is line-based, contrary to token-based macro engines such as m4. In this, Org is closer to Markdown, in which blank lines are a staple of the syntax. In both Org and Markdown then, the document is delimited into blocks and inlines [8]. This is important for ergonomics, since it reveals document structure graphically. This is also one of TeX's strong points, and probably one of the many reasons why TeX is often deemed easier to learn than the SGML family. In fact, though my colleagues preferred writing their TEI XML by hand, I allowed them to write all other content types in Markdown, freely interspersed with HTML as they needed. Upon reception, I processed their documents with Pandoc, then *refactored* them, so to speak, in Emacs. Whenever I wished to integrate multiple complex sources, Org was my intermediate language of choice.

The `begin` ... `end` pairs in Org have extra power, making them invaluable for producing SGML-style markup: they can conditionally write attributes in the resulting element tags. This can be used to add *classes* for CSS selection, identifiers, target URI's, microformats or even Resource Description Framework statements. In the above example, `id` and `class` attributes will be added to the `header` block upon HTML5 export. If the export target is LaTeX, simple environments will be produced that bear the names of the Org special blocks. This affords one the occasion to write some sophisticated definitions for said environments, perhaps going as far as to reproduce a given CSS layout. It is still an open question for me what exact set of packages and commands one would need to adequately reproduce some of the more idiomatic CSS stylings, for example `relative`, `absolute` and `fixed` element positioning — and the true holy grail consists of that plus a TeX implementation of *Flexbox*!

The above example is used in a Web demo [4], with some typeset examples, which acts as a companion piece to the present article.

3 Applying WEB-style substitutions to Web content

The constructs nested inside the element blocks, namely the macro objects, have an `mt_` prefix meaning 'multi-target', since the macros define different substitutions for different target formats. In the accompanying Web demo, examples are given for an HTML rendition, plus an SVG rendition produced with a TeX engine and an SVG converter.

The initial motivation for publishing SVG text blocks typeset with TeX comes from the Medieval Studies project: I wanted to produce facsimile excerpts of the manuscript transcriptions. I found that the best way to reproduce the beautiful hand-written text was to procure a historically-accurate font with many alternates and make a proof-of-concept example by manually adding OpenType substitutions. This proof-of-concept facsimile is available at [3]. The XeTeX engine was used with the `standalone` class and the `fontspec` package. Because the Web browsers cannot at present be trusted with such complex typography, I used the Poppler tools to convert the PDF to SVG with all text converted to paths.

In [1, p. 45], Robert Bringhurst mentions the rich textures that Renaissance typographers achieved with a single type size and hand-drawn additions. This same 'sensuous evenness of texture' is very much present already in fifteenth-century manuscripts such as those reproduced on our Web site, and which the early Renaissance typesetters duly imitated. In fact, this has a very material basis in the tools of calligraphy, for the broad-nib pen that dominated European writing until the Early Modern period had a more or less fixed width, depending on the pressure applied by the scribe. This width yields a fundamental tone, a stroke that serves as the basis for a scale. With the appropriate stroke modulation techniques, it allows the scribe to play within friendly neighboring scales, but the relative evenness of texture is to be seen as a blessing.

I strongly believe in the power of harmony and counterpoint in all media, so I always seek such effects, including with modern type. That is why I speak of my workflow as generally applicable to Web content work in any field, as long as the contributors care about aesthetics and believe in the dignity of the craft. In my facsimile, I have attempted to approximate the scribe's creative freedom by applying many OpenType substitutions, which recovers some of the rich graphical harmonics created by a steady and energetic hand. After many facsimile pieces have been created, one could create appropriate macros to group OpenType substitutions, thus creating a mini-macro package giving the flavor of a particular scribe's craft!

I must stress that the TEI documents referred to are very much fragments and are not validated in any way. In order to ease my colleagues into the workflow, I decided to treat anything they produced as relevant and to act upon the principle that they had good reasons for using whatever structures they had written. This remained scalable because only a dozen node types were allowed, including both elements and attributes. From the beginning, I had planned to write a Relax NG grammar when the process stabilized, but the production of this grammar remains an open ticket to this day. This has turned out to be a blessing, since it allowed my colleagues to inform the vocabulary themselves according to their needs, without any external pressure from a *pesky grammarian*. Again, the process was established from the bottom up, and this has turned out to be a boon. The graphical fidelity of the transcriptions is all the better for it: they are actually *readable*, which is a noteworthy milestone in and of itself.

In his presentation of the WEB literate programming system, Donald Knuth discussed how he chose to make his lexical macros as simple as possible, with only one parameter allowed, in a section entitled 'Occam's Razor' [7, p. 121]. The elegant simplicity of the approach summarized there is one of my chief inspirations in developing the present workflow and in forming my conviction concerning the general applicability of substitution in the act of writing. What is more, The Occam with whom originates that saying, being one of the greatest logicians of the Late Middle Ages, is most appropriately mentioned here, since it is with him and some contemporaries that the foundations were laid for a mathematical consideration of language items as discrete structures that can be manipulated algorithmically, as in the substitution techniques discussed here. I mention this in passing, o readers, knowing all too well that defending this statement properly would lead us down a much longer road which we will have to travel another time.

4 Equational inspiration

I have shown how the many aspects of macro substitution have shaped my experience of writing technical documents. Another theoretical development has led me to further consider macro-based writing as a most appropriate way of writing. This year, I have been introduced to equational logic through a computer science course given in my department. This course is based on the Gries-Schneider approach [5], which is in turn influenced by Edsger Dijkstra among others — and Leslie Lamport adopts a similar approach in his work. I read the course's introductory material and learned that equality is taken as the most fundamental relation and is *defined in terms of substitution*. Finding this epistemologically pleasing, I asked around whether substitution and the Leibniz rule, so defined, functioned as a theoretical base for macros in Lisp or TEX, or rather whether macros were effectively applications of the substitution principle, to which the reply was that, given the definitions we were handling, this was certainly the case. I then thought, with my usual inclination towards holistic conclusions, that surely this was yet another sign that macro languages are most appropriate to the very nature of *writing with a computer*.

5 Conclusion

In the end, even though I am presenting an Emacs-based writing workflow, what I am truly committed to is the epistemology of writing that emerges from all this and the algorithmic principles that make it possible in practice. That is why I am still trying to find better notations for writers. Recently, this has meant Web writers especially, as I strongly wish to see the riches of TEX-style composition being distributed liberally among as many people as possible, be they front-end developers, engineers, blog writers or designers. That is the meaning of my syntax essay at the end of the companion demo page [4]: reflecting upon ways to make macro-based writing more alluring to people who would traditionally never have thought of using Emacs or TEX. In a word, my wish is for Web writing to become more *organic*, both technically and culturally.

References

[1] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, 2005.

[2] TEI Consortium. *Text Encoding Initiative Guidelines*. Oct. 5, 2015. `http://www.tei-c.org/Guidelines`.

[3] Christian Gagné. *Miroer du monde: comparaison et mises en relation entre BnF fr. 684 et BnF fr. 328*. 2016–. `http://hu15.github.io/histoires-universelles-xv/miroir-du-monde/comp/comp_fr684-fr328.xhtml`.

[4] Christian Gagné. *Organic TEX Demo*. Aug. 2016. `https://waidanian.github.io/organic-tex-demo`.

[5] David Gries and Fred B. Schneider. *Calculational Logic*. `http://www.cs.cornell.edu/gries/Logic/intro.html`.

[6] Groupe de recherche HU15. *(H)istoires (U)niverselles 15*. 2016–. `http://hu15.github.io/histoires-universelles-xv`.

[7] Donald E. Knuth. Literate Programming (1984). *Literate Programming*. Center for the Study of Language and Information, 1992, pp. 99–136.

[8] John MacFarlane. *CommonMark Spec*. July 15, 2016. `http://spec.commonmark.org/0.26/#blocks-and-inlines`.

[9] Org Dev Team. *Org Syntax (draft)*. Apr. 6, 2016. `http://orgmode.org/worg/dev/org-syntax.html`.

[10] David Walden. "Macro memories, 1964–2013". *TUGboat* 35:1 (2014), pp. 99–110. `http://tug.org/TUGboat/tb35-1/tb109walden.pdf`.

[11] Mark Zachry and Charlotte Thralls. "An Interview with Edward R. Tufte". *Technical Communication Quarterly* 13:4 (2004), pp. 447–462. `https://www.edwardtufte.com/tufte/s15427625tcq1304_5.pdf`.

◇ Christian Gagné
`christiangagne (at) outlook dot com`
`https://waidanian.wordpress.com`

TeXcel? An unexpected use for TeX

Federico Garcia-De Castro

Abstract

I recently discovered the surprising fact that TeX seems to be more appropriate for keeping financial records, and especially preparing different kinds of reports (to funders, to the board, by season, by project, by calendar year), than the spreadsheets that I was using, and which had become highly convoluted as years of information accumulated. Here is a description of the task, the problems with the spreadsheets, and the incipient but already useful system I developed in TeX.

1 The problem

As the director of a contemporary chamber music company (`www.aliamusicapittsburgh.org`), I am in charge of designing and executing the budgets for individual projects and for whole concert seasons. And then reporting: to funders, board, government, throughout projects and after their completion.

In retrospect I see that it is this — the wide range of reports, each with its own format, budget lines and subdivisions — that reveals the inadequacy of any static-spreadsheet model for bookkeeping, and the counterintuitive fact that TeX's macro capabilities come in handy for financial tracking and reporting.

1.1 Ways of reporting

Different funders, and therefore different budget reports, focus on different things. Music foundations usually have budget formats that include items that are specific to concert production — things like performance licenses, that in more general purpose budgets go under something like "fees and dues" — while in typical art grant reports a concert's lighting design (a relatively straightforward expense) must be split into, for example, "equipment rental" and "other program professionals".

On the other hand, reports also vary by time scale and scope. Season-wide reports, for example, lump together all season ticket sales, while in project-specific reports for a concert's funder these need to be reported by project. On the other hand, the fees for online ticket sales, which are in principle project-specific expenses, in year-long budgets may better fall under "banking fees".

And so on: the assignment of each transaction to a particular budget line depends on the latter's purpose and format. In general, the only way of implementing this is to record all transactions in a report-independent database, from which each report gathers transactions into the appropriate lines, according to its own format and needs.

1.2 The spreadsheet model

I've long had such a database for Alia Musica, in the form of a spreadsheet:

	Season	Prj1	Prj2	…	Prj3
Donations					
Ticket sales					
⋮					
Grants					
Donations					
⋮					
Performers					
Guests					
⋮					
Ads					
Postcards					
⋮					

… and so on (although much more detailed). Columns specify information relevant to each project (or general season transactions in special season columns), while the rows were arranged following our most common report budget formats. Extra columns were used to gather season totals, for season-wide reports. In fact, since tax returns go by calendar year, which is different from the season year, these 'total columns' come in two flavors (one for each season, one for each calendar year); this also meant needing to have, for each season, two 'general season transaction' columns — one for the part of the season year that fell in one calendar year (September–December), the other one for the other one (January–August). In fact, this detail is just one of the many complications in the spreadsheet model. All of which eventually led me to look for an alternative.

Alia Musica's spreadsheet had been built and in use since around 2010, with ad hoc adjustments here and there as new needs emerged. But in the meantime the organization has changed and has grown significantly. In 2015–16, with two major productions (check out in particular the Pittsburgh Festival of New Music, `www.pghnewmusic.com`), the spreadsheet had become clearly obsolete — now, for example, overlaps happened not only among seasons and calendar years, but among the productions themselves — some parts of one being part of another, and so on.

Toward the end of the season and of all of those productions — i.e., at reporting time — I decided to update the system, and started re-designing an array of spreadsheets, more in tune with current needs.

I did not get far: the deficiencies of the spreadsheet model, as I found out, are structural, and not due to a particular design or implementation.

1.3 Deficiencies of the spreadsheet model

Dimensions: A spreadsheet is basically a table in two dimensions. At best, you can use a couple of tricks and count them as two additional pseudo-dimensions:

- The user can attach 'notes' to each cell. In these notes, a total can be split into several components. For instance, the total ticket revenue of a concert can be input into the cell, with a note saying how much was door revenue and how much was online sales.
- What a cell shows on the spreadsheet is the result of the formula contained in the cell. You can make use of this to record some extra information: "`$420`" can be input, say, as "`=0+320+(40+40)+20`". At Alia Musica we used this trick to locate transactions by month: the above would mean "$0 in January; $320 in February; two transactions for $40 each in March; and $20 in April".

Of course these tricks do not provide for real extra dimensions: they afford extra information, but the processing program has no access to it — e.g., reports have no way eventually to take only a subset of the components of the cell's total. The information can be recorded, but its actual use requires user intervention.

User-time decisions: Transactions are input into the spreadsheet at different times (as they happen, or, more systematically, at month-end). That entails the ever-present risk of inconsistencies: where did we put parking expenses — sometimes as meeting expenses, sometimes as travel expenses? What did we decide about the guest's payment, did we put all of it as "guest performer", or did we split it into "guest performer", "lecture honoraries", "per diem", or who knows what else?

Completeness and feedback: Assume, however, that the inconveniences above can somehow be worked around: that the information is all consistently and clearly input into the big repository. Now the reports "simply" have to gather the relevant cells from here and there, according to their design.

An inescapable deficiency of the spreadsheet model shows up at this point: there is no mechanism to ensure that *a*) individual cells are not counted wrongly in two or more report lines, for which they might both be relevant; or *b*) that all relevant transactions for a particular line are included — maybe there was one obscure one (an extra parking expense, an extra bank transfer fee) that doesn't come to mind when manually gathering transactions into the report.

1.4 A necessary component: tags

Such reflection on the deficiencies of a spreadsheet points to one necessary component of any satisfactory system: assigning tags to transactions. A complete description of a transaction should be enough for the system to be able to pull it into whatever budget line each report needs. This in effect implements an open number of dimensions, and it even works toward the problem of user-time decisions: nothing prevents a report to gather tags for both "Fall 2015" and "Fall 15", relieving the user from having to remember a rigid list of possible tags.

Tags are implemented in financial tracking programs. But there are still problems: using tags typically involves dialog boxes, saving buttons, scrolling through lists... And in any case we still have the main problem of a spreadsheet model: nothing checks for completeness or duplications when at a later time the transactions are gathered into reports.

2 TeXcel

After realizing that the problem with my spreadsheet was not my particular spreadsheet, but the spreadsheet model itself, I came to wonder, almost as an afterthought, whether this was a task for TeX. The intuition was strong that there would be many problems, but that in principle something like this would be workable:

```
\deposit: 45.50 (Ticket sales, Festival
   subscription, Spring 2016)
\deposit: 8320 (Foundation Grant, T.W. Dunns
   Char. Fund, Fall 15)
...
\expense: 400 (Performer, Spring 16, Festival)
\expense: 19.80 (Stamps, Office expense,
   Mailing)
\expense: 1.59 (Facebook, Online advertising,
   Fall 2015)
\expense: 950 (Booklet printing, Festival)
...
```

With such a database (in a database "document"), reports could then be requested (in different documents) through something like this for a season-wide report:

```
\begin{expenseline}{Marketing}
  \include{Poster distribution}
  \include{Poster printing}
  \include{Flyer printing}
  \include{Advertising}
  \include{Online advertising}
  \include{Booklet printing}
  \include{Booklet shipping}
  \include{Website}
  \include{Project website}
  \include{Postcards}
\end{expenseline}
...
```

Or for a project report:

```
\begin{expenseline}{Spring 2016}
  \include{Spring 2016}
  \include{Spring 16}
  \include{Sp 2016}
  \include{Sp 16}
\end{expenseline}
...
```

Indeed, this is the backbone of a system I have developed for financial tracking in TeX.

2.1 The basic \deposit and \expense

So, \deposit and \expense are at the base of the whole system. They do not really 'mean' anything by default: what exactly TeX does when it finds them depends on the task at hand, as explained in a moment. This flexibility, and in general TeX's ability to define anything as anything, is a key reason why TeX turns out to be an appropriate environment for financial tracking.

Thus, \deposit (\expense is fully analogous) is defined, at bottom, as follows:

```
\long\def\deposit: #1 (#2){%
    \ifreporting
        \ifnum\yearindex=\z@
            \@deposit: #1 (#2)\relax
        \fi
    \else
        \@addtoacct{\acct}{#1}%
    \fi
    }
```

It's a simple fork: if we are compiling a report (gathering transactions from the repository into the appropriate budget lines of a report), we'll do one thing (\@deposit), and we'll need the comma-separated list of tags that comes as #2. Otherwise, in the database document, the transaction is merely being recorded, and we'll just update the corresponding account's balance through \@addtoacc, for which the tags are unimportant and we focus only on the amount (#1).[1]

\@addtoact has an extra argument, passed on to it by \acct, as seen above. This is because the transactions are entered within one of three LaTeX environments, one for each of Alia Musica's accounts: checking (in which case \acct is defined as "chk"); money market ("mmk"); and PayPal ("ppl").

In fact, the latter illustrates another use of the flexibility of \deposit. In PayPal transactions, every deposit has a fee. Accordingly, the paypal environment in TeXcel redefines \deposit to take care of both the revenue amount and the associated fee. For example, a ticket sale could be:

```
\deposit: 15-.62 (Ticket sales, Spring 16)
```

Within the paypal environment, TeXcel then knows to add \$15 and subtract 62 cents (and, furthermore, it knows that the \$15 is 'ticket sales' and the −.62 is 'bank fees').

2.2 Convenience bundling macros

Transactions like ticket sales are (hopefully) very frequent, and there's no point in requiring the corresponding tag from the user. Ticket sales also come in groups (more than one at a time). So, TeXcel has a further macro, \tickets (a straightforward front-end for \deposit), that takes care of it:

```
\tickets: 30-1.17, 60-2.04 (Fall 15)
```

There are similar 'shorthand' macros throughout the system, including \donation and recurring expenses like \stamps, \servicecharge, and so on, all of which built on top of the basic \deposit and \expense.

One such macro is \check. TeXcel provides for an independent database of checks, where checks are defined in full detail. The transaction database can then call checks up through

```
\checks1131-1136,1138,1140,1141,1143-1150.
\checks3153,3177-3193,3195-3199.
\check3176
```

(notice the flexibility in syntax!) This saves a lot of time when entering the transactions, and in addition is extremely useful to keep track of checks that have been issued but not cashed yet.

In the spreadsheet model in use until now, uncashed checks have been a source of nearly-intractable discrepancies between projected budgets, reports, and account balances. With TeXcel this is no longer a problem: \check, \checks, etc., can themselves be redefined for any purpose (just as \deposit and \expense) — notably, to run through the checks and make a list of pending liabilities.

[1] Now looking at the definition, I can't see a reason for the immediate handling of the arguments; \deposit could simply let \@deposit and \@addtoact pick them up later. This (and the \long, by the way) must be a residue from some initial try or a different basic model. The wonders of organic growth.

2.3 Time-based reporting

A further utility worth noting: as mentioned above, reports can go by season (September–August) or by calendar year. This was a tough nut to crack, not least because sometimes a season has revenue and expenses that happen actually before the beginning of the season (a grant that's awarded in July, say), or after its end (a check that we only get in September). This is the reason for `\yearindex` above in the definition of `\deposit`. The internal mechanism will be detailed below. At user entry time, any transaction can be recorded as the argument of `\late` or `\early` — TEXcel will know what to do with it.

This means that TEXcel keeps track of the time transactions occur. In fact, the user actually enters transactions within new LATEX environments for the months — `\begin{January}` and so on. (So, each month has three nested environments, for each of the accounts.)

This brings a major benefit over the spreadsheet model: the time dimension is preserved. Back in the spreadsheet, with transactions assigned vertically by project/season and horizontally by budget line, there was no way to keep track of exact account balances by date. When discrepancies occurred at reporting time, locating them required going over all the statements manually, one by one, until something popped up. In contrast, with the month environments in TEXcel, the system is able to report account balances at the end of each month — catching a discrepancy is now trivial.

2.4 Automatic consistency checking

Beyond all the above features and benefits, probably the most important utility offered by the new system is the automatic check for duplications and omissions. When compiling a report, the user instructs TEXcel to gather transactions by tag into different budget lines. To repeat an example from above:

```
\begin{expenseline}{Marketing}
    \include{Poster distribution}
    \include{Poster printing}
    \include{Flyer printing}
    \include{Advertising}
    \include{Online advertising}
    \include{Booklet printing}
    \include{Booklet shipping}
    \include{Website}
    \include{Project website}
    \include{Postcards}
\end{expenseline}
```

Other `expenseline` environments include, for example, performer honoraria (tags like 'performer', 'performers', 'conductor', 'soloist', etc.), operation expenses ('insurance', 'office supplies', etc.), and so on.

After the series of user-requested expense (or revenue) lines, TEXcel automatically compiles a further list, "Unassigned Transactions", of those which were not included (probably due to the user's unintentional omission) in any of the environments.

On the other hand, the system keeps track of which transactions have already been assigned, and if a later budget line matches an already used tag, it will warn that the transaction number x on month y was already counted in section z.

With these two features, the system provides reliable consistency and completeness checks, a major advantage over any static tracking system.

2.5 Programming tricks

It is still the case that TEX is not exactly the most adequate programming environment for either database or spreadsheet handling. There are some structural limitations to what TEX can do — no easy alphabetization, for example, and a certain clumsiness in holding information for later use, which all but discourages trying to present the report in table form. (TEXcel makes LATEX sections for each budget line, an `itemize` environment for each tag included, followed by a total of the transaction amounts for each section.)

Even those features that are implemented required a bit of hacking. Here are some of the most interesting tricks.

