TUGboat

Volume 44, Number 3 / 2023

TeX Users Group

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

Individual memberships

2023 dues for individual members are as follows (all rates will likely have a small increase next year):

- Trial rate for new members: $30.
- 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. Members may also choose to receive *TUGboat* and other benefits electronically, at a discount. All membership options are described at `tug.org/join`.

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 carries with it such rights and responsibilities as voting in TUG elections. All the details are on the TUG web site.

Journal subscriptions

TUGboat subscriptions (non-voting) are available to libraries and other organizations or individuals for whom memberships are either not appropriate or desired. Subscriptions are delivered on a calendar year basis. The subscription rate for 2023 is $115.

Institutional memberships

Institutional membership is primarily a means of showing continuing interest in and support for TeX and TUG. It also provides a discounted membership rate, site-wide electronic access, and other benefits. For further information, see `tug.org/instmem` 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.

About the cover

The Alhambra-inspired tessellation on the cover was created by Laurence Finston. His article on pages 378–389 describes the process, using his program GNU 3DLDF.

[printing date: November 2023]

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

`tug.org`
`tug.org/TUGboat`

Electronic mail

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

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

Volunteer TeXnical support, public mailing list:
`support@tug.org`

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

A typewriter (or a computer-driven printer of similar quality) that justifies its lines in imitation of typesetting is a presumptuous machine, mimicking the outer form instead of the inner truth of typography.

Robert Bringhurst
The Elements of Typographic Style
(1992)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 44, NUMBER 3, 2023

PORTLAND, OREGON, U.S.A.

TUGboat editorial information

This regular issue (Vol. 44, No. 3) is the last issue of the 2023 volume year. The deadline for the first issue of next year is March 24, 2024. Contributions are requested.

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

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

TUGboat editorial board

TUGboat advertising

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

Submitting items for publication

Proposals and requests for *TUGboat* articles are gratefully received. Please submit contributions by electronic mail to `TUGboat@tug.org`.

The *TUGboat* style files, for use with `plain` TEX and LATEX, are available from CTAN and the *TUGboat* web site, and are included in TEX distributions. We also accept submissions using ConTEXt. For deadlines, templates, author tips, and more, see `tug.org/TUGboat`.

Effective with the 2005 volume year, submission of a new manuscript implies permission to publish the article, if accepted, on the *TUGboat* web site, as well as in print. As of fall 2023, submission also implies permission to be indexed in the EBSCO (a large subscription management company) databases. We made this agreement with EBSCO to provide more visibility to *TUGboat* articles. See `tug.org/TUGboat/tubperm.html` for more.

If you have any concerns about these permissions, please notify the editors at the time of submission and we will do our best to make suitable arrangements.

Other TUG publications

TUG is interested in considering additional manuscripts for publication, such as manuals, instructional materials, documentation, or works on any other topic that might be useful to the TEX community in general.

If you have such items or know of any that you would like considered for publication, please contact the Publications Committee at `tug-pub@tug.org`.

TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits:
`tug.org/instmem`
Thanks to all for their support!

American Mathematical Society, *Providence*, *Rhode Island.* `ams.org`

Association for Computing Machinery, *New York*, *New York.* `acm.org`

Aware Software, *Newark*, *Delaware.* `awaresw.com`

Center for Computing Sciences, *Bowie*, *Maryland.*

CSTUG, *Praha*, *Czech Republic.* `cstug.cz`

CTAN. `ctan.org`

Duke University Press, *Durham*, *North Carolina.* `dukeupress.edu`

Hindawi Foundation, *London*, *UK.* `hindawi.org`

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

L3Harris, *Melbourne*, *Florida.* `l3harris.com`

LATEX Project. `latex-project.org`

MacTEX. `tug.org/mactex`

Maluhy & Co., *São Paulo*, *Brazil.* `maluhy.com.br`

Marquette University, *Milwaukee*, *Wisconsin.* `marquette.edu`

Masaryk University, Faculty of Informatics, *Brno*, *Czech Republic.* `fi.muni.cz`