Arithmetic We're dealing with dollars and cents, but TEX's arithmetic is limited to integers... At first I used the trick of dealing with dollar amounts in "dimen" registers — TEX is good at arithmetic with lengths and dimensions. It was a little funny that all amounts reported by TEXcel would have "`pt`" after them, but one could live with that.

I thought I was so clever. The problem, of course, is TEX's upper limit — a sum like 35000 is a longer dimension than TEX can handle. I tried to salvage this by working in terms of 'scaled points' (the true internal unit that TEX uses, much smaller physically and therefore with a much much higher upper limit). But TEX still presents dimensions translated into normal `pt` units — so that the numbers reported would be meaningless. (And if you simply try to reconvert `pt` into `sp` right before typesetting, you again exceed TEX's numeric limit.)

There was no other way than to implement decimals "manually". I checked a couple of existing packages, but in general they provide for much more complicated functions (trigonometry, floating point, etc.), not exactly what I needed.

So, in TEXcel all arithmetic is done through two streams of integers: one for the dollars, one for the cents. Then there is a function that converts that into the usual decimal point presentation.

In this context the decimal period has no mathematical meaning. As a result, TEX would read `12.4` as 4 cents, not 40. Very annoying, and very annoying to fix! But it had to be done: leaving the task to the user (possibly months from now) would invite potentially untraceable mistakes.

Feedback mechanism When a report is compiled (selecting out transactions by tag), what happens roughly — very roughly, almost entirely notionally — is that TEX goes through the list of transactions, and creates new commands for each tag. That way, when a tag is requested by the user, it is an active command that 'executes' the corresponding transaction … but this command also redefines itself, so that when the transaction is called for a second time, it now does something else (warn of the duplication).

Months and years I was amazed at how quickly this could get extremely confusing when I was trying to implement it. The problem is that TEX needs to know, according to what kind of report we're doing (by season or by calendar year) which months to include. You would say it's a matter of adding an offset to the month counter (so that September is 1 when going by season, but 9 when going by calendar year), and so did I. This basic offset is in fact somewhere in the code.

But that's not enough, because transactions outside of the requested year are still relevant: sometimes we have a grant for season n that was actually awarded and cashed toward the end of season $n-1$; sometimes liabilities and receivables overflow to season $n+1$. On the one hand, these outer transactions are still necessary for the full picture of a season; and on the other, they should be kept *out* of the reports for the seasons where they actually took place. (Calendar-year reports do not need this nuance, since they are basically balance-sheet reconciliations for the IRS.)

So in the end the model uses a 'year index:' 0 for the current (requested) season, -1 for the previous one, 1 for the following one. The counter is updated by the month environments: `\begin{September}` (the first month in a season) steps `\yearindex` by 1. Then the program knows what to do.

2.6 Future

The system is complete in the sense that any user (say, an intern) can use it. It is also very flexible — key functions like `\deposit`, `\expense`, and `\check` are essentially black boxes that can be redefined for any future needs that might arise.

But these future needs are not implemented. That is to say, it is only Alia Musica's needs that are implemented right now. In that sense the program is incomplete, and only a model for what could be done for more general purposes.

Were this to be done for a public release, then it would probably be a good idea to translate the code into LuaTEX — Frank Mittelbach's suggestion — so that we get a true general-purpose programming language. Desirable features would include reporting by tables, alphabetization and other sorting capabilities, and, less cosmetically, a more powerful engine to handle the tags. Right now the program compares tags, and when it finds a match it assigns a transaction right away; future matches of the same transaction are discarded (with a warning). It would be great if the user could request more complicated conditions: "include this tag but only if this other one is not there"; or "only include transactions with the indicated tags if in addition they have tag x".

For now, it was a lot of fun, not that hard, and in any case very surprising, to work on implementing Alia Musica's financial needs through TEX. The resulting system is enormously, structurally, superior to any spreadsheet.

◇ Federico Garcia-De Castro
Artistic Director,
Alia Musica Pittsburgh
federook (at) gmail dot com
http://www.garciadecastro.net

Hyphenation in TEX and elsewhere, past and future

Mojca Miklavec and Arthur Reutenauer

1 The past eight years: hyph-utf8

Hyphenation, or word division, is an essential feature of TEX and related systems, which was a pioneer in the area. Frank Liang, a student of Donald Knuth, devised an algorithm to efficiently store the information that specifies how to break words. Liang's PhD thesis on the subject was published in August 1983, and TEX82 already included the algorithm. Liang also wrote the program `patgen` that, given a list of hyphenated words, produces a set of *hyphenation patterns* that embed the information.

TEX82 would store only one hyphenation table, but with TEX 3 in 1990 it became possible to include multiple pattern sets, identified by the value of the primitive `\language`. At the same time TEX's character set was extended from 7 bits to 8 bits, thus widening the range of supported encodings. This would prove essential for many languages. Development had indeed started early to devise sets of patterns appropriate for different languages; for example Italian, for which patterns were produced and described in *TUGboat* volume 5, issue 1 in 1984; and French and German, in the next issue. However, with only 7 bits to use, most languages needed a number of tricks to work correctly, some of which could rightly be called dirty, and which were kept even after TEX3 came along.

The terms of use of the different pattern sets, when there were any, were equivalent to those of TEX itself: free to use and distribute, and modified versions should have another name. Most of the time, however, there was no clear licence. When the LATEX Project Public Licence, the LPPL, was created in 1999, some authors adopted it for their patterns, and over the years the majority of the files became available under this licence.

When XeTEX and LuaTEX were included in distributions, encoding became once again a problem since these engines expect UTF-8 input by default and couldn't accommodate the various 8-bit encodings the different pattern files were using. In order for XeTEX to be added to TEX Live in 2007, its creator Jonathan Kew devised a solution whereby patterns were converted to UTF-8 on the fly when read by XeTEX. This worked but seemed awkward, and when the following year it was LuaTEX's turn to be integrated in TEX Live, we felt this decision needed to be reconsidered.

We decided to adopt the converse strategy of what was originally done for XeTEX: convert all the files to UTF-8, and devise a system to convert the patterns back to the appropriate 8-bit encoding if necessary. That way XeTEX and LuaTEX could read the files in UTF-8 directly, while pdfTEX and Knuth's TEX would also work because they'll see 8-bit versions of the patterns, converted on the fly. It should be noted that all this happens when generating formats, as — except for LuaTEX — this is the only moment when hyphenation patterns are read by TEX, in its iniTEX incarnation. Once that job is finished and the formats are dumped, each engine will be fed the characters in the encoding appropriate to its kind.

The initial work was done in the spring of 2008 and was completed in time to be included in TEX Live 2008, as was the original intention. We also used this opportunity to rationalise the names of hyphenation patterns, most of which used relatively cryptic two- or three-letter codes to identify languages: after some research, we made the decision to use the standard BCP 47, which to our knowledge is the only one that allows the level of precision we need to distinguish between all the languages TEX supports. BCP stands for "Best Current Practice" and is used for a number of specifications by the IETF, the Internet Engineering Task Force. This standard is thus also used in most Web technologies, and the exact same language tags can be used in HTML and HTTP, for example. Since all BCP specifications are published in the RFC (Request for Comments) series, it's probably useful to mention that BCP 47 is currently equivalent to the combination of RFC 5646 and RFC 4647; these numbers may change in the future, when the document is updated.

The result of this effort was the package hyph-utf8 that is now used in MiKTEX as well. It was soon picked up by external projects: Hyphenator, a JavaScript program for supporting hyphenation in browsers; then Firefox, that implemented hyphenation in the browser itself; and finally Apache FOP (Formatting Objects Processor), an XSL-FO implementation. All these programs took patterns directly from our package, usually with just one straightforward conversion to adapt them to their format.

Since then, we've been keeping track of the updates to the hyphenation patterns in the TEX world; most of the time we're in direct contact with authors, who sent us their contribution directly, but we regularly find isolated updates for some languages. We've also welcomed pTEX in TEX Live in 2010, specialised in typesetting Japanese, for which we had to adapt the pattern loading strategy, since it didn't support UTF-8 input; this meant we had to give up on the idea of converting patterns on the fly, and thus provided 8-bit versions of all patterns that would be used for pTEX only; the UTF-8-encoded files serve as the master data. And we're constantly trying to clarify the licence terms of the patterns. We feel we have a very good momentum in the TEX community, and the `tex-hyphen` mailing list (`http://lists.tug.org/tex-hyphen`), that's been driving the effort since we started it, has become some sort of town square to discuss many language-related topic in the TEX world, far beyond the subject of hyphenation.

The rest of the free software world, however, is a completely different story. The attentive reader will have noticed that some names are missing from the above list of projects we're collaborating with, and indeed we had little interaction with the developers of the existing free word processors. Some conversations took place, to be sure, but there was no concerted effort to collect all patterns in a central place, or decide what list of licences was acceptable for the different projects.

At the time OpenOffice was the most widespread of the word processors from the free software world, and pattern sets had already been adapted for its use, starting some time before hyph-utf8 was created. The conversion was usually done for one file at a time, with no coordination between the languages, much like in the past individual pattern sets were uploaded on a one-off basis to CTAN. On occasion patterns were created for new languages, which we took over when we became aware of it. There was also a lot of talk about the licences of different pattern files, and some changes came back to us because of that. By then all major free software licences were used by pattern files: GPL, LGPL, n-clause BSD for different values of n, MIT, LPPL of course, and some free-form text. The expansion of these acronyms is left as an exercise to the reader (but read on for a partial cheatsheet). In addition, some people were apparently asked to sign a contributor licence agreement (CLA), that is, an express agreement between the authors and the organisation responsible for OpenOffice. We are still unclear as to why it was so, but to our knowledge this hasn't been the case since the Apache Software Foundation took over the maintenance of OpenOffice in 2011.

Not much happened on this front for a few years, until it was time for Google to join the party. In September 2015, and then again in December of that year, many pattern authors were contacted with requests to once again change the licence of the files, with little explanation of why they were asked to do so. It took us some research to understand what was happening: hyphenation had been added to the operating system Android, and Google was thus interested in using the hyphenation patterns available for TEX and other free software, but they had some restrictions relating to licences. As it turned out, the LPPL was the one that caused most problems for them. There was also a secondary suggestion to add the hyphenation patterns to Unicode's Common Locale Database Repository (CLDR), a large project collecting linguistic data for many languages; there were also legal obstacles to that at the time.

There followed several weeks of extensive discussion, hardly interrupted by Christmas and New Year's Eve, between pattern authors, ourselves, and representatives of Google, soon joined by developers from Mozilla (for Firefox), LibreOffice — by now the most active free software office suite — and even Amazon (for software running on Kindle), over the course of which many, many emails were exchanged and went several times round the planet. Conversations that had started among maintainers of patterns for some language inflated to include more and more contributors, spilled over unto mailing lists, took a side road to discuss the respective merits of different licences, turned around to question the motivation of the requesters, deflated back again to wonder what the exact wording of a licence statement should be, and finally died down when absolutely, absolutely everything had been said and pondered, even that which should probably never, ever have been said nor pondered. When the dust finally settled in the cyberspace and the protagonists were recovering from what can only be described, in the words of one of the authors of this article, as "a huge wave of ???", we came to a decision, and we now have a plan, that will be developed in the next section. It also transpired a few months later that in order to include the patterns in the CLDR, it would be necessary for each contributor to sign a CLA, which will probably be very close to the individual CLA for contributors to Android; however, the exact text is not finalised yet.

2 The past few months

We need to say a few words about the LPPL, since it did as mentioned cause the most amount of talk. It is a licence characterised by two main conditions: first, that any work derived from a work under the LPPL identify itself as such, and second, that each work come with one particular person known as the maintainer, responsible for keeping it up-to-date. Both these conditions represent specific challenges, which we'll attempt to explain.

The first one, specified in clause 6.1 of the current version of the LPPL (1.3c, dating from 2008), is well-known to TEX users as it is equivalent to the condition under which Donald Knuth made TEX available: that any modified instance of the program that didn't fulfil a strict series of tests be given a different name. Earlier versions of the LPPL actually used a wording much closer to Knuth's (it changed with LPPL version 1.3 in 2003). This clause, however, is virtually unheard of outside the TEX community, and whenever external projects want to use hyphenation patterns that are placed under the LPPL, we need to do some education about the terms of the licence (who does that does not matter as long as a conversation is had, but in practice it often boils down to the two authors of this article). This may turn out harder than it looks, as we've experienced some resistance to that notion, that sometimes gets questioned or even ignored; we did for example have a discussion with a lawyer from a technological company who found the wording of the LPPL ambiguous and stated that it "imposes a lot of confused definitions of derivative works". This project rejected the LPPL based on its text alone. Even without such a strong reaction, the LPPL is generally frowned upon and pattern authors are thus often asked to make their files available under a different licence (to "relicense" them), the exact one depending on the project.

The reasons for third-party developers to request another licence are not only psychological: the identification condition of the LPPL, while seemingly simple to comply with, actually makes it incompatible with many licences. Roughly speaking, that's all the copyleft licences — those characterised by the requirement that any modification of the work they apply to be made available under the same conditions — such as the GNU General Public Licence (GPL) and Lesser General Public Licence (LGPL), or the Mozilla Public Licence (MPL). In these cases the patterns have to be relicensed under the copyleft licence, or a licence compatible with it, in order for the external project to be allowed to use them; it is not a matter of taste. This is one main reason for the multiplicity of licences, the other chief one being authors' personal preferences, although often they don't have strong opinions. Attempts to work around that incompatibility are known to not work; in at least one case we are aware of a project trying to incorporate patterns under the LPPL into copylefted code (by documenting the situation and giving proper credit to the original authors, of course), but we now understand this to be a violation of the LPPL.

For external projects, this is not such a problem in practice, however, as authors generally develop their patterns in the spirit of collaboration and open access, and thus readily agree to make their patterns available under any alternative licence when asked. The one issue is logistical: it can sometimes be hard to contact some authors, and there are many external projects trying to have the patterns relicensed, which leads to the only case of reluctance we've experienced: on occasion an author will display a clear (and understandable) expression of frustration at being asked the same question over and over again — a situation named "relicensing fatigue" by a developer used to being on the other end of these conversations. It should be noted that in this situation the authors of this article have thus far been only passive, witnessing discussions between pattern authors and third-party projects.

The other chief condition of the LPPL is that each work placed under it should have a designated maintainer, the one person allowed to make changes without needing to change the identification of the work as per clause 6.1. By default it is the original author of the work, who may nominate another person when they are no longer willing or able to look after the work themselves.

The issue we're having with that condition is that we don't understand who is the maintainer of the pattern files in hyph-utf8: in our case, we are not the original authors of the patterns and we have not been appointed by them as maintainers (except in a few minor cases), but we are definitely the point people responsible for changes in hyph-utf8, whose core is a set of pattern files with new names. Who are thus the maintainers of the individual files? If it is the pattern authors, this would be a statement of fact that isn't true: the original authors are not the people allowed to make arbitrary changes to the file in hyph-utf8; we are. If we are the maintainers, this would — formally — deny the authors any say in their patterns (as they are packaged in major distributions). This is a serious problem in the application of the text of the LPPL.

In practice this doesn't actually make any difference: the pattern authors communicate with us in a number of ways, and since our job is only to package the files in a format amenable to TEX distributions, we usually adopt any change to the actual patterns straightaway; and on the other side, we collaborate with the core developers of said distributions to keep our package up-to-date with the latest requirements, such as for example when we had to devise a new encoding strategy for including the patterns in pTEX formats, or when a bug in XƎTEX was revealed by the patterns at format-generation time in the development of TEX Live 2016. This is in our opinion the way it should be done: we volunteer our time to work on the low-level support of the patterns, and the authors volunteer theirs for the linguistic aspect; there doesn't need to be a formal recognition of these roles.

The notion of a maintainer thus brings no practical benefit while introducing theoretical problems that come to light during the post-apocalyptic discussions mentioned above. Two examples will serve to illustrate this fact: in one case, the author of a set of patterns under the LPPL had been asked to relicense their patterns. They seemed relatively willing to do that provided the relicensed files would only be used outside the TEX community, but showed some attachment to the idea that the original files should stay under the LPPL. They then finally stated "I will change the licence if and only if the TEX-hyphen working group allows me to" (meaning us). This is heartbreaking to us: we are not in the business of prescribing to volunteers how they should make their work available to the world; this should be entirely their choice. We are able to make recommendations, of course, but we have refrained as much as possible from interfering in other people's decisions. Since, however, our input was clearly called for at that point, we of course gave our "authorisation", without feeling entitled to do so.

The other situation was even stranger: in that case the original author had appointed a maintainer as they no longer felt in a position to look after the patterns. The maintainer was keeping a strict policy of no modification to the patterns and thus only added a few lines of comment to the file when they took over maintenance. This was before we started hyph-utf8, and when we did, we simply adopted the file with minor modifications. However, during the discussion about relicensing a few months ago, the maintainer, who had not contributed to the actual patterns at all, felt empowered to decide the terms of the new licence on behalf of the original author (who admittedly seemed a little confused by the situation), and then proceeded to dictate the exact comments we were to put in the file, and even the name of the file itself. This is of course not acceptable to us and we did our best to ignore the maintainer's whims.

These examples are cause for concern, because they show a certain amount of misunderstanding around the core conditions of the LPPL, on many authors' part (the two cases above are by far not the only ones); they also are a clear waste of everybody's time, caused solely by use of the LPPL.

It is thus clear to us that the LPPL is a poor choice for a project such as ours and we are going to change our policy and start recommending against it. We recognise its goals and want to achieve them too, which can't be through formal definitions of different roles such as maintainer. In our case, this has no practical effect and generates too much confusion, as can be seen from the examples above. The correct way to proceed, in our opinion, is by fostering healthy discussions among all the parties involved, from package writers, to distributors, to end users. The situation for hyph-utf8 is of course special since the authors of this article are in some way an intermediary between package writers and distribution developers, but that only strengthens the need for close collaboration.

It has been suggested that since we're effectively responsible for all the patterns in TEX distributions it was also our role to defend their status under the LPPL and enforce it in other projects, if necessary. We don't agree. We're looking after hyphenation patterns and the licence shouldn't matter. What's more, the LPPL doesn't come with any procedure to enforce it, or any compliance team that could help track down violations and rectify them — contrary to other licences such as the GPL and the LGPL, where one can assign copyright to the Free Software Foundation (FSF), that also provides guidelines on how to investigate and remove suspected violations. In any case, that's a discussion we'd (much) rather not be part of, even if we're sometimes co-opted into it.

For that matter, we don't even have guidelines on what the desired level of stability for the hyphenation patterns is. There is no agreement on what type of modifications are acceptable, and depending on the language the authors may develop them actively, or not at all. We follow the changes while trying to be conservative for long-established pattern files (this is relatively easy to do since files of a certain age generally get few updates); but there is nothing to prevent someone from one day sending mass changes to decades-old patterns for a major language, and a difficult conversation will need to take place at that point: will we ignore the changes, or create a new file for the same language, or . . . ? We are willing to follow any reasonable policy that is agreed upon, but we can't make it ourselves: that clearly has to be done by the wider community.

In short, we want to ensure the best possible future for the patterns and this goes through collaboration with all the projects that are interested in developing them. If we kept strictly to the LPPL, this would create an artificial divide among the languages, separating those that can be shared with other projects and those that can't; inevitably the latter would tend to get less attention and become second-class citizens. We had already noticed that among the languages whose patterns had moved away from the LPPL were some of the world's major languages and we wanted to look into this fact in more detail. We thus ordered them by decreasing number of speakers; having grouped together all language variants together, we had exactly 60 languages. From this we took the top third, to see how many were still under the LPPL, and the result was striking: only 2 out of 20. And even then, the status of these two was quite artificial, since one had seen its licence being chosen arbitrarily as the "popular" choice, and the other one actually comes with several variant files, some of which are not under the LPPL; the one that ships in hyph-utf8 is, but external projects could just as well take the ones whose licences are more acceptable to them. One can of course argue about the methodology and the difficulty of determining the number of speakers for each language, but this figure alone clearly points to the fact that the divide is already there. It is very likely this had been brought about by the fact that major languages naturally got more attention and that pattern authors have thus been asked to relicense more often. The "less important" languages are thus at risk of seeing their patterns progressively lose ground.

In conclusion, we think that in spite of all the difficulties, we are at a point where we're finally able to put TEX at the centre of all efforts to create good hyphenation patterns in the free software world. Let's use that opportunity and not close ourselves to change; we want to become proactive in that effort and we now have a plan: we are going to start recommending to pattern authors to switch to the MIT licence, a very permissive one that has a simple text; other permissive licences would of course be acceptable, if that's the author's preference. We also want to start talking about hyphenation proper instead of politico-legal issues. We accept that the latter is inevitable but we've had far too much of it lately, as the disproportionate structure of this article makes clear.

Back to the work in progress: we have made a few updates for TEX Live 2016. Apart from some licence business (as usual ...), it has a few low-level changes: we've broken up the main package into different language groups to reflect the many packages called hyphen-⟨*language*⟩; the patterns used to all be in the former package while the latter were shells merely containing instructions on how to populate `language.dat`. We have also rewritten the top of each pattern file to make the comments machine-readable, in an effort to make the many language files easier to process; and we've renamed the plain text versions of the patterns to more human-friendly names. Finally, we have migrated the repository to GitHub for better visibility. These changes, however modest, should be seen as preparatory work for making our package more palatable to external projects.

In a separate effort, we also started looking into `patgen` and, in an endeavour to understand in detail how it worked — and to support UTF-8 input — we rewrote it in Ruby using object-oriented programming. As a result, it is hopelessly slow, by a factor of about 60, but we have been able to reproduce the results exactly, and we are going to look into making it more efficient, and hopefully enhance it. The code is currently available at `https://github.com/hyphenation/hydra` and will be distributed through CTAN and TEX Live in due time.

3 The next few months

The future plans right now include:

- Setting up a new website for the project at `http://www.hyphenation.org`.
- Getting as many pattern authors as possible to agree to the MIT licence and Unicode's CLA.
- Unify all the patterns from different sources and finally become the central hub for all things hyphenation in the free software world.

4 The future

Some more distant plans involve looking at the hyphenation algorithm in more detail: the hyphenation library used by the free word processors, `libhyphen`, actually does a little more than TEX's original algorithm, which is not surprising since the current version dates from the mid-2000s, and it would certainly be nice to see if the additional features couldn't be included into TEX engines too. There's always more to do!

◇ Mojca Miklavec
Sežana, Slovenia

◇ Arthur Reutenauer
late of the Royal Opera House,
Covent Garden
`arthur.reutenauer (at) normalesup dot org`

Zebrackets: A score of years and delimiters

Michael Cohen, Blanca Mancilla and John Plaice

1 Introduction

In this paper, we present the resurrection of the *Zebrackets* project, originally initiated by the first author 20 years ago, with which parentheses and brackets are *zebra-striped* with context information. There are two reasons for this innovation: first, to improve visual presentation of the necessary linearization of hierarchical structures in text, and second, to make a first step away from the assumption that documents must be built up from a set of unchanging atoms called characters.

Parentheses and other pairwise delimiters are important because they are the primary way by which text, which is serialized, can denote higher-order dimensionality. For example, two-dimensional structures can be directly expressed, as in $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$. For 2D data structures such as matrices, such graphical expression is natural, but unnecessary, as serial expression is logically equivalent, albeit less perspicuous, as in $((a_{11}\ a_{12})\ (a_{21}\ a_{22}))$, and generalizable to arbitrary rank and dimension, as in $(a_1\ (a_{21}\ a_{22})\ (a_{31}\ (a_{321}\ a_{322})))$. Such notation usually assumes "row-major" order, in which the horizontal index changes fastest in a canonical (depth-first) enumeration. This convention can be made explicit, by introducing grouping delimiters, as in $\begin{pmatrix} (a_{11} & a_{12}) \\ (a_{21} & a_{22}) \end{pmatrix}$. The transpose ("column-major") representation, is given by $\left(\begin{pmatrix} a_{11} \\ a_{12} \end{pmatrix} \ \begin{pmatrix} a_{21} \\ a_{22} \end{pmatrix} \right)$.

Zebrackets[1] were originally developed more than a score (20) of years ago, demonstrating the expressive power of microarticulated glyphs [Coh92] [Coh93] [Coh94]. The basic idea is to allow parentheses [Len91] and square brackets to take on stripes and slits ("poles 'n' holes") to carry extended information, such as functional rôle, logical position, and nesting level in an expression. Pairwise delimiters are "scored", by cutting aligned typographical grooves, to associate balanced mates and ease visual parsing.

Below, we show one possible scoring of this sample text:

[(a [b (c) d] [e (f) g]) (a [b (c) d] [e (f) g])]

Here it is, with greatly magnified brackets and parentheses:

[1] The name, suggested by Bob Alverson, is a play on words as the delimiters resemble zebra stripes: (ze(bra)kets).

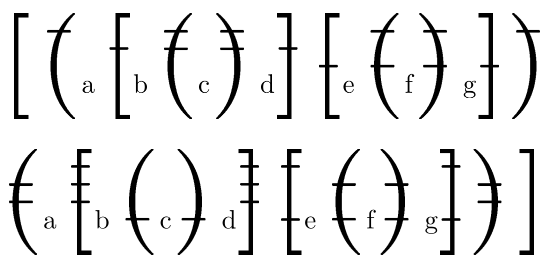


Table 1 shows a number of examples of the use of the Zebrackets infrastructure on the same sample text.

The intervening score of years has not been especially kind to the original implementation: it was hardly sturdier than a "paper-clips and bubble-gum" contraption in the first place, and the slide into deprecation and disuse of METAFONT, accelerated by the emergence of PDF as the interchange format of choice, which cannot natively use characters generated by METAFONT, hastened the obsolescence of the Zebrackets prototype. Adobe's "Multiple Masters" (such as Adobe Sans and Adobe Serif) and Apple's TrueType GX were similarly ahead of their time, and failed to achieve critical mass and widespread adoption. Jacques André's contextual fonts, dynamic fonts [AO89] [AB89], and Scrabble font [And90] were Type 3, so also withered.

Nevertheless, we believe that the principles underlying the system are still valid. There is a huge multidimensional space of potential characters and glyphs, too big to be precompiled, and so a lazy, demand-driven, image-time generation, both of fonts and glyphs, with caching or memoization, as is used in dynamic programming, is the only practicable solution. Contemporary assumptions about fonts do not allow this possibility [Har07], so reviving the existing implementation strategy is still of relevance. The presentation here presents the font structure, and the use, both implicit and explicit, of these fonts.

2 The fonts

The Zebrackets project relies on a set of fonts generated from the METAFONT version of the Computer Modern fonts. The names of the fonts are all of the form `z`$(a)(b)(c)(d)(e)$, where:

(a) is a single letter, either '`b`' or '`p`'; the font contains either all brackets (`b`) or all parentheses (`p`).

(b) is a single letter, one of '`b`', '`f`', or '`h`'; the marks in the font are all either slots (`b` for background), ticks (`f` for foreground), or ticks within slots (`h` for hybrid).

(c) is a single letter, one of '`a`' through '`h`'; the font will contain 2^m pairs of left and right delimiters,

Table 1: Stripes, slits, and slots: Examples of zebrackets with various arguments. Each zebracket has a set of slots (here computed automatically), which can be striped according to the chosen style: plain "foreground" stripes (style 'f' in the table); more subtle, erasing "background" slits (style 'b'); or "hybrid" (style 'h'), which creates a slit for each slot, then places foreground stripes on top thereof. Stripes generation can automatically count unique pairs or track nesting depth, or count unique pairs at a given depth ("breadth"). The encoding can be unary, binary, or "demultiplexing", up through the maximum as calculated by initial pass of a parser. Note that all encodings have 0 as origin, but the rendered index origin can be changed to unity.

`encoding`	`style`	`index` = unique	`index` = depth	`index` = breadth
unary	b	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
	f	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
	h	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
binary	b	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
	f	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
	h	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
demux	b	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
	f	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))
	h	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))	(a (b (c) d) (e (f) g))

Table 2: Font `zphecmr12`.

	`0/8`	`1/9`	`2/A`	`3/B`	`4/C`	`5/D`	`6/E`	`7/F`
`0`	(	(	(	(	(	(	(	(
	(	(	(	(	(	(	(	(
`1`	)	)	)	)	)	)	)	)
	)	)	)	)	)	)	)	)

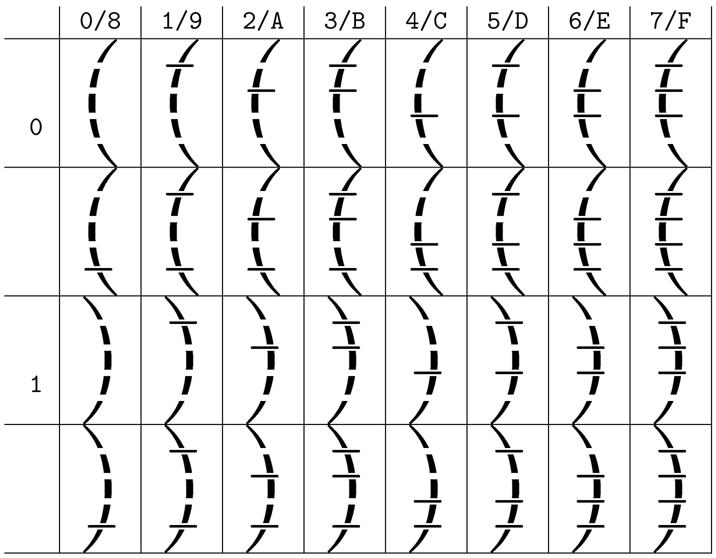
Table 3: Font `zbfdcmtt12`.

	`0/8`	`1/9`	`2/A`	`3/B`	`4/C`	`5/D`	`6/E`	`7/F`
`0`	[	[	[	[	[	[	[	[
	]	]	]	]	]	]	]	]

Table 4: Font `zphecmr12, magnification` 2.

	`0/8`	`1/9`	`2/A`	`3/B`	`4/C`	`5/D`	`6/E`	`7/F`
`0`	(	(	(	(	(	(	(	(
	(	(	(	(	(	(	(	(
`1`	)	)	)	)	)	)	)	)
	)	)	)	)	)	)	)	)

Table 5: Font `zphecmr12, magnification` 4.


0/8
1/9
2/A
3/B
4/C
5/D
6/E
7/F
0
1

with $m \in 0..7$, where `a` corresponds to $m = 0$ (i.e., 1 pair or 2 glyphs), `b` corresponds to $m = 1$ (i.e., 2 pairs or 4 glyphs), ..., and `h` corresponds to $m = 7$ (i.e., 128 pairs or 256 glyphs).

(d) is the name of a Computer Modern font family, such as '`cmr`'.

(e) is a font size, such as '`10`'.

We consider two examples. In the first example, font `zphecmr12` (Table 2) was generated by calling the `zebraFont.py` script with arguments specifying parentheses striped in a hybrid visual style across 4 slots, using 12 pt Computer Modern Roman as the base:

```
python3 zebrackets/zebraFont.py
--kind parenthesis --style hybrid
--slots 4 --size 12 --family cmr
```

The font contains exactly $2^{4+1} = 32 =$ `0x20` parentheses. In the first half of the table, for $i \in$ `0x00` .. `0x0F`, glyph i is an opening (left) parenthesis, encoding i as a binary number with ticks placed in four always-drawn slots. Similarly, in the second half of the table, for $i \in$ `0x10` .. `0x1F`, glyph i is a closing (right) parenthesis encoding ($i -$ `0x10`) as a binary number.

In the second example, font `zbfdcmtt12` (Table 3) was generated by:

```
python3 zebrackets/zebraFont.py
--kind bracket --style foreground
--slots 3 --size 12 --family cmtt
```

specifying square brackets striped, using a foreground style, across 3 slots, with the 12 pt typewriter font as a base. The font contains exactly $2^{3+1} = 16 =$ `0x10` brackets. For $i \in$ `0x00` .. `0x07` across the top of the table, glyph i is an opening bracket encoding i as a binary number with ticks placed in three fixed slots. Similarly, in the bottom half of the table, for $i \in$ `0x08` .. `0x0F`, glyph i is a closing bracket encoding ($i -$ `0x08`) as a binary number.

The `zebraFont.py` script generates a new METAFONT file whose name is the font name followed by `.mf`, placing that file in the directory

```
$TEXMFHOME/fonts/source/public/zbtex
```

then calling `mktextfm` on that font name, before finally calling `mktexlsr` to add the generated `.mf`, `.tfm`, and `.600pk` files to the cache for `$TEXMFHOME`.

Thus, for the second example above, the script `zebraFont.py` generates the METAFONT file:

```
.../source/public/zbtex/zbfdcmtt12.mf
```

and `mktextfm` generates files (assuming `ljfour` is the default output mode):

```
.../tfm/public/zbtex/zbfdcmtt12.tfm
.../pk/ljfour/public/zbtex/zbfdcmtt12.600pk
```

and `mktexlsr` updates the cache of the list of files:

```
$TEXMFHOME/ls-R
```

We can also make magnified versions of these fonts. Our third example (Table 4) is the `zphecmr12` font with magnification 2, which corresponds to TEX magnification $\sqrt{2} \approx 1.414$. The font was generated by:

```
python3 zebrackets/zebraFont.py
--kind parenthesis --style foreground
--slots 3 --size 12 --family cmr
--magnification 2
```

Our last example (Table 5) shows the same font with magnification 4, corresponding to TeX magnification 2. The font was generated by:

```
python3 zebrackets/zebraFont.py
--kind parenthesis --style foreground
--slots 3 --size 12 --family cmr
--magnification 4
```

When a magnification argument is passed to `zebraFont.py`, the `mf-nowin` and `gftopk` scripts are called to produce larger versions of the fonts. The magnification argument is multiplied by 600. Hence we get:

```
2   .../zbfdcmtt12.1200pk
4   .../zbfdcmtt12.2400pk
8   .../zbfdcmtt12.4800pk
```

The METAFONT file that is created by the script `zebraFont.py` is an eight-line file. It inputs the base Computer Modern font, sets the parameters for the number of slots and whether the marks are foreground, background, or hybrid, then inputs a file for generating a set of parentheses or a set of brackets. For example, font `zpfbcmr10.mf` contains the following lines:

```
if unknown cmbase: input cmbase fi
mode_setup;
def generate suffix t = enddef;
input cmr10; font_setup;
let iff = always_iff;
stripes:=1;
foreground:=1;
input zeromanp;
```

The last file input, by the last line above, is `zeromanp.mf`, which is derived from the original Computer Modern `roman.mf` (prefixing "ze" to indicate its adaptation to zebrackets). It is one of four METAFONT files distributed with the *Zebrackets* project. The `zeromanp.mf` file sets some parameters, then inputs the punctuation file `zepunctp.mf`, itself derived from the Computer Modern `punct.mf`. For brackets, there are corresponding files `zeromanb.mf` and `zepunctb.mf`.

3 Using the fonts explicitly

There are two ways to use the fonts generated by the *Zebrackets* project: explicitly and implicitly. In this section, we present the explicit approach, and show how it is used to produce the bibliographic references of this article.

Suppose that a font `zbfhcmr10` has been generated and we wish to use it. Then we need to declare the font and its size with a line like this — the `J` encodes size 10 and the `A` encodes magnification 1:

```
\ifundefined{zbfhcmrJA}
\newfont{\zbfhcmrJA}
{zbfhcmr10 scaled 1000}\fi
```

Font `zbfhcmr10` is a font of 256 brackets, all with foreground ticks. To ⁅ bracket some text ⁆ with a pair of delimiters with four selected ticks (in this example, using all but two slots at the top and one at the bottom), the `.tex` source code can use

```
{\zbfhcmrJA\symbol{60}} bracket some
       text {\zbfhcmrJA\symbol{188}}
```

since binary $0111100 = 2^5 + 2^4 + 2^3 + 2^2 = 60$ and $128 + 60 = 188$.

To hint at the expressive flexibility of such functionality, the bibliographic references of the original article on Zebrackets [Coh93] placed a tick for each page in which a `\cite`-ation appeared in the left bracket of the corresponding citation label and a tick for each page in which a `\nocite`-ation appeared in the right bracket.

In this article, we show the same extension throughout (not just in the bibliographic References section), with the now explicit convention that should a document have more than seven pages, then all references beyond the seventh page activate the seventh tick. In the reference [Coh93], the activation of the first and fifth ticks in the opening bracket indicate that reference's citation on the 1^{st} page of this paper as well as here on the 4^{th} page.

Below is some of the code needed for this functionality. There are two counters used as temporary variables:

```
\newcounter{bracei}
\newcounter{bracej}
```

For each citation x, a pair of counters is set up, `ze:`x for the left bracket, and `zeno:`x for the right bracket. The `\zecite` macro is like the standard LaTeX `\cite` macro, but it also calls `\zecitation`, which bitwise-ors-in 2^{p-1} to counter `ze:`x for the left bracket, should there be a `\zecite{`x`}` on page number p:

```
\newcommand{\zecite}[2][]%
{\def\tmp{#1}\ifx\tmp\@empty\cite{#2}%
             \else\cite[#1]{#2}\fi
 \zecitation{#2}}

\newcommand{\zecitation}[1]%
{\ifundefined{c@ze:#1}%
 \newcounter{ze:#1}%
 \setcounter{ze:#1}{0}%
 \newcounter{zeno:#1}%
 \setcounter{zeno:#1}{128}\fi
 ...
 \addtocounter{ze:#1}{...}}
```

There are macros corresponding to `\nocite`, namely `\zenocite` and `\zenocitation`, bitwise-oring-in 2^{p-1} to counter `zeno:`x for the right bracket, should there be a `\zenocite{`x`}` on page number p.

Finally, the generation of the `\bibitem` is extended, so that

```
{\zbfhcmrJA\symbol{\arabic{ze:x}}}
```

appears as the citation's left bracket, and

```
{\zbfhcmrJA\symbol{\arabic{zeno:x}}}
```

appears as its right bracket.

4 Using the fonts implicitly

Although providing an explicit interface to the Zebrackets infrastructure provides great flexibility, most of the time such invocation is "under the hood" and used implicitly, through the use of pseudo-LaTeX commands appearing in a LaTeX document.

A Zebrackets-enabled LaTeX file (with conventional extension `.zbtex`) is passed through a preprocessor, `zebraParser.py`, which recognizes four constructs:

1. `\zebracketsfont` declares the need for a font, provoking its creation should it not exist.
2. `\zebracketsdefaults` sets default values for the parameters of the other two constructs.
3. `\zebracketstext` designates text in which the parentheses and brackets are to be replaced automatically with zebrackets (including "zeparentheses").
4. `\begin{zebrackets}` ⋯ `\end{zebrackets}` designates a block of text for the same treatment as for `\zebracketstext`.

Because of this precompilation, from `.zbtex` to `.tex`, the workflow for such zebracketed word-smithing is not as convenient as with, for instance, TeXShop:[2] Compilation can be managed in Unix-like shells with a `Makefile` to check dependencies and invoke the required processes, but there is no automatic preview, synchronization, or other accustomed conveniences.

4.1 The `\zebracketsfont` instruction

The previous section explained how Zebrackets fonts are generated by the `zebraFont.py` script. This script cannot be called directly from a LaTeX document, but can be invoked indirectly through the `\zebracketsfont` instruction. Consider, for example, the following call to `zebraFont.py`:

```
python3 zebrackets/zebraFont.py
--kind parenthesis --style foreground
--slots 7 --size 10 --family cmr
--magnification 1
```

[2] `http://pages.uoregon.edu/koch/texshop/`

The invocation of that call can be made implicitly in the LaTeX document with the following line.

```
\zebracketsfont[
  kind=parenthesis,style=foreground,
  slots=7,size=10,family=cmr,
  magnification=1]
```

As a prelude to LaTeX compilation, the preprocessing script `zebraParser.py` reads and parses this line, directly calls `zebraFont.py` with the appropriate parameters, and removes the line from the LaTeX document, which is exported with the usual `.tex` file extension.

One need not include the full set of key–value arguments, as default values can be used (as explained below). Further, each of the parameter names can be abbreviated, down to just the first three letters, and the keyword arguments can also be abbreviated, as in:

```
\zebracketsfont[kin=p,sty=f,slo=7,
                siz=10,fam=cmr,mag=1]
```

The `\zebracketsfont` instruction takes six arguments, which can appear in any order:

1. `kind` can be either `parenthesis` (`p`) or `bracket` (`b`).
2. `style` can be any one of `foreground` (`f`), `background` (`b`), or `hybrid` (`h`).
3. `slots` is a natural number between `0` and `7`, inclusive.
4. `size` is a natural number for a font size, such as `10` or `12`.
5. `family` is a Computer Modern font family name, such as `cmr` or `cmtt`.
6. `magnification` is a natural number between `1` and `32`, inclusive, representing the square of the TeX font magnification, i.e., a power of $\sqrt{2}$.

4.2 The `\zebracketsdefault` instruction

If Zebrackets is used extensively within a document, then a lot of calls thereto are made, perhaps with similar or even identical parameters. In order to reduce typing (and introduction of errors), default values for any of the Zebrackets parameter names can be assigned.

For example, in the following lines, four fonts are declared, all of family `cmr`, size `10`. All but one use parentheses, all but one are foreground style, and all but one have seven slots.

```
\zebracketsdefaults
  [size=10,family=cmr,
   slots=7,kind=parenthesis,
   style=foreground]
```

```
\zebracketsfont[]
\zebracketsfont[kind=bracket]
\zebracketsfont[style=background]
\zebracketsfont[slots=1]
```

4.3 The zebrackets environment

When `zebraParser.py` is called, whenever it parses text to be transformed (when the document contains either the `\zebracketstext` command or the `zebrackets` pseudo-LaTeX environment), then the `zebraFilter.py` script is called. The latter reads the text, determines what fonts are needed (invoking `zebraFont.py`, as necessary), then replaces the brackets and parentheses in the text with font–symbol pair invocations.

Consider the following example, presented in §3 with explicit font–symbol pairs:

[bracket some text]

That example can also be generated implicitly, with the lines:

```
\begin{zebrackets}
  [style=f,number=60,
   slots=7,encoding=binary]
[ bracket
some text ]
\end{zebrackets}
```

The `number=60,slots=7` specifications in the parameter list summons a font using seven slots, from which glyph `60` ($= 2^5 + 2^4 + 2^3 + 2^2$) and its partner `188` ($= 2^7 + 60$) are drawn.

For automatic processing, inputs are handled as follows:

- Parameter `index` can take one of three possible values — `unique`, `depth`, `breadth` — as exemplified in Table 1.
- Parameter `number` overrides the settings for parameter `index`. When `number` is set, all parentheses and brackets in the text being processed get that specific glyph in the font.
- When a value for parameter `slots` is not provided, then the number of slots for the fonts is the minimum needed in order to encode all of the glyphs for the text (taking into account the value of the index parameter).

For example,

(a (b (c) d) (e (f) g))

was generated by the lines:

```
\begin{zebrackets}
  [index=depth,enc=unary,style=f]
(a (b (c) d) (e (f) g))
\end{zebrackets}
```

There are also three additional parameter pairs, each with two values:

- `mixcount=true` states there should be a single counter for striping parentheses and square brackets; `mixcount=false`, two distinct counters.
- `origin=0` states that counting starts from zero; `origin=1`, from one.
- `direction=topdown` means that striping starts from the top of delimiters, whereas `direction=bottomup` starts from the bottom.