Modular Font Editor K. `mfek.org`

Nagwa Limited, *Windsor*, *UK.* `nagwa.com`

Overleaf, *London*, *UK.* `overleaf.com`

StackExchange, *New York City*, *New York.* `tex.stackexchange.com`

TEXFolio, *Trivandrum*, *India.* `texfolio.org`

Université Laval, *Ste-Foy*, *Québec*, *Canada.* `bibl.ulaval.ca`

University of Ontario, Institute of Technology, *Oshawa*, *Ontario*, *Canada.* `ontariotechu.ca`

University of Oslo, Institute of Informatics, *Blindern*, *Oslo*, *Norway.* `uio.no`

VTEX UAB, *Vilnius*, *Lithuania.* `vtex.lt`

Editorial comments

Barbara Beeton

John Warnock (1940–2023)

Although John Warnock was not a member of the TeX community, his work was central to the environment that allowed it to flourish. Born October 6, 1940, in Holladay, Utah, he pursued an academic program in mathematics, but after a few years in a doctoral program at the University of Utah switched to electrical engineering, which at the time encompassed computer science. There, he developed a computer graphics algorithm to render a three-dimensional image in two dimensions.

After receiving his doctorate, he moved to the San Francisco Bay area, and in time went to work for Xerox PARC (the Palo Alto Research Center). It was there that he met Chuck Geschke. A product of their work was InterPress, a tool for getting printers to render an image from a computer screen. But Xerox wasn't interested, and Warnock's disappointment led to his leaving Xerox to pursue his vision.

In 1982, he and Geschke founded Adobe Systems. Among Adobe's many groundbreaking products directed at document production using personal computers and laser printers are PostScript, PDF, InDesign, Illustrator, Photoshop and Acrobat.

In July of this year, Don Knuth, in a message to Chuck Bigelow, mentioned "I don't know the name of the genius who invented the hints that were used in Adobe's Type 1 fonts."[1] Bigelow responded

> John Warnock has publicly claimed credit for Doug Brotz, Bill Paxton, and himself for the idea and implementation of distorting character outlines to fit outline characters to raster grids before rasterization.

In an effort to find the authoritative answer, I reached out to Frank Romano, president of the Museum of Printing and long-time observer of and commentator on all things about printing, and he, in turn, asked Warnock directly. In a message dated July 4, Warnock responded with a copy of the slides for the lecture he had given when presented in 2004 with the Lovelace Medal by the British Computer Society. In it, he doesn't say unambiguously that he was the inventor of hints, but presents the method used by PostScript: "The Good Idea!" This comprises two parts: "Don't try to figure out what bits to turn on or off." and "Instead, change the letter forms so the result is good." The illustrations demonstrate the effectiveness of the method.

[1] Quoted in a personal message from Chuck Bigelow dated 2 July 2023.

Dr. Warnock died on August 19. Romano mused that the last email he'd received was Warnock's communication regarding hinting.

Taco Hoekwater and METAPOST

METAPOST, a graphics program created by John Hobby, inspired by Knuth's METAFONT, has continued to be developed by a team centered in the Netherlands and populated largely by members of the ConTeXt community.[2] A key member of this group is Taco Hoekwater.

Taco has written a series of articles on various facets of METAPOST that have appeared online in the pages of the *context group Proceedings*. These articles address the following topics. The last two are from presentations at the 2023 ConTeXt meeting, and will not be posted until March 2024.

- "Sparks, tags, suffixes and subscripts",[3] *context group Proceedings* 2019, pp. 10-21.
- "METAPOST definitions",[4] *context group Proceedings* 2020, pp. 53–72.
- "METAPOST paths and pairs",[5] *context group Proceedings* 2021, pp. 48–92.
- "METAPOST control structures".[6] *context group Proceedings* 2023, pp. 70–78.
- "METAPOST pictures",[7] *context group Proceedings* 2023, pp. 79-93.