The automatic striping of delimiters in a region of text is done with a two-pass algorithm: a) the maximal depth and breadth, and the number of distinct delimiter pairs are computed, in order to determine the number of distinct slots needed (maximum of 7), and b) the correct fonts are generated, if need be, and the correct LaTeX source is created.

5 Conclusion

The Zebrackets infrastructure does not assume that characters are changeless atoms, as standard computing infrastructures do. We consider below this innovation from several perspectives.

5.1 Representative characters

The idea of characters or words as pictures is of course not new. Most characters — including Chinese characters, Japanese kana, and the Roman alphabet — have origins in pictographic associations, albeit with prehistoric abbreviations and stylizations that make the original inspiration obscure or all but indiscernible. Illuminated manuscripts often embellished initials with vines, flowers, animals, and other inventions. Almost a century ago, Apollinaire published books [dK25] featuring "calligrams", instances of "concrete poetry" or "visual poetry", in which the typeface and arrangement of words on a page informs the meaning of a poem as much as the words themselves. Contemporary typography often plays with pictorial suggestions [KH08], especially for special-purpose or display faces.

5.2 Context sensitivity

TeX has always featured non-locality, including "butterfly-effect" propagation, in which, for instance, a seemingly small change at the end of a document can affect layout at the beginning, especially in the presence of floating figures and tables. However, such effects are large-scale, macroscopic, rearranging the glyphs, but not mutating the glyphs themselves. Zebrackets suggests subatomic alteration, analog isotopes of the heretofore inviolate characters. A character is the smallest visual part of a notational system that has semantic value. A glyph is one possible representation of a character. Ligatures can be thought of as locally context-sensitive glyph adaptation, as can some kinds of accents, kerning, and hyphenation. But Zebrackets represents a larger context sensitivity, adapting symbols to the broader circumstances. In an extreme case, its filters could be applied to an entire document.

5.3 Analog articulation

Fonts can be thought of as embeddable in a manifold[3] [CK14], and perturbations on this manifold are equivalent to variations of the font characteristics. Microtypography [Kar15] is an unexploited aspect of font design and electronic publishing. Zebrackets challenges the assumption that a glyph is the smallest representation of a character that has semantic value. Such capability hints at giving glyphs depth, not in the sense of a 3D, sculptural sense [Ann74] [FVJ11] [HF13], but logical depth, in the sense of alternate projections of a set of variations on a character. Current technology discourages such generality, and, since the character/glyph/font system is so deeply and tightly interwoven with any operating system, application, or program, traditional computer typography and character-handling have a lot of inertia.[4] Even the idioms for selection in contemporary viewers have a resolution (understandably enough) of the character level. It is impossible, for instance, to select just an accent (without also getting the letter to which it is attached). Even generating kerning tables for systems like zebrackets is somewhat daunting, suggesting the need for algorithmic kerning.

5.4 Charactles

Authors Mancilla and Plaice [MP12] proposed the *charactle* — a portmanteau word combining character and tuple — as a generalization of characters and glyphs. A charactle consists of an index into a dictionary, along with some variant or versioning information; it incorporates the Unicode character as a special case. According to this model, a text would be a sequence of charactles. The zebrackets presented in this paper are completely consistent with this approach.

[3] `http://vecg.cs.ucl.ac.uk/Projects/projects_fonts/projects_fonts.html`

[4] Of course, for specialized purposes, such as display fonts and "Word Art" (as in Microsoft Word or PowerPoint), characters are unique. These are sort of "one-off"s, with no attempt to optimize their rendering by caching them into OS tables: singletons meant to be seen as much as read, leaning towards the pictorial and away from the purely textual.

5.5 The future of literacy

The "take home message" is not only the extensibility of parentheses and brackets, but the ability to articulate any character, like a metaMETAFONT. Every glyph, stroke, mark, and pixel should be deliberately and explicitly determined for the exact circumstances of its apprehension. A character set should not be precompiled as an operating system resource, a cache of common letter forms. Such a model patronizes characters by treating them as cliches, overused forms of expression. Digital typography, electronic publishing, and computer displays allow generalization of such forms by considering characters as semi-custom instances of a richly expressive class, with factory (instantiation) specifications including not only such qualities as font family, size, and magnification, but also optical balance, reader characteristics and preferences, and arbitrary relations with any other document qualities, a kind of negotiation between aspects. Factors related to reading in the context of ubiquitous computing ("ubicomp") and IoT ("internet of things") — such as ambient illumination, whether a reader is wearing glasses or not, and time of day — should be referenced as parameters to optimize legibility and experience [Coh14]. Such exponential explosion of expression space, a hoisting of a quantized model into a seemingly continuous one, can still run on a digital computer but requires virtually arbitrary smoothness, promoting, as it were, integers into reals, necessitating on-the-fly compilation. Such display is optimally realtime, but need not be, since a document browser could initially display unadorned versions of characters, perhaps preconditioned to reflect anticipated layout, dynamically and progressively refreshing by swinging in the embellished versions as they are generated.

References

[AB89] Jacques André and Bruno Borghi. Dynamic fonts. In Jacques André and Roger D. Hersh, editors, *Raster Imaging and Digital Typography*, pages 198–204. Cambridge University Press, 1989. ISBN 0-521-37490-1.

[And90] Jacques André. The Scrabble font. *The PostScript Journal*, 3(1):53–55, 1990.

[Ann74] Mitsumasa Anno. *ABC Book* (in Japanese). Tankobon, 1974. ISBN-10 4-8340-0434-1, ISBN-13 978-4834004342.

[AO89] Jacques André and Victor Ostromoukhov. Punk: de METAFONT à PostScript. *Cahiers GUTenberg*, 4:123–28, 1989.

[CK14] Neill D. F. Campbell and Jan Kautz. Learning a manifold of fonts. *ACM Trans. Graph.*, 33(4):91:1–91:11, July 2014.

[Coh92] Michael Cohen. Blush and Zebrackets: Two Schemes for Typographical Representation of Nested Associativity. *Visible Language*, 26(3+4):436–449, Summer/Autumn 1992. `http://visiblelanguagejournal.com/issues/issue/98/`.

[Coh93] Michael Cohen. *Zebrackets*: A Pseudo-dynamic Contextually Adaptive Font. *TUGboat*, 14(2):118–122, July 1993. `http://tug.org/TUGboat/tb14-2/tb39cohen.pdf`.

[Coh94] Michael Cohen. Adaptive character generation and spatial expressiveness. *TUGboat*, 15(3):192–198, September 1994. Proceedings of the 1994 TUG Annual Meeting, Santa Barbara, CA. `http://tug.org/TUGboat/tb15-3/tb44cohen.pdf`.

[Coh14] Michael Cohen. From Killing Trees to Executing Bits: A Survey of Computer-Enabled Reading Enhancements for Evolving Literacy. In *VSMM: Proc. Int. Conf. on Virtual Systems and Multimedia*, Hong Kong, December 2014. `http://www.vsmm2014.org`.

[dK25] Guillaume Apollinaire (Wilhelm Apollinaris de Kostrowitzky). *Poémes de la paix et de la guerre 1913–1916 (Poems of war and peace 1913–1916)*. Nouvelle Revue Française, Paris, 1918, 1925.

[FVJ11] FL@33, Tomi Vollauschek, and Agathe Jacquillat. *The 3D Type Book*. Laurence King Publishing, 2011. ISBN-10 1856697134, ISBN-13 978-1856697132.

[Har07] Yannis Haralambous. *Fonts & Encodings*. O'Reilly, 2007. ISBN-10 0-596-10242-9, ISBN-13 978-0-596-10242-5.

[HF13] Steven Heller and Louise Fili. *Shadow Type: Classic Three-Dimensional Lettering*. Princeton Architectural Press, Thames and Hudson Ltd., 2013. ISBN-10 1616892048, ISBN-13 978-1616892043.

[Kar15] Peter Karow. Digital typography with Hermann Zapf. *TUGboat*, 36(2):95–99, 2015. `http://tug.org/TUGboat/tb36-2/tb113zapf-karow.pdf`.

[KH08] Robert Klanten and Hendrik Hellige, editors. *Playful Type: Ephemeral Lettering & Illustrative Fonts*. Dgv, 2008. ISBN-10 3899552202, ISBN-13 978-3899552201.

[Len91] John Lennard. *But I Digress: Parentheses in English Printed Verse*. Oxford University Press, 1991. ISBN 0-19-811247-5.

[MP12] Blanca Mancilla and John Plaice. Charactles: More than characters. In Cyril Concolato and Patrick Schmitz, editors, *ACM Symposium on Document Engineering*, pages 241–244. ACM, 2012.

◇ Michael Cohen
University of Aizu, Japan
`mcohen (at) u-aizu.ac.jp`

◇ Blanca Mancilla
Mentel, Montreal, Canada
`blancalmancilla (at) gmail.com`

◇ John Plaice
Grammatech, Ithaca, USA;
UNSW, Sydney, Australia
`johnplaice (at) gmail.com`

A Telegram bot for printing LaTeX files

Amartyo Banerjee and S.K Venkatesan

Abstract

A proof of concept of a Telegram bot running on a Raspberry Pi is described here. The bot will accept a LaTeX file from the user, process it and send back to the user a PDF file resulting from that processing. The following are discussed:

1. The genesis of the idea of the bot.
2. The use case for the bot as it exists at present, and after additional functionality is implemented.
3. The learning process and obstacles encountered in developing it.
4. Additional functionality planned to be implemented, such as processing a multi-file LaTeX document, and printing the PDF file.
5. Steps needed to make it production ready, including robust error handling and proper input sensitization.
6. Potential for a complete rewrite to meet scalability requirements and get around file download limitations in the Telegram bot API.

1 Introduction

The authors present a proof of concept of a Telegram [1] bot [2], meant to run on the Raspberry Pi [3]. The purpose of this bot is to accept (LA)TeX files submitted by a user running the Telegram client on a mobile phone or on a PC/laptop browser, and to send back a PDF file produced by compiling the (LA)TeX files using pdfTeX or XeTeX or whatever (LA)TeX compiler is invoked by the files.

2 Genesis of the idea

The idea for implementing this bot arose out of a brainstorming session when the various uses that could be made of the Raspberry Pi were discussed. The question that came up was if it is possible to run TeX on the Raspberry Pi. After checking online, it was found that not only could TeX be run on the Raspberry Pi, but people were in fact running it [4]. At this point we considered a potential use case for the Raspberry Pi, wherein people could type up a TeX file on their mobile phone, send it to server software running on the Pi which could compile the TeX file into PostScript or PDF and print it out. The person who had submitted the TeX file could then come and collect the typeset and printed output of that TeX file. This service could be a paid service.

Figure 1: A Raspberry Pi

3 The use case for the bot

At the moment the printing functionality is not yet implemented. Even the current functionality, however, has some use cases. A user could use (LA)TeX to write a music score [5], and see what the typeset output would look like. Or they could type a chess game of a certain move in LaTeX chess notation, and get back a view of what that looks like when typeset [6].

The idea is to eventually have a system where a person can submit one or more (LA)TeX files which will be processed and the resulting output will be printed out. The print can be collected later on, which would make the bot suitable for use in a print shop. So a user could submit a (LA)TeX file to the bot, it would be processed into a Postscript or PDF file and this file could then be printed out. The user could then pay to collect the print.

4 The learning process and obstacles encountered

Initially the idea was for the TeX files to be submitted via SMS. However, we decided against this on the basis of our recollection of the difficulties in working with SMS messages in computer programs, especially as there are also restrictions in India on how many SMS messages can be sent by a particular number and to a particular number, as also restrictions on what sort of attachments can be sent by SMS, and on their size. Last but not least each SMS costs money.

For these reasons we decided to use messaging software that does not depend on SMS messages to send messages and attachments. In the authors' circle of friends and acquaintances, the most often used messaging applications on their mobile phones are WhatsApp [7] and Viber [8], with more WhatsApp users than Viber, and those contacts in the first author's phonebook who use both are far more likely to check WhatsApp than Viber.

This seemed to make WhatsApp the software of choice for transmitting TeX files to the proposed server software. The case seemed to be strengthened by the fact that in past searches for ideas of things that could be done with a Raspberry Pi, the first author had seen a write-up on using WhatsApp on the Raspberry Pi to send notifications to the owner of the Raspberry Pi [9] whenever events occurred which the owner was interested in knowing about.

To the best of the first author's recollection however, in the comments made on the web page [9], it had been pointed out that the use of WhatsApp in this manner involved the use of reverse engineered knowledge of the WhatsApp protocol, which is proprietary. The protocol could be changed anytime by WhatsApp, which would break any software relying on the reverse engineered understanding of the protocol. Writing such software and using it also had the potential of violating WhatsApp's terms of service, which could result in termination of the WhatsApp account of the person whose mobile phone number had been used to register the unofficial software that used the reverse engineered WhatsApp protocol.

In these same comments, it was pointed out that in contrast to WhatsApp, the Telegram messaging application had an officially documented protocol [10], permitted the existence of open source clients [11], and provided not just one but two APIs [12] which could be used to interact with Telegram and its servers, and which could be used to write software using Telegram as the messaging part of that software.

As it turns out, the first author's recollection turns out to be faulty in this case. They have been unable to find any comments stating anything listed in the above two paragraphs. Perhaps such comments had indeed been seen on that website but in that case they have subsequently been deleted. Checking in the Wayback Machine [13] for past versions of the url in [9] does show certain comments which have been deleted from the live site, but not the comments the first author seems to recall. Alternatively, the first author saw such comments elsewhere but now cannot recall where that could have been, nor have they been able to discover in their recent web searches any other website containing such comments. It is possible that in the first author's memory they have conflated their reasoning with a url where they thought they had seen someone else make those points.

That being said, the substance of the first author's objections remains valid. The url mentioned in [9] refers to the use of a python library named Yowsup [14]. This library uses a reverse engineered API for WhatsApp, named Chat API [15]. On one of the Chat API web pages [16], they mention that they have received a cease and desist letter from WhatsApp's lawyers, a copy of which they have made publicly available at the url listed in [17]. Although the author of Chat API states in [16] that they will maintain the repository mentioned whose url is [15], the letter shown in [17] does confirm that Chat API is a reverse engineering of WhatsApp's API, and that WhatsApp considers the development and use of Chat API to be a violation of its terms of service.

It is also the case that the author of the tutorial in [9] mentions partway through [18] the risks of getting one's mobile phone number banned through repeated attempts to register the same number, and recommends using Telegram.

Remembering these objections, however hazily, when the time came to choose how a potential user could send TeX files to the proposed server software running on the Raspberry Pi, it was decided to use Telegram. The authors discussed the reasoning and concerns of the first author. Given that SMS is intrinsic to every mobile phone and WhatsApp is already widely installed the second author might have insisted on using either of these. Instead the second author agreed with the first author, reasoning that the potential users they had in mind could be persuaded to install Telegram on their mobile phones, and in any case at some future date a mobile phone application could be written integrating a text editor with the ability to send TeX files to the server software. In that case potential users would have to be persuaded to install that application on their phone anyway.

As mentioned above, Telegram provides a choice of two APIs to interact with it. One is an API to write a full-fledged Telegram client [19]. Our program, if written using the Telegram client API, would thus be a client from the point of view of the Telegram servers while also being a server. In other words, it would be a client from the point of view of the Telegram servers, but a server from the point of view of our users. The service provided by it would be the fact that as mentioned above, if sent a (LA)TeX file it would return a typeset PDF and/or print said PDF.

The second API [20] allows one to write a bot for Telegram. The exact definition of what a Telegram bot is and can do can be found on the Telegram website [2, 20]. That being said, the closest analogy that comes to mind is the variety of bots [21] written for IRC [22]. In some sense it seems to the authors that Telegram bots are a repackaging of an old idea for a generation of Internet users whose primary interaction with the Internet is via smartphones, who might have never come to know about IRC and IRC bots.

On reviewing the two APIs, it seemed to the first author that writing a bot might be an easier thing to get started with, especially if one wanted to get a proof of concept implementation up and running quickly. It helped that a write-up on how to implement a Telegram bot on the Raspberry Pi [23] was easily found. Before going any further it should be mentioned that a third possible way exists, which is to use a command line desktop/laptop Telegram client [11], which would run on the Raspberry Pi, and could be configured to perform certain tasks on certain events, such as the receipt of a file from a user, etc. This approach has both advantages and disadvantages. The disadvantages led to the bot approach being chosen, but the advantages might lead to the software being implemented using the command line Telegram client after all in a future iteration.

Just as in the case of running a Telegram bot on the Raspberry Pi, it was easy to find web pages giving instructions on using the Telegram command line client on the Raspberry Pi [24], as a means of controlling the Raspberry Pi remotely [25] and receiving notifications from the Raspberry Pi when certain events happened. Although we worked through the tutorials listed on those web pages, it was not immediately obvious how to extend the examples given to achieve what we wanted to achieve.

In contrast, the web pages providing instructions on implementing a Telegram bot on the Raspberry Pi [23] lead to a GitHub page containing the software used to implement the bot [26], which in turn had slightly more complex examples [27, 28] that could be extended to achieve what we wanted. The fact that the bot examples were written in Python, which the first author is more familiar with, as compared to Lua, which the Telegram command line client examples were written in, was also a factor that led the first author to choose the bot approach. Yet another factor in choosing the bot approach was the fact that using the Telegram client required the first author to use their mobile phone number to register with the Telegram server [24], as compared to a bot, which does not require a mobile phone number [20, 23].

Nevertheless, there is one advantage of using the Telegram command line client which may lead to it being used in a future iteration of the software. The advantage is that as per the existing Telegram bot API, the maximum size of a file that may be downloaded by a bot from the Telegram servers is 20MB [29]. This means of course, that the maximum size of a file that can be sent by a user of our software is 20MB. For the command line client, as also official Telegram clients released for various platforms, the size limitation is much higher, perhaps even as much as an order of magnitude.

Now, 20MB seems like more than enough for (LA)TEX files, which are after all like source code. However, one can never predict how large a (LA)TEX file a user might wish to compile and print. Furthermore, this 20MB limit can be reached much more quickly if the user chooses to make references to figures and other graphics files in their (LA)TEX files, to be included in the final typeset PostScript/PDF file.

During discussions, the second author stated that they did not anticipate many of the users they had in mind needing to exceed that 20MB limit, so for now the bot approach is the only implementation that exists. If the software is found generally useful, a new implementation using the command line client approach will be done in future.

The implementation of the code took a few days. Much of this time was spent working through the tutorial for implementing a Telegram bot on the Raspberry Pi [23], trying to understand and then extend the more complex examples given on the website of the software used to implement the bot API [27, 28, 30], known as telepot [26], trying to understand the Telegram bot API [31] per se, and simply brushing up the first author's Python and general programming knowledge.

Further time was taken up when the function provided by telepot to download a file [32] was found not to work. In this case that meant the code would hang at that point, until one was forced to kill the bot. Initially a workaround was coded whereby `wget` [33, 34] was used to directly download the file from the Telegram servers. After digging into the implementation of the file download function, and doing some research online [35], using comments in the telepot source code itself [36], we were able to patch the telepot source files to make the download file function work correctly. The patch itself is trivial although the effort to figure out what to do was not. It will be submitted to the author of telepot.

There was also the time involved in searching for LATEX server implementations, i.e. some software that listens on the network, receives LATEX files from clients and does something with it, either compile it and send back the typeset output or print it or something else. We found a few interesting links but nothing that fit our purpose [37–41]. On the other hand, those web searches did provide pointers to what was eventually implemented [42, 43].

5 User experience and implementation

As currently implemented, the user will have the following experience when interacting with the bot.

First, the prerequisites. The user will have to download and install the official Telegram client for their smartphone [44–46]. There is a registration process which requires providing a mobile phone number, which is used by the Telegram server for verification and to complete the installation of the client program. This is similar to the process used to install other mobile phone centric messaging applications like WhatsApp and Viber. Any user who has successfully installed and used these other applications to communicate with others should have no problem completing this process on Telegram.

After this, they have to search for the bot, named AmartyoFirstBot. On selecting this bot from the search results, initially a button labelled 'Start' will appear, which has to be pressed. According to the bot API documentation [47], this is one of the commands to which the bot must give a response, but at present it does nothing. At this point a chat session will be opened with the bot. The interface at this point is similar to opening a chat with a user on the other messaging applications mentioned earlier.

Now the user has to select an attachment to send, by pressing the button on the phone touchscreen that resembles a paper clip. This brings up a series of icons, which are meant to select photos, videos, music, voice clips and documents. The icon to send documents must be selected, and then a LaTeX file selected using the file manager interface. Once selected the user is taken back to the chat session, and an icon appears the pressing of which sends the LaTeX file to the bot.

It must be emphasized at this point that the process of installing Telegram, searching for a contact or another Telegram user or a bot, is identical for every Telegram user whether or not they ever interact with our bot. The same applies to starting a chat session, either with a user or a bot, and for selecting an attachment to send and sending it.

What is unique in our case is that when a LaTeX file is sent, the user will ultimately see that a bot has sent a PDF file as a message, and will have an option to download it. On downloading it, they can tap on the icon and the PDF file will display in whichever application is configured to display PDF files.