Frans Goddijn has now pulled together all five articles to form an issue of the NTG *MAPS*, Nummer 53, Voorjaar 2023,[8] under these slightly changed titles: "Variables", "Definitions", "Paths, pairs, pens and transforms", "Conditions and loops", and "Colors and pictures". This issue, like current issues of some other TeX user group journals, is available only for members of the group for some period. For NTG, the period is two years.

In the introduction to the collection, Taco writes

> There is a lot of information in this set of articles, but this is not a manual. [...] The point of this set of articles is not to replace [the] manual, but to elaborate and clarify some parts of it.

Thanks to Henning Hraban Ramm for providing authoritative editing assistance.

[2] Editor's note: The variations on the presentation of "ConTeXt" here follow the usage within the ConTeXt group.
[3] articles.contextgarden.net/journal/2019/10-21.pdf
[4] articles.contextgarden.net/journal/2020/53-72.pdf
[5] articles.contextgarden.net/journal/2021/48-92.pdf
[6] articles.contextgarden.net/journal/2023/70-78.pdf
[7] articles.contextgarden.net/journal/2023/79-93.pdf
[8] www.ntg.nl/maps/53 (when available)

doi.org/10.47397/tb/44-3/tb138beet

TUG annual meetings: 2023 and 2024

The videos of the talks at TUG 2023 have been post-processed and uploaded to YouTube. They can be accessed at `youtube.com/@TeXUsersGroup`. Only a single playlist is provided for 2023 since, unlike the previous two years, no interviews took place, so everything is collected under "Lectures". Thanks to everyone who worked hard to make this happen.

As for 2024, an invitation was extended to, and accepted by, the Board to hold the meeting in Prague. Dates have not been set yet, but it will most likely take place in July or early August. When more information is known, the dates will be added to the calendar, and details posted at `tug.org/tug2024`. Stay tuned.

TeX Collection DVD discontinued for 2024

As time goes on, an increasing portion of the TeX community has taken to updating their personal TeX installation directly from a Web source like CTAN, or has started to use a third-party service like Overleaf. More and more members have notified the TUG office that they don't need the DVD.

It takes considerable time and money to produce and mail the DVD. Thus, the TUG Board has decided to no longer provide, as of 2024, the DVD as a standard benefit of membership (for non-electronic members).

Nonetheless, there are valid reasons to need an offline mechanism to install or update a TeX system. As practiced by Don Knuth, a user's hardware may not be connected to the net; that might be for either personal or corporate reasons, but the disconnect can be absolute. In such a case, a DVD provides the most convenient installation method. The TUG office received about 20 requests for DVDs last year, mostly from nonmembers; it's not possible to know how many of the DVDs distributed to members were actually used to install the system.

Preparation of TeX Live for 2024 will continue to build images suitable for burning to DVD, and the other components of the TeX Collection are also expected to be available, if desired; only the manufacturing step will be omitted. Instead, several volunteers have already agreed to burn DVDs from the official release on request; the actual procedure hasn't yet been established.

If you are willing to participate in this effort, or know already that you will need the 2024 software releases in physical form, the best way to keep track of what is happening is to follow `tug.org/texcollection`, where the current status will be communicated. Another possibility is to subscribe to the public mailing list `lists.tug.org/texdvd` which is intended for discussion among those interested. Or do both.

Shift Happens: How keyboards got lowercase

Remember teletypes? Keypunches? All 26 letters of the English alphabet were there on the keyboard, but only in uppercase. What would be printed out now looks, to contemporary eyes, like SCREAMING. First-generation typewriters also had only one case — upper — and if these machines were to become effective for use in business, something had to be done.

The story of how the case shift was added to the typewriter has now been told, in great graphic detail, in *Shift Happens*, by Marcin Wichary. The book hasn't yet been delivered, but is expected before the end of the year. (If you are interested, go to `www.kickstarter.com/projects/mwichary/shift-happens` for more information.)