On the server side, the bot, on receiving a chat message, checks if it has been sent a document, and if that document is a LaTeX file. If so, it downloads that file and saves it in a temporary directory. It then changes to that directory and invokes the `latexmk` [48] command on the received LaTeX file, using Python's facility for calling external programs. In the invocation it passes certain parameters to `latexmk`, along with the name of the LaTeX file. One of these specifies that `latexmk` should try to create a PDF file, which by default `latexmk` does by invoking the `pdflatex` [49] command. The bot checks the return code of `latexmk`, and if the return code indicates that `latexmk` succeeded, it proceeds to send the PDF file created by `latexmk` to the user who had sent the LaTeX file. At this point the bot changes back to the directory it was in before it received the LaTeX file. It continues running, waiting for messages.

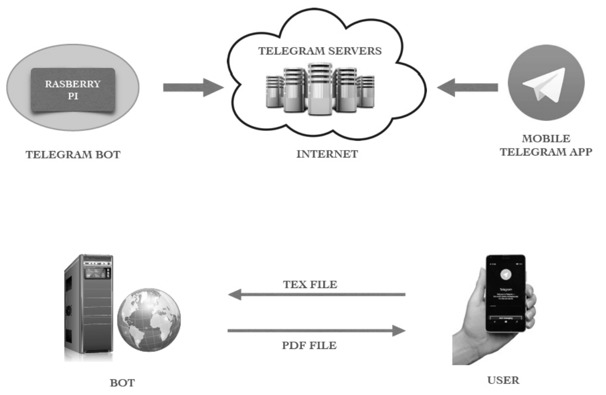


Figure 2: The LaTeX bot

6 Future functionality

As mentioned above, the bot is strictly a proof of concept at this point. It lacks functionality and also robust error handling, not to mention any effort at sanitizing what it receives from the user. In terms of functionality, the most obvious thing missing is the ability to handle anything more than a single TeX file. Thus, a user who has written a thesis or report in LaTeX, which typically include a single master TeX file with reference to multiple other LaTeX files, each containing either the text of a chapter, or maybe references to other LaTeX files, as well as figures and other pictures, will not be able to use the bot to compile and print that report or thesis.

The authors have an idea for implementing this functionality. It involves allowing the user to send the bot a zip file containing all the LaTeX and other files needed to compile and produce a report, such as figures. It will be the user's responsibility to make sure the folder structure inside the zip file corresponds to the declarations in the main LaTeX file, such that invoking `latexmk` on the main file will successfully find all the other LaTeX files and any (Encapsulated) PostScript or other graphics files needed, as well as any non-standard fonts or other files. If the user wishes to use fonts beyond those that every TeX installation is expected to have by default, it will be their responsibility to include those font files inside the zip file at the correct path such that latexmk can find them.

The bot will expect the name of the zip file, the part before the .zip extension, to be the same as the name of LaTeX file inside it. This LaTeX file containing the same name as the zip file will be expected to be the main LaTeX file, on which latexmk will be invoked. On successful compilation, the resulting PDF file will be sent back to the user and/or printed.

Other ideas for functionality include options for letting the user indicate if they wish to simply receive the PDF file or to actually print it. In the latter case, they will initially receive the PDF file to verify that the output is as they expect, and then an option to approve the final print. This will be important in case the bot is used in a business, such as a print shop. Requiring the user to approve the PDF before printing should help in avoiding disputes where the user claims the printed output does not look like they wanted it to and refuses to pay for the print.

At this point it is not clear how this "approve before printing" functionality will be implemented. The Telegram bot API already provides a variety of information such as the user id, the chat session id, the message id, etc. This will have to be kept track of in some sort of database, using which the bot will be able to know whom it received a particular TeX or zip file from, that the compilation succeeded and the PDF file was successfully sent back to that user, and that the user has approved it and agreed to print it. Some sort of record will have to kept of the fact that a user has approved a printout. On the user end, perhaps a combination of custom keyboards, which is functionality provided by the Telegram bot API, can be used to provide a user interface for this approval functionality.

Other functionality that needs to be implemented is the /start command, which every bot is expected to implement, along with a few others [47].

7 Becoming production ready

As far as error handling is concerned, at present if latexmk returns a non-zero error code, the bot simply exits with an error code of 1. This is obviously not suitable for production purposes. If latexmk fails to compile the TeX document, the bot should send relevant error messages back to the user, to enable the user to correct whatever errors caused latexmk to fail, whether in the syntax of a single LaTeX file, or in the paths where other LaTeX files or graphics/font files are supposed to exist.

It might also be the case that latexmk fails because the user intended to produce PostScript as the final output, not PDF, and that pdfLaTeX has failed because the LaTeX files refer to PostScript files for use as figures, not Encapsulated PostScript. In that case, since PostScript is perfectly suitable for printing, the bot should attempt to invoke latexmk with the option to produce a PostScript file as the final output, not PDF. Software for viewing PDF files is widespread nowadays, but this is not the case for PostScript files. For the purpose of sending something back to the user, the PostScript file produced by latexmk should be converted using ps2pdf [50], and this PDF file can then be sent by the bot. On the other hand, the PostScript file produced by latexmk should be the one sent to the printer by the bot.

Similarly, the user may have intended for their files to be compiled by XeLaTeX [51, 52] rather than pdfLaTeX, in which case the bot should invoke latexmk appropriately.

Perhaps one approach is to invoke latexmk by default with the -pdf option, then try with the -xelatex option, and if that fails try with the -ps option. If all these fail, the bot should give up and send an appropriate error message back to the user.

A more tricky case arises in cases where latexmk returns 0, but the resulting PDF still does not contain what the user intended. This can happen for example when trying to typeset music using LaTeX and the abc [53] notation. We noticed that if there was a syntax error in the abc notation within the LaTeX file [54, 55], a PDF file might get successfully created but with the actual typeset music missing. This is because in this case latexmk, or even pdflatex, invokes another program called abcm2ps [56, 57], which fails to compile the erroneous abc syntax. The abcm2ps program does print error messages indicating that it has failed. However, it either does not exit with appropriate error codes, or those are ignored by pdflatex and/or latexmk, most likely pdflatex. As a result latexmk returns 0 to indicate success when in fact this is not the case.

Perhaps one option is to check if the LaTeX file is using the relevant LaTeX package for abc notation, and in that case check the output of latexmk for the known error messages by abcm2ps indicating that it has failed. In which case the misleading zero exit code of latexmk should be ignored, and the bot should send back the relevant error messages of abcm2ps. However, this will make the bot code more complicated, so perhaps a better approach is to make changes to either abcm2ps or pdflatex, which ever it needs to be, such that if abcm2ps fails then pdflatex should exit with a non-zero value.

What is to be done if latexmk received a signal [58] causing it to exit uncleanly is not clear to the authors at the moment. It seems to the first author at present that the options are for the bot to treat it as a transient error and try invoking latexmk again, or assume that something is seriously wrong with the system it is running on, try to send an appropriate error message to the user, and exit as cleanly as it can. More thought and discussion is required before any decision can be taken.

Last but not least, to be production ready, the bot must implement sanitization of inputs submitted by the user. This is a necessity for any internet facing server software in this era of SQL injection attacks [59], cross site scripting attacks [60], privilege escalation [61] and arbitrary remote code execution vulnerabilities [62], which are often made possible by not sanitizing or improperly sanitizing user inputs [63, 64], especially when provided by some unknown and untrusted user over the Internet. Measures to be taken that come to mind immediately are making sure the names of files submitted by the user are sane, and that certain metadata needed by the bot are present, even in cases where the bot API says such metadata is optional [65]. Other measures include making sure that the contents of files submitted by the user match their claimed mime type, and that they are not in fact viruses and/or some other malware. This is a subject in which the first author does not have much expertise, and more research is needed to ensure that the bot implements proper input sanitization.

8 Potential for a complete rewrite

In the long run the bot may need to be re-written in Python 3, simply to take advantage of the fact that telepot provides an API to write asynchronous code, which might well be a necessity for scalability reasons at some point in the future, but only for Python 3 [66]. This will involve learning about the differences between Python 2 and Python 3, and about how to program asynchronously in Python 3, and then how to use telepot in an asynchronous manner. This will take quite a lot of time, so we only expect to do it if and when the bot becomes popular enough that scalability issues matter. At any rate, it will be done after implementing necessary functionality and fixing the error handling and sanitizing user inputs as listed above.

9 Epilogue

At the time of completion of the first draft of this paper, the bot had been written and was running on the first author's laptop. The Raspberry Pi on which the bot was meant to run was delivered to the first author on 10th June, 2016. Subsequently the Raspberry Pi was powered up and made to run and the prerequisites for getting the bot running on the Raspberry Pi were installed and configured. As of the current date the bot is installed on the Raspberry Pi. It currently needs to be started manually, once the Raspberry Pi is powered up.

Making sure the bot starts running automatically upon powering up the Raspberry Pi is one of the improvements to be done in future, along with all the other improvements outlined above, including printing functionality.

References

[1] telegram.org
[2] core.telegram.org/bots
[3] www.raspberrypi.org/help/what-is-a-raspberry-pi
[4] www.raspberrypi.org/forums/viewtopic.php?f=63&t=8279
[5] martin-thoma.com/how-to-write-music-with-latex
[6] www.highschoolmathandchess.com/2011/10/06/creating-chess-diagrams
[7] www.whatsapp.com
[8] www.viber.com/en
[9] www.instructables.com/id/WhatsApp-on-Raspberry-Pi/?ALLSTEPS
[10] core.telegram.org/mtproto
[11] github.com/vysheng/tg
[12] core.telegram.org/api
[13] archive.org/web
[14] github.com/tgalal/yowsup
[15] github.com/mgp25/Chat-API
[16] github.com/mgp25/Chat-API/wiki/WhatsApp-incoming-updates#11-july-2015
[17] www.docdroid.net/gWpFsXz/whatsapps-cease-and-desist-and-demand-against-chat-api.pdf.html
[18] www.instructables.com/id/WhatsApp-on-Raspberry-Pi/step2/Registration
[19] core.telegram.org/api#telegram-api
[20] core.telegram.org/api#bot-api
[21] en.wikipedia.org/wiki/IRC_bot
[22] en.wikipedia.org/wiki/Internet_Relay_Chat
[23] www.instructables.com/id/Set-up-Telegram-Bot-on-Raspberry-Pi/?ALLSTEPS
[24] www.instructables.com/id/Telegram-on-Raspberry-Pi/?ALLSTEPS
[25] www.instructables.com/id/Raspberry-remote-control-with-Telegram/?ALLSTEPS

[26] github.com/nickoala/telepot

[27] github.com/nickoala/telepot/blob/master/examples/simple/skeleton.py

[28] github.com/nickoala/telepot/blob/master/examples/simple/skeleton_route.py

[29] core.telegram.org/bots/api#getfile

[30] github.com/nickoala/telepot/blob/master/doc/reference.rst

[31] core.telegram.org/bots/api

[32] github.com/nickoala/telepot/blob/master/telepot/__init__.py#L460

[33] en.wikipedia.org/wiki/Wget

[34] www.gnu.org/software/wget

[35] stackoverflow.com/questions/17285464/whats-the-best-way-to-download-file-using-urllib3

[36] github.com/nickoala/telepot/blob/master/telepot/__init__.py#L473

[37] alex.nederlof.com/blog/2013/02/22/latex-build-server

[38] web.archive.org/web/20120120210031/http://bugsquash.blogspot.com/2010/07/compiling-latex-without-local-latex.html

[39] scribtex.wordpress.com/2010/01/17/the-common-latex-service-interface

[40] github.com/jpallen/clsi

[41] launchpad.net/rubber

[42] superuser.com/questions/173914/dropbox-latex-automated-pdf-compile

[43] latex-community.org/forum/viewtopic.php?f=28&t=9512

[44] telegram.org/dl/android

[45] telegram.org/dl/ios

[46] telegram.org/dl/wp

[47] core.telegram.org/bots#global-commands

[48] users.phys.psu.edu/~collins/software/latexmk

[49] tug.org/applications/pdftex

[50] www.ghostscript.com/doc/current/Ps2pdf.htm

[51] xetex.sourceforge.net

[52] tug.org/xetex

[53] abcnotation.com

[54] martin-thoma.com/how-to-write-music-with-latex/#abc

[55] martin-thoma.com/how-to-write-music-with-latex/#example

[56] moinejf.free.fr

[57] abcplus.sourceforge.net/#abcm2ps

[58] en.wikipedia.org/wiki/Unix_signal

[59] en.wikipedia.org/wiki/SQL_injection

[60] en.wikipedia.org/wiki/Cross-site_scripting

[61] en.wikipedia.org/wiki/Privilege_escalation

[62] en.wikipedia.org/wiki/Arbitrary_code_execution

[63] en.wikipedia.org/wiki/Improper_input_validation

[64] xkcd.com/327

[65] core.telegram.org/bots/api#document

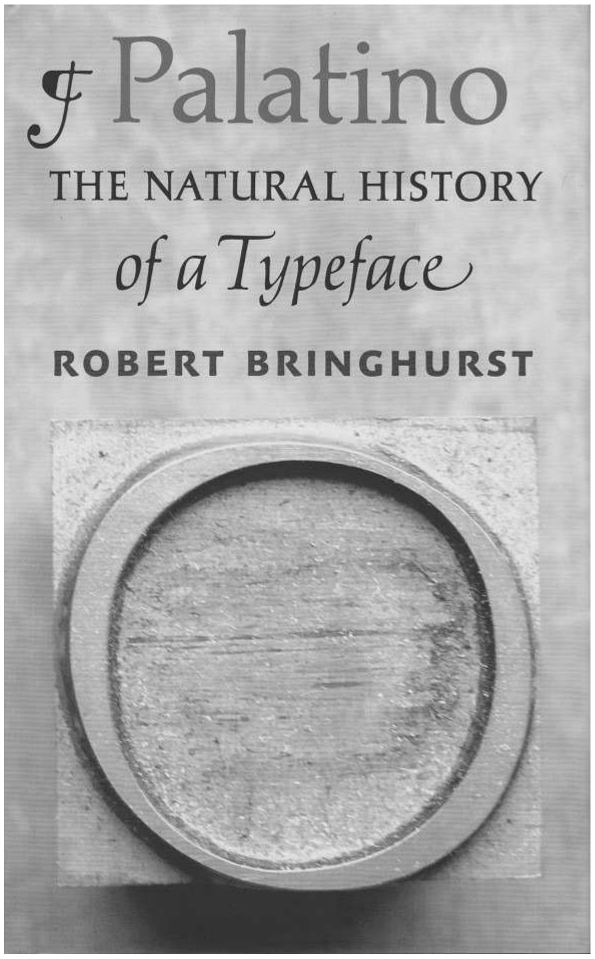
[66] github.com/nickoala/telepot#async

◇ Amartyo Banerjee and S.K Venkatesan
TNQ Books and Journals
Chennai, India
http://tnqsoftware.co.in

Book review: *Palatino: The natural history of a typeface* by Robert Bringhurst

Boris Veytsman

Robert Bringhurst, *Palatino: The natural history of a typeface*. David R. Godine, Publisher; Boston, 2016, 296pp, ill. US$65.00. ISBN 978-1-56792-572-2.



The participants of TUG2016 in Toronto had the rare treat of attending lectures by two prominent contemporary typographers, Robert Bringhurst and Chuck Bigelow. *The Elements of Typographic Style* by Bringhurst is considered one of the most influential treatises on book design; Hermann Zapf wished "to see this book become the Typographers' Bible".

Interestingly enough, both Bringhurst and Bigelow discussed major typeface designs. The latter told the story of *Lucida* by Bigelow and Holmes, while the former discussed the *Palatino* family by Hermann Zapf. Bringhurst's lecture could be considered a presentation of his book *Palatino*, published in a limited (order now) trade edition this fall by David R. Godine.

The name Hermann Zapf strikes many chords in the TeX community. He collaborated with DEK, and was until his death an honorary member of the TUG board, the Wizard of Fonts. Thus while a book by a great typographer about a great font designer is a gift to any bibliophile, this book has a special meaning for our community.

The book discusses the long evolution of the Palatino family by Zapf. It lists all known variants from No. 1, PALATINO TEXT ROMAN TRIAL CUTTING (1949) to No. 112, ALDUS NOVA BOLD ITALIC, released in True Type and Open Type in 2005 (the classification and enumeration are by Bringhurst). Hermann Zapf was notable for his eager embrace and understanding of new technologies, and Palatino fonts were made for many different typesetting systems: letterpress, Linotype, phototypesetting and many digital formats. TeX users are quite familiar with this font family. The printed version of this review is typeset in Adobe Palatino with LaTeX package *mathpazo* (using `sc` and `osf` options for real Small Caps and old style figures in text). This font remains one of the most elegant and noble fonts in modern digital typography.

The technological changes require changes in the fonts themselves: the color and feel of copy produced by different means are quite different. The subtle changes in each redesign of Palatino show the care and skill of the great master Zapf. The font family includes Cyrillic and Greek letters as well as accented Latin ones, titling fonts, and many other typographic niceties.

The world of font design, even when we talk about one (admittedly large) family, is complex. A journey into this world requires a wise guide, generous to share his knowledge and experience with the reader. Robert Bringhurst is, without a doubt, such a guide. His book fortunately avoids the trap of becoming a catalog of font designs, interesting only to a few connoisseurs. Instead, the lucid explanations of the reasons behind the evolution of the design, the challenges and Zapf's ingenuity in meeting them, make reading the book a wonderful experience. For example, he devotes several pages to study of just one character, the humble asterisk (∗), and uncovers the beauty behind this modest typographic device.

Besides being a typographer, Robert Bringhurst is a renowned poet, and this shows in the book. For example, it is clearly seen in the description of the difference between the original Palatino and the later variation Aldus:

> Aldus is not just narrower than Palatino, it also has a slightly lighter stroke and smaller

> x-height with taller ascenders. The transition from thick to thin (or from pull stroke to edge stroke), which in Palatino often have a slightly angular articulation, are more distinctly and consistently angular in Aldus. This gives a page of Aldus greater crispness — some would say coldness — than a page of Palatino. If Palatino is like a big, round, fully flavored red wine, then Aldus is like a flinty, dry white — equally deep but more narrow in flavor, and best served chilled, to keep the flavors closely focused. The proportions of the letters, with their modest eye and tall ascenders, emphasize their Italian humanist heritage, yet the hint of angularity in the round forms also alludes, ever so subtly, to the blackletter tradition. This allusion is reinforced by the shapes of Aldus apostrophe and quotation marks: they are long, sloped, tapered but uncurved — something rarely found in roman and italic but altogether typical of fraktur.

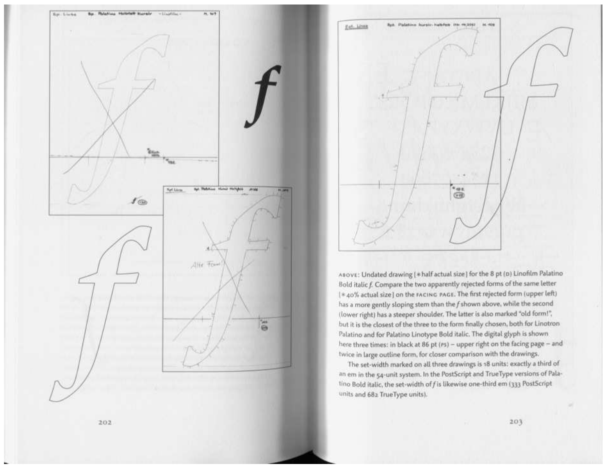
The same poetic eye shows in the best description of the difference between serifed and sans serif fonts I ever read:

> Serifs — those little entry and exit strokes through which the writing hand and the reading eye like to find their way into and out of a letterform — are also a means by which letters tie themselves into a line: a form of graphic social bonding. They are as old as the letters themselves; but sanserif letters — the socially disengaged — are no younger.

Several good metaphors are used throughout the book, illuminating its main themes. Bringhurst compares the fonts to classic musical instruments like cellos and pianos. This metaphor becomes especially apt when he notes that it is difficult to judge some variants of Palatino because they were not actually used — as we cannot judge an instrument which was never played by a skilled musician.

Another important comparison is between typography and architecture. Bringhurst spends some time discussing *entasis:* a slight convexity or concavity of lines, in architecture and font design. His juxtaposition of the profile of classic columns and the elegant curves of the uppercase Roman "I" in Palatino is quite enlightening.

Bringhurst further compares classification of fonts to botany, where a naturalist must decide which plants are close relatives, which plants belong to different species and which are subspecies


ABOVE: Undated drawing [∗half actual size] for the 8 pt (D) Linofilm Palatino Bold italic *f*. Compare the two apparently rejected forms of the same letter [∗40% actual size] on the FACING PAGE. The first rejected form (upper left) has a more gently sloping stem than the *f* shown above, while the second (lower right) has a steeper shoulder. The latter is also marked "old form!", but it is the closest of the three to the form finally chosen, both for Linotron Palatino and for Palatino Linotype Bold italic. The digital glyph is shown here three times: in black at 86 pt (PS) – upper right on the facing page – and twice in large outline form, for closer comparison with the drawings.

The set-width marked on all three drawings is 18 units: exactly a third of an em in the 54-unit system. In the PostScript and TrueType versions of Palatino Bold italic, the set-width of *f* is likewise one-third em (333 PostScript units and 682 TrueType units).

202

203

Figure 1: Zapf's drawings for Linofilm Palatino Bold Italic *f*

of the same kind. This is actually a fundamental metaphor for the book: not for nothing does the latter have the ambitious subtitle *The natural history of a typeface*. Bringhurst clearly sees his work as akin to the work of Linnaeus and Cuvier and makes us recall the times when the lines between a scientist, an artist and a poet were somewhat blurred.