In anticipation of publication, a discussion took place at the Museum of Printing (MoP) on July 8. Entitled "Shift happens meets etaoin shrdlu", the panelists were: Frank Romano (moderator, president of MoP), Marcin Wichary (the author), Glenn Fleischman (editor of *Shift Happens*, author and long-time technical journalist), Jeff Jarvis (journalist and observer of the transition from hot type to cold in newspapers), and Doug Wilson (creator of *Linotype: The Film*[9]).

The discussion, more than an hour long, was recorded, and is posted on MoP's YouTube channel in three parts.[10] Sadly, I wasn't present at the event to kibitz, but would have had some comments on a few topics mentioned by the panelists:

- Does anyone still recognize the name Ottmar Mergenthaler, or remember what he was known for?[11]
- Nobody mentioned that an early method for producing copy for material to be printed was to type it into machines that could punch paper tape (which could be edited), then read that tape back in, with the possibility of the typed output being automatically justified. The Friden Justowriter was such a machine.[12]

[9] `linotypefilm.com`

[10] `www.youtube.com/watch?v=HrlcFB_Q_UE`
`www.youtube.com/watch?v=-8a4XwOtd9k`
`www.youtube.com/watch?v=SWfDrcdXfaM`

[11] Mergenthaler developed the Linotype in Baltimore, and a public high school, Mergenthaler Vocational-Technical High School — MerVo-Tech — is named in his honor.

[12] For many years the membership list of the American Mathematical Society was produced from edge-punched cards, one per name and address, which were stored alphabetically in card trays, updated as changes were received, and once a year run back through the Justowriter to produce the camera copy for publication.

- The "correcting" IBM Selectric had two ribbons, one black carbon that produced crisp images, and another that appeared white, but was actually covered with adhesive that, when a letter was retyped in place, could lift the carbon image off the paper and leave a clean surface to type a correction.

Do watch the video — it touches on many facets of composition and printing, not just typewriters. And while you're at it, consider subscribing to the MoP YouTube channel; other videos posted there cover a variety of print-related topics.

Nelson Beebe's bibliographic archive

For many years, Nelson Beebe has been compiling an archive of carefully curated BibTeX files recording published material in subject areas of interest to scientists of all stripes, in particular those who are users of TeX. An introduction to the archive is at:

`www.math.utah.edu/pub/tex/bib`.

The page carries the title "The TUG bibliography archive".

The collection is segmented into many subareas; these are listed:

- Journals on the history, philosophy, and popularization of mathematics and science
- Computer science journals and topics
- ACM Transactions
- Markup, programming, scripting, and symbolic algebra languages and systems
- Cryptography
- Fonts and typography
- IEEE journals
- Computational/quantum chemistry/physics journals
- Numerical analysis journals
- Probability and statistics journals
- SIAM journals
- Mathematics journals
- Mathematical and computational biology
- General topics
- Internet
- Markup, programming, scripting, and symbolic algebra languages and systems
- Operating and database systems
- Fisheries research
- Publication metrics
- Standards

Coverage for all issues is complete, if data can be found. The number of entries is now over a million.

Included in the detailed entries are many elements not defined in the original BibTeX spec, including URLs and ORCID information. If relevant information for an entry is available in online metadata, it will be present in the entry in this collection; where essential elements are missing from an online source, Nelson researches it thoroughly and includes what he finds. But the presence of a particular journal in the BibTeX archive does not guarantee that the original is available online.

A related archive has been compiled by the BibNet Project: `www.math.utah.edu/pub/bibnet`.

Question regarding hyphenation: chemistry

I would like to find an answer to this question:

> Can chemistry be considered a "language" for purposes of hyphenation?