As a poet, Bringhurst does not just reveal profound truths in nature and art. His eye sometimes turns to society, and again his observations are deep and revealing. For example, he discusses the changes made by Zapf to the fonts when the Stempel foundry prepared variants for sale in North America in the 1950s. To make the fonts closer to "the limits of American typographical taste", Zapf revised nine letters and recut two ligatures, making the result "tamed to suit the *goût américain*." Bringhurst discussed the changes, and then surprisingly notes the difference between the copy used to demonstrate the fonts:

> The European specimens are full of quotations from Goethe and Shakespeare, sample title pages for books by Bertrand Russell and George Bernard Shaw, sample posters for art exhibitions, menus for five- and six-course meals, business cards for barristers and physicians, and snippets of typographic history. The American materials demonstrate instead how to use some of the world's most elegant printing types to say such things as "Best 100-watt bulb ever" and "Buy wash-and-wear! Not wash-and-beware!"

The book is very well illustrated. Font samples, drawings, photographs help the author to convey his thoughts clearly and convincingly (see, for

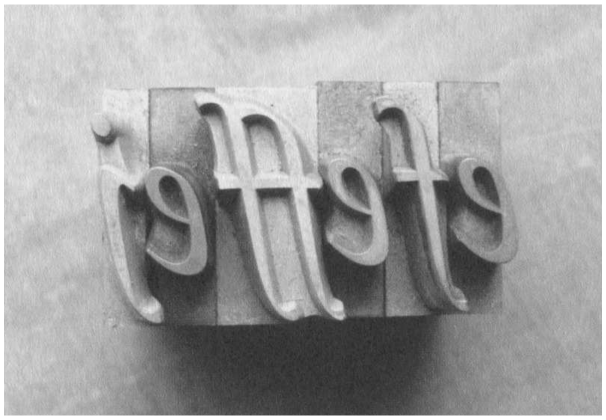
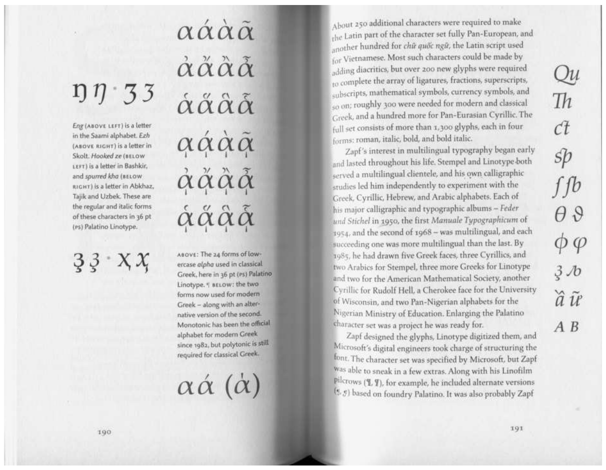


Figure 2: Kerning and ligatures in metal



ŋ *ŋ* · ʒ *ʒ*

Eng (ABOVE LEFT) is a letter in the Saami alphabet. *Ezh* (ABOVE RIGHT) is a letter in Skolt. *Hooked ze* (BELOW LEFT) is a letter in Bashkir, and *spurred kha* (BELOW RIGHT) is a letter in Abkhaz, Tajik and Uzbek. These are the regular and italic forms of these characters in 36 pt (PS) Palatino Linotype.

ᶾ *ᶾ* · ҳ *ҳ*

α ά ὰ ᾶ

ἀ ἄ ἂ ἆ

ἁ ἅ ἃ ἇ

ᾳ ᾴ ᾲ ᾷ

ᾀ ᾄ ᾂ ᾆ

ᾁ ᾅ ᾃ ᾇ

ABOVE: The 24 forms of lowercase *alpha* used in classical Greek, here in 36 pt (PS) Palatino Linotype. ¶ BELOW: the two forms now used for modern Greek – along with an alternative version of the second. Monotonic has been the official alphabet for modern Greek since 1982, but polytonic is still required for classical Greek.

α ά (ά)

190

About 250 additional characters were required to make the Latin part of the character set fully Pan-European, and another hundred for *chữ quốc ngữ*, the Latin script used for Vietnamese. Most such characters could be made by adding diacritics, but over 200 new glyphs were required to complete the array of ligatures, fractions, superscripts, subscripts, mathematical symbols, currency symbols, and so on; roughly 300 were needed for modern and classical Greek, and a hundred more for Pan-Eurasian Cyrillic. The full set consists of more than 1,300 glyphs, each in four forms: roman, italic, bold, and bold italic.

Zapf's interest in multilingual typography began early and lasted throughout his life. Stempel and Linotype both served a multilingual clientele, and his own calligraphic studies led him independently to experiment with the Greek, Cyrillic, Hebrew, and Arabic alphabets. Each of his major calligraphic and typographic albums – *Feder und Stichel* in 1950, the first *Manuale Typographicum* of 1954, and the second of 1968 – was multilingual, and each succeeding one was more multilingual than the last. By 1985, he had drawn five Greek faces, three Cyrillics, and two Arabics for Stempel, three more Greeks for Linotype and two for the American Mathematical Society, another Cyrillic for Rudolf Hell, a Cherokee face for the University of Wisconsin, and two Pan-Nigerian alphabets for the Nigerian Ministry of Education. Enlarging the Palatino character set was a project he was ready for.

Zapf designed the glyphs, Linotype digitized them, and Microsoft's digital engineers took charge of structuring the font. The character set was specified by Microsoft, but Zapf was able to sneak in a few extras. Along with his Linofilm pilcrows (¶, ¶), for example, he included alternate versions (§, §) based on foundry Palatino. It was also probably Zapf

Qu *Th* *ct* *sp* *ſ ſb* θ ϑ ϕ φ ᶾ *ᶀ* ȃ *ũ* *A B*

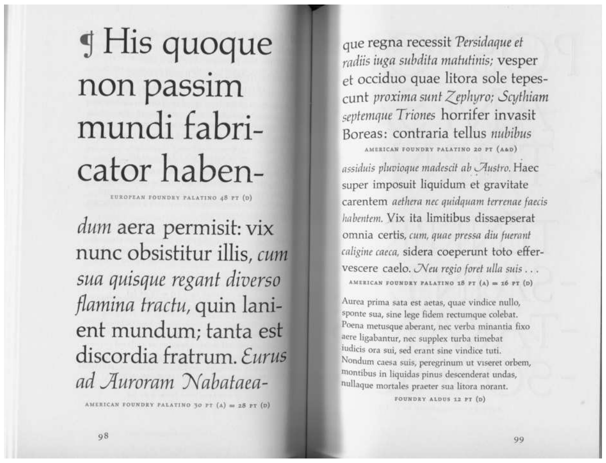
191

Figure 3: A spread from the book

example, Figure 1, showing the pages describing the making of LINOFILM PALATINO BOLD ITALIC ***f***). Many illustrations will warm the heart of a typophile; for example, the one in Figure 2 shows how kerning and ligatures were done in metal: sorts are shaved on the sides to effect kerning and there are separate sorts for ligature glyphs.

The book was designed by Robert Bringhurst himself, and the design is daring and beautiful. It uses margins for a wonderful variety of illustrations; full page and part-page ones subtly interplay, as shown in Figure 3. The main text is typeset in Aldus Buchschrift and Palatino Sans. The book includes letterpress pages printed by the well regarded book designer and artist Jerry Kelly (Figure 4).

The publisher of the book, David R. Godine, is also well known among bibliophiles. We have reviewed several books from his catalogue in these pages, and an article about his work appeared



¶ His quoque non passim mundi fabricator haben-

EUROPEAN FOUNDRY PALATINO 48 PT (D)

dum aera permisit: vix nunc obsistitur illis, *cum sua quisque regant diverso flamina tractu,* quin lanient mundum; tanta est discordia fratrum. *Eurus ad Auroram Nabataea-*

AMERICAN FOUNDRY PALATINO 30 PT (A) = 28 PT (D)

98

que regna recessit *Persidaque et radiis iuga subdita matutinis;* vesper et occiduo quae litora sole tepescunt *proxima sunt Zephyro; Scythiam septemque Triones* horrifer invasit Boreas: contraria tellus *nubibus*

AMERICAN FOUNDRY PALATINO 20 PT (A&D)

assiduis pluvioque madescit ab Austro. Haec super imposuit liquidum et gravitate carentem *aethera nec quidquam terrenae faecis habentem.* Vix ita limitibus dissaepserat omnia certis, *cum, quae pressa diu fuerant caligine caeca,* sidera coeperunt toto effervescere caelo. *Neu regio foret ulla suis . . .*

AMERICAN FOUNDRY PALATINO 18 PT (A) = 16 PT (D)

Aurea prima sata est aetas, quae vindice nullo,
sponte sua, sine lege fidem rectumque colebat.
Poena metusque aberant, nec verba minantia fixo
aere ligabantur, nec supplex turba timebat
iudicis ora sui, sed erant sine vindice tuti.
Nondum caesa suis, peregrinum ut viseret orbem,
montibus in liquidas pinus descenderat undas,
nullaque mortales praeter sua litora norant.

FOUNDRY ALDUS 12 PT (D)

99

Figure 4: An example of letterpress pages

in a recent issue (David Walden, *Note on the publisher of the Bodoni book: David R. Godine*, TUGboat **37**:1, pp. 97–98, 2016, `http://tug.org/TUGboat/tb37-1/tb115walden.pdf`). The book is beautifully printed and bound.

Of course, even the best book ever printed could be made better. I missed two features in this book, one minor, one more important. First, the list of fonts and font variants in the end of this book would be more useful if accompanied by the page numbers where the font is discussed or shown. Second, if a font is a musical instrument, then a discussion of it is not complete without a sketch of musicians who played it and the pieces where it shone brightest. While it would be impossible to list *all* books and typesetters working with Palatino (which is more evidence of the greatness of Zapf's creation), it could be interesting and useful to mention at least some notable publications that involved Palatino.

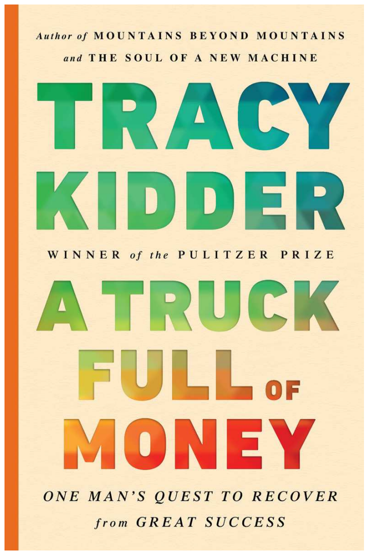
Still, these gripes are about the things that are *not* in the book. The things that *are* there, in my opinion, are more than sufficient to justify its price for any typophile, bibliophile or anyone interested in the history and art of making elegant fonts & beautiful books.

◇ Boris Veytsman
Systems Biology School and
Computational Materials
Science Center, MS 6A2,
George Mason University,
Fairfax, VA 22030
`borisv (at) lk dot net`
`http://borisv.lk.net`

Book review: *A Truck Full of Money* by Tracy Kidder

David Walden

Tracy Kidder, *A Truck Full of Money*. Random House, 2016, xxiv+260 pp. Hardcover, US$28.00. ISBN 9780812995224.[1]


Author of MOUNTAINS BEYOND MOUNTAINS
and THE SOUL OF A NEW MACHINE
TRACY KIDDER
WINNER of the PULITZER PRIZE
A TRUCK FULL OF MONEY
ONE MAN'S QUEST TO RECOVER
from GREAT SUCCESS

In 1981 Tracy Kidder published *The Soul of a New Machine*, about the team effort to develop a new computer at Data General. Now, over thirty years and eight books later, Kidder has returned to a topic in the world of computing. *A Truck Full of Money* is primarily a profile of computer programmer and serial entrepreneur Paul English, who may be best known as one of the founders of Kayak which was sold in 2012 to Priceline for $1.8 billion (including, as noted in the book, $120 million for English).

The book also gives insight into the mentalities and sensibilities of various (top notch) computer programmers associated with English over the decades: how much they enjoy programming (or at least how easily it comes to them), how they think about the risk of joining a startup, their reactions to having financial success and failure, and so forth. English is especially conflicted about making money; the jacket line below the title on the front cover of the book is, "One man's quest to recover from great success."

Paul English, now in his early 50s, grew up in a large working class family in Boston. He was not a particularly good or well-behaved student at Boston's renowned Latin (High) School, but he did find computers there and found that he was a natural with them. He went to the University of Massachusetts Boston for college, taking many classes at night so he could work days to pay his way through college, and over seven years received Bachelor's and Master's degrees in computer science. More importantly, perhaps, he fell in with a small group of gifted computer programmers there (including Karl Berry) and faculty member Robert Morris. In the late 1980s, English got a programming job at Interleaf through Morris who worked there part time.

At Interleaf, English once again found himself in a group of like-minded people (compatible technical viewpoints), who worked well together, and were happy that English was willing to be "the manager". He got things done, and he rose through the ranks relatively quickly to senior vice president positions of engineering and product marketing.

At Interleaf and in later entrepreneurial activities (sketched over the course of the rest of the book), English "collected" smart people, various of whom joined him in startup ventures after his time at Interleaf (and Intuit which bought Interleaf while English was there). His latest startup, Lola, was being developed as this book was being written and was announced between the time the author sent his final manuscript to the publisher and the date of the book's publication (`tinyurl.com/globekidder`).

It seems to me that author Tracy Kidder likes to tell a certain kind of story in the nonfiction he writes. He focuses on a few characters and their conflicts (often internal rather than with each other or with outside forces). He likes craft, for example, building construction in *House* (1985) and computer programming in the current book. His characters are fundamentally good people, albeit with foibles, who are on a mission — people that readers, the author, and other people in the story can admire despite whatever warts they may have — people who other people want to follow, or at least put up with.

Paul English is such a character. He has an idea a minute, he makes things happen, and he can be brusque with people working for him even though he likes them and mostly he is a great boss. People in turn want to follow him, including those who tamp down some of his wildest ideas and execute his more plausible ideas. They trust him and believe they can count on him despite some of his wild or idiosyncratic ways. English is particularly conflicted about money: he makes it, he risks it, he gives it away. An important aspect of the book is the author, the reader, and English himself trying to figure out what makes him tick.

I recommend the book to readers. As is typical for Tracy Kidder, the book is easy to read with frequent

[1] A second review is at `http://walden-family.com/ieee/truckfull.pdf`.

nice turns of phrase; it features a quite fascinating character and sketches activities in domains with which the reader is perhaps not familiar — in this case the worlds of computer programming, high tech start-ups, and relatively hands-on philanthropy. The book may be particularly relevant to members of the TEX and TUG communities: (1) Donald Knuth and TEX are discussed several times over the course of the book; (2) Karl Berry plays an interesting role in the Paul English story told by Kidder.

More personally, I especially enjoyed the book as a one-time computer programmer myself. I got to relive through Kidder's prose some of the joy, excitement, and feelings of changing the world that come with creating innovative computer programs and finding users for them in the world beyond the development lab.

The first Kidder book I read was *The Soul of a New Machine* (1981); it was a revelation to me in how engaging nonfiction writing about computing could be. I don't think I read another of his books until *Mountains Beyond Mountains* (2003). Reading that took me to the library and bookstore to obtain and read five more books in his oeuvre. Now reading *A Truck Full of Money* has reminded me that I need to get my hands on *Among Schoolchildren* (1989), *Old Friends* (1993), and *Home Town* (1999).

The attendees at TUG 2014 in Portland, Oregon, had a chance to hear Tracy Kidder speak and chat with him at coffee breaks, lunches, and the banquet. (And it's not every day that one gets to sit near, let alone talk to, a Pulitzer Prize-winning author.)

As part of his research for *A Truck Full of Money* Kidder was in contact with Karl Berry, both as someone who knew Paul English well and as someone with connections into the non-commercial part of the software world. Karl invited Kidder to attend TUG 2014, and in return Kidder graciously agreed to give a short presentation — which was primarily about his research and writing of *The Soul of a New Machine* and was highly engaging.

I have read a lot of books, but never before had the opportunity to chat with an author who is researching a book I later read. Having a little connection with Kidder during the writing of this book has led me to think about how books get written, or at least how he writes and structures books.

Kidder's books add context (and information) to the story telling by having some chapters or sections of chapters provide background information on an aspect of the domain of the central character(s). For instance, in his book *House* there is a discussion of wood and the lumber industry.

The inclusion of Knuth in the book is a way to provide background on the world of computer programming. Kidder talks about Knuth's assertion that approximately one in fifty people "have a peculiar way of thinking that makes them resonate with computers" and that computing projects need such "geeks" to succeed. He also recounts Knuth's belief that programming is an art in the sense that programs can be written that are works of art. (I hope the reader will not take from this discussion the idea that programming is an art in the sense that its practitioners don't need learned craft and discipline.) Knuth is also quoted as saying, "our pleasure is significantly enhanced when we accomplish something with limited tools." Kidder uses the span of Knuth's career to note that lots of programming these days is about putting existing pieces of software together instead of writing innovative new programs. Finally, Kidder uses Knuth's parody about an "earth shaking announcement" about iTEX at TUG 2010 in San Francisco to make the point that we now live in an app-oriented computing world. This Knuthian point of view may help readers see computer programmers as having a fascinating life when many people undoubtedly think of computers as highly bothersome ("why can't I make it do what I want", is a question I've heard) and sometimes completely infuriating ("who are the idiots that create these things").

Kidder's TUG 2014 presentation, chats I had with him at the conference and in a few follow-up emails, and what he wrote in *Good Prose* (with Richard Todd, 2013) brought home to me that in writing there are two voyages of discovery: first, discovering information about a topic; second, discovering a tellable story based on aspects of the discovered information. (For more tips on writing, read *Good Prose* or at least look at `tinyurl.com/kidderonwriting`.[2])

As I ponder the content of the book, its writing, and its structure, I wonder to what extent Tracy Kidder saw parallels between his kind of non-fiction writing and the code writing of computer programmers in the book — both involving editing and rewrites, both seeking a popular audience, both highly creative. Did he come at all to identify with the computer programmers he was writing about?

◇ David Walden
`walden-family.com/texland`

[2] Also, here is an interview with Kidder about how he writes: `tinyurl.com/kidder-interview` (part 1), `tinyurl.com/kidder-interview2` (part 2).

TUG 2016 abstracts

Editor's note: Slides and other related information for many of the talks are posted at `http://tug.org/tug2016/program.html`.

— — ∗ — —

Kaveh Bazargan

A graphical user interface for TikZ

TikZ is a powerful vector graphics program. The capabilities are far higher than those of any interactive graphics program, e.g. Adobe Illustrator. The fact that TikZ can be coded logically, keeping the structure of a graphic intact has many advantages, including the generation of "semantic" SVG output, allowing for better accessibility of graphics, e.g. for blind and visually impaired readers. The main barrier against popular usage of TikZ outside the TEX community is that it is purely code driven, and has a steep learning curve.

At River Valley we have a long term program to produce rich, semantic illustrations which are accessible to readers who are blind or visually impaired. We have chosen TikZ as the engine to generate these.

To make easier to create the diagrams in the first place, we have been working on a graphical user interface (GUI) to generate TikZ code and to show its output in near real-time. The first implementation addresses the relabelling of existing illustrations (vector or bitmap) using TikZ. The system has the following advantages: relabelling can be done as fast or faster than conventional methods; labels do not change size as figures are resized (similarly to Pinlabel); labels are saved as text and can be spell-checked along with the main text of a TEX file.

I will give a live demo of the software.

Charles Bigelow

Looking for legibility

Scientific studies of legibility and their contributions to typography and type design from the 19th century to the present. With amusing anecdotes, ingenious contraptions, clueless assumptions, cooked results, Herculean efforts, and occasional progress toward understanding one of our most puzzling cultural activities.

The following bibliography covers most of the content of the talk on legibility studies of the 20th century, especially the first half of the century, but sadly omits the images and the jokes in the talk.

• *From 1900 to 1950*

Émile Javal (1905). *Physiologie de la lecture et de l'écriture.* Félix Alcan, Éditeur, Paris.

Edmund Burke Huey (1908). *The Psychology and Pedagogy of Reading.* Macmillan, New York.

Barbara Roethlein (1912). "The Relative Legibility of Different Faces of Printing Types", *The American Journal of Psychology*, Vol. 23, No. 1, Jan. 1912, pp. 1–36.

British Association for the Advancement of Science (1913). *Report on the Influence of School-Books upon Eyesight.* Offices of the Association, London.

Lucien Alphonse Legros and John Cameron Grant (1916). *Typographical Printing Surfaces.* Longmans, Green, and Co., London.

Lucien Alphonse Legros (1922). "A Note on the Legibility of Printed Matter: prepared for the information of the Committee on Type Faces." H. M. Stationery Office, London.

L. Richard Pyke (1926). "Report on the legibility of print." Medical Research Council, Special report series, no. 110. His Majesty's Stationery Office, London.

Gerrit Willem Ovink (1938). *Legibility, Atmosphere Value and Forms of Printing Types.* A.W. Sijthoff's Uitgeversmaatschappij N.V., Leiden.

Donald G. Paterson and Miles A. Tinker (1940). *How to Make Type Readable.* Harper & Brothers.

Mergenthaler Linotype Company (1941). The Readability of Type. Brooklyn, N.Y.

Matthew Luckiesh and Frank Kendall Moss (1944). *Reading as a visual task.* D. Van Nostrand.

• *After 1950*

Cyril Burt (1959). *A Psychological Study of Typography.* Foreword by Stanley Morison. Cambridge University Press, Cambridge.

Miles Tinker (1963). *Legibility of Print.* Iowa State University Press, Ames.

E. C. Poulton (1965). "Letter differentiation and rate of comprehension in reading," *Journal of Applied Psychology*, Vol. 49, No. 5.

Bror Zachrisson (1965). *Legibility of Printed Text.* Almqvist & Wiksell.

François Richaudeau (1984). *Recherches actuelles sur la lisibilité.* Retz.

Herbert Spencer (1969). *The visible word: problems of legibility.* Lund Humphries/Royal College of Art, London.

Erich Schulz-Anker (1970). "Syntax-Antiqua, a Sans Serif on a New Basis; Le Syntax Romain, un Linéale sur une base nouvelle; Syntax-Antiqua, eine serifenlose Linearschrift auf neuer Basis." Gebrauchsgraphik 7/1970. Reprinted by D. Stempel AG, n.d.

Rolf F. Rehe (1974). *Typography: how to make it most legible.* Design Research International, Carmel, Idaho.

Keith Rayner and Alexander Pollatsek (1989). *Psychology of Reading.* Prentice Hall.

Dirk Wendt (1994). "What do we mean by legibility", in *Font Technology*, by Peter Karow, Springer.

Paul Kolers, Merald Wrolstad, and Herman Bouma, editors (1979). *Processing of Visible Language, Volume 1.* Plenum Press.

Paul Kolers, Merald Wrolstad, and Herman Bouma, editors (1980). *Processing of Visible Language, Volume 2.* Plenum Press.

Ole Lund (1999). "Knowledge construction in typography: the case of legibility research and the legibility of sans serif typeface". Ph.D. thesis, Department of Typography and Graphic Communication, University of Reading.

Gordon E. Legge (2007). *Psychophysics of Reading in Normal and Low Vision.* Lawrence Erlbaum Associates, Mahwah N.J.

Sofie Beier (2009). "Typeface Legibility: Towards defining familiarity". Ph.D. thesis, Royal College of Art.

Robert Bringhurst

The evolution of the Palatino tribe

Palatino is not just a single typeface but a large and varied group of faces: a taxonomic tribe produced over more than half a century. The members include Aldus, Enge Aldus, Aldus Nova, Heraklit, Michelangelo, Phidias, Sistina, and Zapf Renaissance, as well as foundry Palatino, Linotype Palatino, American Export Palatino, Linofilm Palatino, PostScript Palatino, Palatino Nova, and Palatino Sans.

This constellation of type designs was Hermann Zapf's most ambitious and enduring design project. It began with the "Medici" sketches of 1948 — which led to the first trial cutting of foundry Palatino roman by August Rosenberger at the Stempel Foundry, Frankfurt, in 1949 — and it continued through 2006, when the last authentic members of the group were drawn on screen under Zapf's direction by Akira Kobayashi in Bad Homburg. In between these dates, the underlying designs adapted again and again to changing conditions, represented by the Linotype machine, Linofilm and other phototype machines, and a variety of pre-PostScript digital systems.

Zapf was not the only type designer whose career spanned the tumultuous transitions from metal type to phototype to digital type, but Palatino and its relatives appear to be unique in the complexity of their evolution and the multiplicity of their successive adaptations, under the hand of the original designer, to repeatedly changing methods of typesetting and printing.

Robert Bringhurst has argued for many years that the most promising system of typeface classification is based on botanical and zoological taxonomies. His new book, *Palatino: The Natural History of a Typeface*, published in a limited edition by the Book Club of California, with a trade edition from David R. Godine coming this fall, is an extended test of this thesis. Over many years of research, he has also accumulated hundreds of illustrations documenting the artistry and care, and the industrial advances and collapses, involved in creating these components of our typographic heritage.

Jennifer Claudio

The case for justified text

Documents must be aesthetically pleasing without appearing deliberately designed. One of the basic functions built into most word processing software is text justification. The algorithms behind this function, however, vary based on the software and the preset rules within the software. Some criticisms include compromised readability. Defenders of justified text argue that as long as the typeface is appropriately sized and kerned, justification does not detract from readability. This presentation succinctly demonstrates the behavioral differences visible in WordPerfect, Word, InDesign, and LaTeX, and examines the ability for people who read common printed media to notice the differences.

Jennifer Claudio

A brief reflection on TeX and end-user needs

TeX attracts users who seek a robust method of creating precise typographic products. However, beginning with the instinct to disambiguate the pronunciation of TeX and LaTeX, new users often feel daunted by the so-called learning curve of TeX and its relatives. Given time to learn the syntax and experience in troubleshooting errors, many fare well; however, a population exists that would benefit from the use of TeX who have insufficient time or comfort in the field.

This presentation describes the following: 1) personal use of products that have been shared by other members of this TeX user group community; 2) the quest to recruit new users, abusers, and developers into the TeX community; and 3) requests for specific end-user products.

Tim Inkster

The beginning of my career

Tim Inkster was one of any number of English undergraduates at the University of Toronto in the late 1960s who were entranced by the lure of Stan Bevington's shop in the alley behind 401 Huron Street.

Inkster applied for an entry-level position at Coach House Press, the first time, in 1969, and then a second time a couple of years later.

Unable to secure gainful employment, Inkster felt he had little choice but to start his own small press, the Porcupine's Quill (1974), which has led to bronze and silver medals at Leipzig, a Citation from the Art Directors' Club of New York, and the Order of Canada (2009) for both Tim and his wife, Elke, "For their distinctive contributions to publishing in Canada and for their promotion of new authors, as co-founders of the Porcupine's Quill, a small press known for the award-winning beauty and quality of its books."

Stefan Kottwitz
TEX in industry I — programming Cisco switches using TEX
I report on the pure text-based, macro-based, TEX-programmed switch configurations which I use at my work at Lufthansa in cruise ship projects. A brief description is available at `http://tex.tips/programming-network-switches`. A demonstration of a non-standard use of TEX in industry.

Stefan Kottwitz
TEX in industry II — designing converged network solutions
Pure graphics, with efficient use of TEX and TikZ for programming drawings of network architectures (LAN, wi-fi, VoIP, ...). "Efficient" in the sense of my note on good practices at `tex.stackexchange.com/a/297029/213`. Sample notes are at `tex.tips/tag/industry`, and example drawings at `tex.tips/LAN-1-2.pdf`.

Presenting a showcase of what can be done with TEX (instead of Visio, PowerPoint or CAD) and how, with a view toward efficiency (time pressure on projects).

Kevin Larson
Reading between the lines: Improving comprehension for students
While reading is arguably a student's most important skill, the technology of reading is relatively unchanged. Can the power of computing improve a student's reading comprehension? We will discuss what has been learned about typography in the last 500 years, about reading psychology in the last 100 years, and what technology can be invented right now.

Bio: Kevin Larson works for Microsoft's Advanced Reading Technologies team. He collaborates with type designers, reading psychologists, and engineers on improving the onscreen reading experience. Kevin received his PhD for studies of reading acquisition.

Frank Mittelbach
Alice in Wonderland — The tale of the long tail in globally optimized page breaking (part 2)
The story of Alice's Adventures In Wonderland by Lewis Carroll contains a long tale about the tail of the mouse, which starts with

```
   'Fury said to a
      mouse, That he
   met in the
  house,
 "Let us
  both go to
   law:  I will
    prosecute
     YOU.---Come,
       I'll take no
       denial; We
     must have a
    trial:  For
  really this
 morning I've
nothing
to do."
 Said the
  mouse to the
   cur, "Such
    a trial,
     dear Sir,
    ...
   many more
  lines ...
 ...
```

Obviously a tail like this should be typeset in full beauty and should not be laid out in knots or cut between pages. Unfortunately, that is more easily said than done, given that all typesetting systems use a greedy page algorithm that cuts page by page. Thus chances are high that disaster strikes and we have to manually adjust earlier page breaks to prevent it.

Using global optimization in pagination has been envisioned already more than 30 years ago by Michael Plass in his PhD thesis and throughout the years other people worked on specific aspects of global optimized pagination, but until today all typesetting engines have taken the "easy" way out and leave the problem essentially to the user.

This is in fact not that surprising, as the problem can get easily out of hand: A naive approach will immediately result in exponential complexity (without introducing float objects into the mix which makes it worse). But even with careful restrictions and specialized algorithms we will soon be reaching the limits of modern hardware with any real live document.

However, computers are getting faster and thus get us closer to make globally optimized pagination a reality. So this year I started to implement a framework that assists in this task, in the hope that for my next book I do not have to hand-adjust half of the page breaks manually — first results are promising!

At Bachotek 2016 I gave some theoretical background to the problem and discussed the basic approach taken by the framework, including two already implemented optimization strategies: The automatic change of page length on double spreads to add flexibility and the use of automatic variation in paragraph breaking (think `\looseness`) to gain further flexibility.

This time around I like to demonstrate new results where Alice goes “floating” and all the beautiful pictures of the orginal magically appear in their appropriate place.

So slowly the work evolves toward a usable solution. Sit back, relax, and enjoy.

Norbert Preining
Security improvements in the TEX Live Manager and installer
Since the switch to the current distribution method and the introduction of network installs and updates, some years ago, many things have changed in the TEX (Live) world. But one thing has not kept up with the new distribution methods: security.

Until now, there has been almost no verification of a package as downloaded from the CTAN mirrors compared to the original package created in TL. Although we have been shipping MD5 checksums and sizes in the accompanying information, these were used only in rare instances (namely, when restarting a failed installation).

We report about the recent improvements and consistent confirmation of checksums and sizes of the downloaded packages, as well as improvements regarding strong cryptographic signatures of the package information.

Arthur Reutenauer, Mojca Miklavec
The TEX Live M sub-project
TEX Live is the most versatile of TEX distributions, available on a variety of platforms, and very actively developed. It is the basis for MacTEX on Mac OS X, and is bundled by many package managers in Unix distributions. It has, however, a major drawback: its titanic size. This talk will discuss a sub-project to address that, with time for general discussion on wishes and ideas for TEX Live’s future.

At its inception in 1996, it was contained in a CD and started growing immediately. Packages are rarely removed, due to compatibility considerations, and only technical considerations are taken into account when considering new packages: if a package fits the requirements, it is added. Today, the `texlive-full` installation scheme includes over 140,000 files and has an installed size of over 4.5 GB.

This situation is a problem for many downstream package developers and also affects the TEX community as a whole. We have started a conversation to see how we could help users find packages. We would like to offer an option to have a more controlled set of packages, probably by creating a new TEX Live “scheme” in the existing distribution by selecting among the 3200+ (to date) packages. We could define strict dependencies between packages, and also strive to do some measure of quality control, in order to create a distribution that’s truly useful for newcomers and long-time users alike. The selection has to be community-driven, but there has to be a selection.

In another area, we also want to improve how the binaries are built: at the moment, they’re compiled once per year by a number of volunteers who work on one or more of the twenty or so different platforms, and never get updated during the year. While this strikes a good balance between stability, the demand for reasonably recent binaries, and the workload of volunteer builders and packagers, we thought we could do better.

We have recently set up a build infrastructure that can automatically build TEX binaries after every source change for a number of platforms, send emails when builds break, show reports, and make the binaries available to users. This approach takes a lot of burden off the shoulders of people previously responsible for building TEX binaries, while at the same time giving us freedom to run the builds a lot more frequently, getting binaries to users much faster and providing earlier feedback about problems to developers. This part is almost ready and we will give some technical details of how it works.

TUG 2016, Bond Place Hotel, Toronto Annual General Meeting Minutes 25 July 2016, 4 p.m.

Sue DeMeritt, Secretary

The meeting was called to order at 4:15 p.m. by Jim Hefferon.

Jim introduced the members of the Board of Directors who were present. He then presented slides giving a snapshot of TUG. These included reasons why people should join TUG and reasons why TUG is so important to the community. Financial matters were reviewed very generally. Challenges to TUG were discussed. The biggest challenge is membership, which is declining. This led to a discussion of dues, with a slide showing changes since 2006. [The slides are available at `http://tug.org/tug2016/slides/agm.pdf`.]

Jim then invited Kaveh Bazargan to speak, per Kaveh's request.

Kaveh Bazargan made a statement announcing his resignation as President of the TEX Users Group. In 2015 he had been elected president; however, he was subsequently suspended by the Board. He noted that if his suspension were lifted, he would be faced with a Board that he would not be able to work with.

Jim pointed out that there is an upcoming election in 2017 when TUG will elect a new president and board members.

There were no immediate comments from the membership.

The floor was opened for questions and discussion.

CV Rajagopal stood up and started to discuss how his company's reputation has been marred as a result of a lawsuit by Kaveh Bazargan. Norbert Preining and Jim Hefferon asked CVR not to bring private legal matters into this meeting. CVR continued and asked the TUG Board if they agreed with the statements that Kaveh had circulated earlier in the day. Jim Hefferon stated that the Board does not wish to be involved in the parties' lawsuits, but noted that the Board does not agree with all of the points in the letter that had been circulated. [The letter will be published verbatim in the TUG 2016 proceedings issue of *TUGboat*.]

Matthew Skala started a discussion about membership: What is needed to increase membership? The number of members attending the annual conference is mostly the same every year.

Mike Sofka stated that he feels membership has dropped because so much is now free on the web. Getting younger people involved is crucial.

David Walden mentioned that many other groups are facing the same issues. People don't want to join a group with only paper journals.

Michael Doob pointed out that younger people are just not joining. It is hard to know what the answer is. Having personal contact with a possible member seems to help.

Kim Nesbitt discussed why it is important to get information out about why people should join TUG.

Jim Hefferon noted that TUG has been trying many different avenues for attracting more members.

Paulo Ney de Souza said that he has no answer to membership issues but does have a suggestion for the resultant financial problems. TUG has a number of institutional members, but many libraries aren't allowed to use funds for institutional memberships. It would be better to make *TUGboat* available as an electronic-only journal to institutions for a larger fee. [Not mentioned at the meeting is the fact that non-member subscriptions to *TUGboat* have been available for years.]

Christina Thiele pointed out that TUG originally had a large membership because there was no web. Today's people are digital. Perhaps TUG could try to get donations digitally from folks in small amounts. If we come in from a different direction it might be successful in finding more members.

Christian Gagné asked if the number of members includes institutional members. [The institutions are not included, but the individuals named by their institutions *are* included.] He would like to see the TUG website used as a portal. Jim Hefferon discussed what ideas are being pursued. Barbara Beeton reported that AMS has asked members what kinds of programs they would like to see; most of the topics mentioned were already in existence, but their availability was not generally known.

Mike Sofka discussed that TUG does not provide all the information that people would like to have. They have to have a reason to join.

Frank Mittelbach pointed out that the original reason for TUG was to get information to users; TUG was the only place where information was available, such as finding support for TEX infrastructure. Asking for support for something that is not obvious is difficult. He supports the idea of donations as being a feasible theme. He also agrees with Paulo that libraries are usually not permitted to use funds to join as institutional members.

Behdad Esfahbod reported that the reason he originally left was because he did not feel he was getting anything for his membership. Jim Hefferon mentioned the electronic access.

Jennifer Claudio suggested a social media approach where people can get what they want online. Perhaps scholarship programs can get people involved. "Lifetime" membership builds commitment. It would give more incentive to keep coming to TUG, spreading info to friends.

John Plaice suggested that there was an elephant sitting in the room. The president was overthrown although a majority voted for him; the Board collectively overthrew the president. Why would people want to join if a coup had taken place?

Federico Garcia asked about the TEX user population and the funds available to TUG. This can be a matter of marketing; people don't know about TUG, and we need to get the word out.

John Kerr suggested advertising in technical magazines. Readers of these magazines may not know that TUG exists.

Dave Walden stated that someone suggested to him that a TUG blog would be a good idea. This is a matter of finding a volunteer.

Matthew Skala proposed that maybe TUG is not giving people what they want. It is not what the organization needs — it is what supporting the product needs. Frank Mittelbach added that people may feel that the dues are too much for what they are getting. Maybe a new model is needed.

Christina Thiele added that a blog is a good idea; users can write about what they are doing so others can see it. Other users will read and spread the word.

The meeting adjourned at 5:08 p.m.

Report: Suspension of Kaveh Bazargan as TUG President

TUG Board of Directors

Until the completion of all steps specified in its bylaws, the TUG Board has, on legal advice, released to the membership only limited information regarding the process by which Kaveh Bazargan was suspended as TUG President. The following is the Board's report on the matter.

Summary

Before filing as a candidate for the TUG presidency, Kaveh Bazargan had instituted a legal suit against another TUG member (not on the Board). He did not inform the Board or the elections committee of such litigation. After the announcement of the election results was posted on the TUG website, Kaveh submitted that document to the court in support of his suit, again not informing the Board or the other candidates, whose information was included in the announcement. (The results of the election were announced on 23 May 2015, and the news about their inclusion in the court papers arrived on 21 August.)

When asked by the Board to either withdraw the announcement from the court records, or submit a notice stating that the TUG Board had requested that it be withdrawn, Kaveh did not acknowledge this request or take any steps to act on it after repeated attempts by the Board to obtain a definitive response.

This action does not, in the Board's opinion, demonstrate the duty of loyalty to the organization, in that Kaveh was holding his own interests above those of the organization.

It was on these grounds that the TUG Board acted to suspend Kaveh as TUG President.

Since the creation of TUG, the TUG Board has consistently held to the principle that Board members should be free from conflict of interest. This principle was perhaps best expressed by Pierre MacKay as part of his valedictory comments on stepping down as TUG President, in *TUGboat* 6:3, page 114, "Statement of Principles" [`http://tug.org/TUGboat/tb06-3/tb13gendel.pdf`]:

> To avoid any real or apparent conflict of interest, all members of the TUG Steering Committee undertake that they shall make no use of their position on that committee for personal advancement and shall make no private use of information acquired by the Steering Committee unless and until such information has been published to the general membership of TUG.

This principle is consistent with the legal underpinnings of TUG's incorporation under the Rhode Island Non-Profit Corporation Act (RINCA) [`http://webserver.rilin.state.ri.us/Statutes/TITLE7/7-6/INDEX.HTM`].

Section 7-6-34.(4)(i) of RINCA states [`http://webserver.rilin.state.ri.us/Statutes/TITLE7/7-6/7-6-34.HTM`]:

> (4)(i) Any provisions, not inconsistent with the law, which the incorporators elect to set forth in the articles of incorporation for the regulation of the internal affairs of the corporation, including a provision eliminating or limiting the personal liability of a director to the corporation or to its members for monetary damages for breach of the director's duty as a director. However, the provision does not eliminate or limit the liability of a director:
>
> (A) For any breach of the director's duty or loyalty to the corporation or its members;

Loyalty is well defined in corporate law as the duty of an individual to hold the interests of the organization above personal interests.

Kaveh Bazargan submitted papers for his candidacy for TUG President in January 2015. At that time he was already involved in a legal dispute in India, which began no later than 2014, with another TUG member (not on the Board); however, this was not made known to either the TUG Board or to the Board members charged with setting up the election. On 21 August 2015, the TUG member who was the target of the suit sent a message to the Board containing the information that the TUG election announcement [`http://tug.org/election/2015/candidates.html`] had been presented by Kaveh to the court as part of the documents supporting his (Kaveh's) case. The relevant part of this message reads:

> 4. Kaveh has used the results of the recent TUG elections as well as the names and statements of other members of the board to back his claims by including these as part of the documents filed in court on behalf of the Plaintiff, which is himself.
>
> Is this being done with the concurrence of the TUG Board?

This was indeed done without Kaveh having notified the Board, or asking the other individuals involved in the election for permission to place their information and pictures in court filings in a legal suit; although the election announcement had been publicly posted on the TUG website, Kaveh should have notified the Board of this action, and requested permission.

The Board acknowledged receipt of this message to its sender, but had no other correspondence with that TUG member on the subject.

We note that use of the position of TUG President for personal benefit would clearly be a conflict of interest. However, we do not know that this was stated directly to the court, only that the election announcement was submitted as part of Kaveh's evidence.

The question asking whether the notice had been submitted with the concurrence of the TUG Board was forwarded in a message to Kaveh, on 27 August 2015, along with the following request from the Board:

> The answer to this is "no". None of the candidates whose material was submitted had any knowledge of this action until well after the documents were submitted, and, had they been asked for permission, their responses would have been "no". It is our request to you that the election statements be withdrawn from the court filings, if possible; if not possible, a note should be added that the TUG Board has made such a request for removal.

This direct request was never acknowledged by Kaveh; instead, his responses skirted the issue. On 28 August he replied:

> It's suddenly becoming clear to me!! There has been a misunderstanding. Allow me to explain. [...]
>
> Let me reiterate: TUG and TUG Board members are *not* involved in this case. They are not accused of anything and not endorsing anyone. My only intention is to prove I have been elected as the president of TUG. How else could I have done that?
>
> [...]

The Board did not accept this as a reasonable explanation, and thus sent the following note on 3 September:

> Kaveh - in your response, you neither affirmed nor denied our request (copied below). Please respond directly. Thanks.
>
> [forwarded copy of the full message dated 27 August]

On 16 September, the following request was received by the Board from Kaveh:

> I really want to put the matter of the 'grievances' to rest so that I can start contributing to TUG without distraction. I replied to your concern again last week. Please confirm that the grievance issue is now closed. In case the board believes it is not, then in the interests of TUG, I look forward to receiving the precise points as soon as possible so I can address them and we can all get on with our main task, namely making TUG even better than it is.

This was clear evidence that a serious disagreement existed, with no recognition by Kaveh that, in the Board's opinion, a line had been crossed. The Board waited, without communication with Kaveh regarding the matter, for a direct reply, which was never forthcoming. Board members discussed the matter in private email, and on 17 September the following notice (included here in full) was sent to Kaveh:

Date: Thu, 17 Sep 2015 19:30:45 -0400
From: TUG Board

Dear Kaveh,

Over the last several weeks the Board of Directors has deliberated intensely over the current situation. Here is our consensus opinion.

A lawsuit between TUG members concerning TeX-related activities is in itself a very unfortunate matter for the user group. Any such involvement by a TUG officer compromises TUG's standing in the community. As you know, the TeX Users Group is not a party in this suit. We cannot even give the appearance of having taken sides. The situation of the position of TUG President having been used for private matters in any manner is unacceptable to the Board. It is important for the community to keep their trust in TUG as an impartial organization representing all its members.

Thus, we think that your involvement in a lawsuit with another TUG member, while you are TUG president, concerning TeX-related activities is a conflict of interest. Recent events have shown that this conflict cannot be mitigated.

We do not see any way to resolve this situation while you are TUG President. Thus we think it would be in the best interests of TUG for you to take a leave of absence until the lawsuit and all related legal matters are settled, or to resign.

Please understand that this recommendation does not imply any judgement about you and your skills. We recognize the valuable contributions that you have made to TUG over the years. The Board is concerned only with the welfare of TUG and what it represents as an organization.

This decision has been very difficult for us all, and we hope you understand that we think this is the best route for the benefit of TUG.

TUG Board of Directors

As of 24 September, no response had been received from Kaveh, and this reminder was sent:

> Kaveh — we sent our message a week ago. Can you please provide an ETA as to when you will reply? Thanks.

Another reminder was sent on 28 September, with this warning:

> We would like to hear from you before this Thursday, October 1. Otherwise we will need to consider possible next steps, including those in Article IV section 5 of the TUG bylaws.

Kaveh responded the same day, but made no acknowledgment of the Board's initial request regarding withdrawal of the election notice from the court documents. A motion for suspension was proposed, subject to a more congenial resolution. The next message from the Board to Kaveh was sent on 6 October:

Date: Tue, 6 Oct 2015 18:59:58 -0400
From: TUG Board

Kaveh,

The conflict is as already stated: as TUG president, you have a duty to represent all TUG members to the best of your ability (just as we do as TUG directors). It is not possible to fulfill this responsibility when you are involved in a lawsuit against another TUG member. Any decision made or initiative undertook by a TUG president while pursuing a lawsuit against another TUG member would, at the very least, appear to be tainted.

From your messages, apparently you do not agree that this is a problem. Nevertheless, after lengthy and careful deliberation, including taking your responses into account, the required majority of the board has concluded that your presidency must be suspended because of this.

We think it would be better both for TUG and for you if this outcome was announced as your decision. If you agree, we could announce that after a discussion with the board you generously decided to step down to avoid the distraction of a pending lawsuit from interfering with TUG business.

However, if you disagree, we will publish this decision as ours after Thursday, October 8.

Sincerely,
TUG directors

On 10 October, Board member Steve Grathwohl had a Skype conversation with Kaveh, urging him to voluntarily step aside. Kaveh refused. (Originally, two other Board members had agreed to participate in the Skype call, but at the only time Kaveh was available, they were not.) The Board voted without dissent for suspension, concluding that further discussion could not lead to a less severe outcome.

On 13 October, the Board wrote to Kaveh:

> [A]s we previously wrote to you, the required majority of the board has concluded that your presidency must be suspended because of the conflict of interest that we see, due to your pending legal actions. Since our attempts at mitigation have not been successful, the suspension is now effective. We greatly regret this outcome.

The Board, on the same day, also notified all members by email:

> [...] We believe that TUG should not take sides, or even appear to take sides, in a lawsuit to which it is not a party. [...]

> Thus, we asked Kaveh to voluntarily suspend his presidency for the duration of the lawsuit and any related legal matters. We were not successful in convincing him that this would be best for TUG. Further, he neither made an explanation as to why he did not reveal the existence of the lawsuit at the time of the election, nor made any offer to mitigate its effects now. [...]

The suspension became effective with this notification, 13 October 2015.

After several messages from Kaveh to the Board requesting an explanation, this message was sent on 21 October:

Date: Wed, 21 Oct 2015 17:14:44 -0400
From: TUG Board

Kaveh,

Since originally becoming aware of the issue, over several emails to you the Board has communicated, in detail, its concerns about the conflict of interest posed by having the TUG President embroiled in a lawsuit with another TUG member. The Board also expressed, very early on, its view regarding the difficulty of TUG being seen as impartial when the member statements for the election are entered as part of court documents supporting one side or the other. You did not accept those concerns.

We also conveyed to you, in writing, that we felt a voluntary resignation or leave of absence, initiated by you, would be the best course of action for TUG. When that effort also failed, a phone call was sought. This call was extended as a courtesy to you. Unfortunately for all of us, this didn't work in the way we hoped for.

Sincerely,
TUG directors

According to Bylaws Section IV.5, a suspended Director "shall have an automatic right of appeal, which must be exercised within 60 days of delivery of notification of suspension." (Sixty days from 13 October is 12 December.) If the appeal is rejected, the suspended Director has the right to appeal to the members in the Annual General Meeting.

Kaveh sent a letter of appeal by email on 10 December. Owing to the size of attachments to the message, it was delayed for several days; however, the original timestamp was accepted as the effective date.

As with previous communications, the appeal did not acknowledge the Board's initial request of 27 August; it also implied that the suspension was invalid. This is the final item in the letter:

> [...] I trust and hope that legal proceedings between us can be avoided by the Board rescinding the purported notice and reinstating me as President of TUG with immediate effect. The Board should be aware that if this does not occur the only conclusion that I can draw is that there is a Board agenda to damage my interests. Accordingly, in that scenario, I will have no option but bring claims for defamation, damages for loss of reputation, breach of statutory duty and tortious interference. The Board is reminded that its members are personally liable for any expenses incurred in connection with the defence or reasonable settlement of any action to which a person is made party by reason of being a director by virtue of Article 10 of the Bylaws

> and, like every member placed in this situation, I shall have no option but to defend my business interests.

In light of Kaveh's appeal letter, the Board believed it prudent to engage legal advice.

With the advice of counsel, the Board concluded that the appeal did not contain any substantive new information, and unanimously affirmed its prior vote. The affirmation of suspension was conveyed to Kaveh by TUG's lawyer on 10 February 2016. A notice to all members was sent on 17 February.

With the suspension, Jim Hefferon, as Vice President, assumed the role of acting President, effective until the issue was resolved.

In accordance with Section IV.5 of the Bylaws, a suspended Board member is provided the right of final appeal at the next Annual General Meeting (AGM). The onus is on the suspended member to register that appeal so that it can be included on the agenda for the AGM. When no such appeal request was received, on 8 July, as a courtesy, the TUG lawyer sent a message to Kaveh asking whether he intended to appeal, and requesting a response no later than 15 July. This date was chosen as it was the last date on which a notice could be sent to members announcing the business to be taken up at the AGM, scheduled for 26 July at 4:15 p.m.; Section III.5 of the Bylaws requires that notice of a meeting be sent no less than ten days before the meeting.

On 15 July, a reply was received from Kaveh, stating "This is to let you know that I have not yet decided on the matter."

On 15 July, a notice was sent to all members announcing the date and time of the AGM, with (in the absence of a decision from Kaveh) the stated purpose "to discuss normal business, including but not limited to, developing and implementing strategies designed to increase TUG membership." This notice was not required, as the date and time of the meeting had already been posted for several weeks as part of the conference schedule on the TUG website, but the formality was observed in deference to agitation on public non-TUG websites and other TEX forums.

At the conference early on the day of the AGM (26 July), Kaveh distributed a document entitled "Recent events in TUG" in which he set forth his point of view, and announced his resignation as President of TUG. He requested, and was granted, time at the AGM to make a brief personal statement. In this statement, he reiterated his resignation, with the reason being that even if he were reinstated, he would be faced with a Board that he could not work with.

Had the suspension come to a vote by the assembled TUG members, the two possibilities would have been: to uphold the suspension or to reinstate. Kaveh's decision to avoid the vote and leave the Board through resignation was recognized and accepted.

No mention is made in Kaveh's document "Recent events in TUG" regarding withdrawal of threats to sue the Board as a whole or its members as individuals. Therefore, the Board still believes that this is an active possibility, and must conduct the business of TUG accordingly.

The document distributed by Kaveh at the AGM is reproduced in full from the original, following this report. The Board does not agree with the points made in Kaveh's statement in the section "The TUG Board", but believes that every member is entitled to reach their own conclusions.

Recent events in TUG

My background with TeX

I started using TeX in April 1983, just in time to write my PhD dissertation at Imperial College. I am officially the first TeX user at Imperial, and possibly first in UK. I attended my first TUG meeting exactly 30 years ago in Strasbourg. From the first time I used TeX I fell in love with it and my love has not diminished.

I started my business in 1988 in the UK, with TeX at the centre of the operation. After years of hard work the business is now established, with multiple offices, and major publishers as clients. TeX remains at the core of my business.

Giving back

I feel a strong sense of gratitude to TeX and friends, their authors and the TeX community that shares my commitment to this marvellous technology. In recent years I have done what I could to "give back" to the TeX community, including presenting at TUG conferences, hosting two TUG meetings and sponsoring several, and recording no less than 14 TeX conferences, including the Toronto conference, free of charge. Last year, a few days before the 2015 TUG board elections, I had an epiphany – why not run for the presidency of TUG? I was aware of many opportunities for developing a stronger awareness of TeX in the publishing industry, and attracting funding for TeX projects and for this Group. What better way to give back than to use my influence in the publishing industry to raise the profile of TUG to where it deserves to be?

The law suit

In 2014 I started proceeding that asked a court in India to resolve a dispute with a former business associate, over ownership of proprietary software. The details are probably not of interest to TUG members, but they are not a secret and I am happy to give more details if people are really interested. The dispute is unconnected with TUG.

Election as President

I was nominated by TUG board member, Arthur Reuternauer, and threw my name in the ring just before the nomination deadline, with a clear agenda for change. When I was elected President by 307 votes to 110, I immediately made plans to deliver what I had promised.

The TUG Board

The members of TUG already know that the Board suspended me as President because of the lawsuit in India. Here are some facts that trouble and concern me about the Board's conduct.

1. They did not tell the TUG members that the other party in the lawsuit asked the Board to remove me as President.
2. They did not get legal advice before acting, and they have not published the legal advice TUG has since received.
3. The Board knew, when they suspended me, that I was addressing their concern about

TUG references in court papers. Their concern was that I had submitted as evidence the public TUG website page that announced the result of the 2015 election.

4. They did not follow the TUG bylaws regarding conducting meetings by email.
5. There are no minutes available relating to the suspension.
6. The principal ground for the suspension was the fact that the other party in the dispute over ownership happened to be a member of TUG.
7. In their response to my appeal, the previous principal ground stated at the outset was replaced by a lesser matter, namely concerns about court papers.
8. The Board's pressure on me to change my court evidence amounts to taking sides in the lawsuit, and may be legally improper.

Resignation

Earlier today, I formally resigned as President of TUG.

I have asked the Board to allow me to make a brief personal statement explaining why at the TUG Annual Meeting, which is at 4.15pm today.

Kaveh Bazargan
Former President of TUG
26 July 2016

Response to Kaveh Bazargan's message

While Kaveh Bazargan certainly has the right to insist on his interpretation of the recent events, we feel that some statements in his letter are not accurate and deserve to be explained from the Board's point of view.

The numbers below correspond to the numbers in his text.

1. It is correct that the Board did not tell the membership about the request to remove Kaveh as a President. What is not correct is the implication that this removal was done as a consequence of the other party's request.

   We considered that request improper and did not discuss it further. Our aim was to avoid any involvement of TUG in the lawsuit and any appearance of such involvement. Only the repeated refusal of Kaveh to do anything towards this aim led us to the difficult decision to suspend his presidency.
2. It is correct that the Board sought legal advice only after the removal was done. What is not correct is the implication that such legal advice is routine in TUG business.

   The decision to hire a lawyer was unprecedented, except for handling TUG's incorporation and application for non-profit status, over the several decades of TUG's existence. The Board took this step only after receiving an explicit threat of lawsuits against the Board as a whole *and* against its individual directors. While we felt a natural aversion to spend TUG funds on lawyers, we felt it was necessary for the organization itself.

   In light of this threat, Kaveh's request to make the legal advice received by the Board public is nonsensical. The lawyer who has been advising TUG has told us that communications between his firm and the TUG Board should not be disclosed as they are subject to the attorney-client privilege, which we will honor.
3. It is incorrect that, at the moment of suspension, Kaveh "was addressing their concern about TUG references in court papers". In fact our repeated requests to delete, remove, and/or file a notice regarding the TUG court references were not met with any understanding at all. We strongly felt at that time — and strongly feel now — that this failure to act violates the duty of loyalty/conflict of interest obligation that Kaveh owed.
4. The statement that our decision did not follow TUG bylaws has no grounds whatsoever and is incorrect.
5. It is correct that there are no minutes for this decision. What is incorrect is the implication that keeping minutes is either required by our bylaws or the law in general — or that it is customary for TUG except for formal in-person meetings and at the AGM.

   When TUG became an international organization with a diverse Board, such in-person meetings became rare. Section 9 was thus added to the TUG Bylaws which allows Directors' consent voting by e-mail following e-mail discussion. However, there is no requirement to keep formal minutes of such Board discussions, and this has never been done.
6. It is incorrect that "[t]he principal ground for the suspension was the fact that the other party in the dispute over ownership happened to be a member of TUG." The grounds of the suspension were the conflict of interest, the failure to disclose it and the failure to eliminate this conflict of interest and involvement of TUG in the lawsuit — which demonstrated the lack of loyalty to the organization.
7. It is incorrect that the grounds for removal were changed between the suspension and appeal. The grounds, listed above, were the same.
8. It is incorrect that the Board's request to eliminate the involvement of TUG in papers supporting a lawsuit amounts to taking sides in the lawsuit. We are emphatically *not* taking sides there. Moreover, Kaveh's statement suggests that the TUG connection is important evidence in the lawsuit, and confirms the existence of the conflict of interest and impropriety of the situation.

2017 TEX Users Group election

Barbara Beeton
for the Elections Committee

The positions of TUG President and nine members of the Board of Directors will be open as of the 2017 Annual Meeting, which will be held in April–May 2017 in Bachotek, Poland.

The terms of these individuals will expire in 2017: Karl Berry, Kaja Christiansen, Steve Grathwohl, Jim Hefferon, Klaus Höppner, Steve Peter, Geoffrey Poore, Arthur Reutenauer, Michael Sofka.

Continuing directors, with terms ending in 2019: Barbara Beeton, Susan DeMeritt, Michael Doob, Cheryl Ponchin, Norbert Preining, Boris Veytsman.

The election to choose the new President and Board members will be held in early Spring of 2017. 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 ... ballot of the entire membership, carried out in accordance with those same Procedures."

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; the petition and all signatures must be received by the deadline published below. A candidate's membership dues for 2017 must be paid before the nomination deadline. The term of President is two years, and 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 complete nomination forms and ballot information is **5 p.m. (PST) 1 February 2017** at the TUG office in Portland, Oregon, USA. No exceptions will be made. Forms may be submitted by fax, or scanned and submitted by email to `office@tug.org`; receipt will be confirmed by email.

Information for obtaining ballot forms from the TUG website will be distributed by email to all members within 21 days after the close of nominations. It will be possible to vote electronically. Members preferring to receive a paper ballot may make arrangements by notifying the TUG office; see address on the form. Marked ballots must be received by the date 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 mid-April, and will be announced in a future issue of *TUGboat* and through various TEX-related electronic media.

2017 TUG Election — Nomination Form

Only TUG members whose dues have been paid for 2017 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 2017 Annual Meeting, **April–May 2017**.

1. ______________________
(please print)

______________________ ____________
(signature) (date)

2. ______________________
(please print)

______________________ ____________
(signature) (date)

Return this nomination form to the TUG office via postal mail, fax, or scanned and sent by email. Nomination forms and all required supplementary material (photograph, biography and personal statement for inclusion on the ballot) must be received at the TUG office in Portland, Oregon, USA, no later than **5 p.m. (PST) 1 February 2017**.[1] It is the responsibility of the candidate to ensure that this deadline is met. Under no circumstances will late or incomplete applications be accepted.

☐ nomination form
☐ photograph
☐ biography/personal statement

TEX Users Group
Nominations for 2017 Election
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

(email: `office@tug.org`; fax: +1 815 301-3568)

[1] Supplementary material may be sent separately from the form, and supporting signatures need not all appear on the same form.

TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits: `http://tug.org/instmem.html`
Thanks to all members for their support!

American Mathematical Society, *Providence*, *Rhode Island*

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

Center for Computing Sciences, *Bowie*, *Maryland*

CSTUG, *Praha*, *Czech Republic*

Fermilab, *Batavia*, *Illinois*

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

Maluhy & Co., *São Paulo*, *Brazil*

Marquette University, *Milwaukee*, *Wisconsin*

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

MOSEK ApS, *Copenhagen*, *Denmark*

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

Overleaf, *London*, *UK*

River Valley Technologies, *Trivandrum*, *India*

ShareLaTeX, *United Kingdom*

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*

TNQ, *Chennai*, *India*

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

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

University of Cambridge, Centre for Mathematical Sciences, *Cambridge*, *United Kingdom*

University of Ontario, Institute of Technology, *Oshawa*, *Ontario*, *Canada*

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

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 `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`
We are your macro specialists for TeX or LaTeX fine typography specs beyond those of the average LaTeX macro package. If you use XeTeX, 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.

de Bari, Onofrio and Dominici, Massimiliano
Email: `info (at) typotexnica.it`
Web: `http://www.typotexnica.it`
Our skills: layout of books, journals, articles; creation of LaTeX classes and packages; graphic design; conversion between different formats of documents.

We offer our services (related to publishing in Mathematics, Physics and Humanities) for documents in Italian, English, or French. Let us know the work plan and details; we will find a customized solution. Please check our website and/or send us email for further details.

Latchman, David
4113 Planz Road Apt. C
Bakersfield, CA 93309-5935
+1 518-951-8786
Email: `david.latchman (at) texnical-designs.com`
Web: `http://www.texnical-designs.com`
LaTeX consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized packages to meet your needs.

Call or email to discuss your project or visit my website for further details.

Peter, Steve
+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.

Sievers, Martin
Im Alten Garten 5
54296 Trier, Germany
+49 651 4936567-0
Email: `info (at) schoenerpublizieren.com`
Web: `http://www.schoenerpublizieren.com`
As a mathematician with more than 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 nineteen years of experience in TeX and three decades 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.

Webley, Jonathan
21 West Kilbride Road
Dalry, North Ayrshire, KA24 5DZ, UK
01294538225
Email: `jonathan.webley (at) gmail.com`
I specialize in math, physics and IT. However, I'm comfortable with most other science, engineering and technical material and I'm willing to undertake most LaTeX work. I'm good with equations and tricky tables. I can also proofread and copy-edit if required. I've done hundreds of papers for journals over the years. Samples of work can be supplied on request.

Calendar

2016

Sep 25 – Oct 1	10th International ConTeXt Meeting, "Piece of Cake", Kalenberg, The Netherlands. `meeting.contextgarden.net/2016`
Sep 29	*TUGboat* **37**:3, submission deadline.
Sep 30 – Oct 2	Oak Knoll Fest XIX, New Castle, Delaware. `www.oakknoll.com/fest`
Oct 7 – 9	American Printing History Association's 41st annual conference, "Black Art and Printers' Devils", Huntington Library, San Marino, California. `printinghistory.org`
Oct 14 – 18	ASIS&T 2016 Annual Meeting, "Creating Knowledge, Enhancing Lives through Information & Technology", American Society for Information Science and Technology, Copenhagen, Denmark. `www.asist.org/events/annual-meeting`
Oct 17	Award Ceremony: The Updike Prize for Student Type Design, Speaker: Fiona Ross, Providence Public Library, Providence, Rhode Island. `www.provlib.org/updikeprize`
Oct 29	GuIT Meeting 2016, XIII Annual Conference, Brescia, Italy. `www.guitex.org/home/en/meeting`

2017

Feb 1	**TUG election:** nominations due. `tug.org/election`
Feb 23 – 25	Typography Day 2017, "Typography and Diversity", Department of Integrated Design University of Moratuwa, Sri Lanka. `www.typoday.in`
Mar 22 – 24	DANTE 2017 Frühjahrstagung and 56th meeting, Deutsches Elektronen-Synchrotron (DESY), Zeuthen, Germany. `www.dante.de/events.html`
Mar 30 – 31	Center for Printing History & Culture, "From Craft to Technology and Back Again: print's progress in the twentieth century", National Print Museum, Dublin, Ireland. `http://www.cphc.org.uk/events`

TUG 2017 & BachoTeX 2017
Bachotek, Poland.

Apr 29 – May 3	The 38th annual meeting of the TeX Users Group, jointly with the 25th meeting of GUST and GUST's 25th birthday. `tug.org/tug2017`

May 21 – 26	16th Annual Book History Workshop, Texas A & M University, College Station, Texas. `cushing.library.tamu.edu/programs/bookhistoryworkshop`
May 25 – 27	TYPO Berlin 2017, "Wanderlust", Berlin, Germany. `typotalks.com/berlin`
Jul 5 – 7	The Fifteenth International Conference on New Directions in the Humanities (formerly Books, Publishing, and Libraries), "New Directions of the Humanities in the Knowledge Society", Imperial College, London, UK. `thehumanities.com/2017-conference`
Jul 30 – Aug 3	SIGGRAPH 2017, "At the ♡ of Computer Graphics & Interactive Techniques", Los Angeles, California. `s2017.siggraph.org`

Status as of 15 September 2016

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.

User group meeting announcements are posted at `lists.tug.org/tex-meetings`. Interested users can subscribe and/or post to the list, and are encouraged to do so.

Other calendars of typographic interest are linked from `tug.org/calendar.html`.

TUGBOAT Volume 37 (2016), No. 2