The names of chemical compounds tend to be quite long, and are themselves compounded of shorter segments in a way that often does not lend itself to being hyphenated acceptably by the default US hyphenation patterns. Two more characteristics are relevant here: Many, if not most, such names seem to be used in the same form regardless of which western European language they appear in, and the longer names are very infrequent in documents outside of the disciplines in which they are common, e.g. chemistry, medicine/pharmacology, and the like. Therefore, adding a large number of exception entries for such terms to `hyphenex` doesn't help a large proportion of (La)TeX users. (Nevertheless, a few of the terms are included in `hyphenex` now, in a separate section.)

This has been mentioned before in the introductory prose accompanying the periodic updates to the exception list.[13] I have tried to solicit opinions from some users who have mentioned in various Q&A forums that they have made such specialized `\hyphenation` lists for themselves, but have received no answer. Therefore I'm now asking here for help.

If a substantial list of chemical terms could be built, with proper hyphenation markings included, it might be possible to derive a specialized set of patterns for use in documents where there is heavy use of this vocabulary.

Send your thoughts and suggestions to me in care of `TUGboat@tug.org`. Thanks.

◇ Barbara Beeton
`https://tug.org/TUGboat`

[13] See *TUGboat* 42:1, p. 11, `tug.org/TUGboat/tb42-1/tb130hyf.pdf` and earlier installments.

Typographers' Inn

Peter Flynn

Identifying a typeface

I see lots of queries posted to various platforms asking, in effect, 'What font is this?' (usually meaning typeface rather than font). There are several very good typeface recognition web sites such as *What The Font?*[1] where you can upload an image of some text, and the rather more accurate question-and-answer interface of *Identifont*,[2] which matches 'characteristics' (aspects of the letter-forms which make the typeface you want distinct from others).

However, I was interested to see if something similar could be done for matches in the LaTeX Font Catalogue. In personal cases you might be able find a typeface in the Catalogue which is 'close enough', whereas an exact match might mean a normal commercial typeface, which can be expensive.

To find a typeface you need to know something about its distinguishing characteristics. There are many sets of classifications which could be considered, but I wanted to avoid the historical approach (blackletter, humanist, antiqua, transitional, grotesque, geometric, modern, etc.) because it is dense with pitfalls, and requires too much explanation. The first distinction is usually much simpler: between serif faces (with tail-pieces on the uprights) and sans-serif faces (without the tails), because this is usually visually obvious. Sometimes you're not matching a sample, though: you just want a typeface that has certain characteristics, one that looks a certain way; and this means that information has to be collected before you can start searching.

For the LaTeX Font Catalogue I put together a few characteristics to test:

- the form of the lowercase 'a': hooked (as here [Computer Modern]) or round (as in Almendra [a]);
- the form of the lowercase 'e': horizontal (as here) or sloped (as in Accanthis ADF Standard No.2 [e]);
- the form of the lowercase 'g': fish-hook (as in DejaVu Serif [g]) or double-loop (as here);
- the height of the ascenders: above the cap-height (as in QT Agate [The]), at cap-height (as in Berenis ADF Pro [The]), or below cap height (as in GFS Neohellenic [The]);
- the type of serif: many different types, often not mutually exclusive (bracketed, slab, flared, spurred, square, or hairline);
- the form of the uppercase 'R': s-shaped (as here), straight (as in Almendra [R]), or curved (as in QT Agate [R]) (with variants);
- the form of the lowercase 'c': a bulb (as here) or a serif (Alegreya [c]).

These characteristics were derived from queries I had received in 2022 and early 2023, and I spent some time applying them to the contents of the LaTeX Font Catalogue to see if this scheme was feasible. It is, and I got far enough to let me answer the queries about fonts, but it clearly needs more work to finish the list and apply other characteristics. Ideally this should be done in parallel with the LaTeX Font Catalogue, so if anyone has strong feelings about specific characteristics that distinguish one face from another *in a way that will help beginners*, please get in touch.

Dashing it off II (em rules revisited)

In this column in *TUGboat* 37:3 I mentioned the differences between publishers' 'house rules' on spaced and unspaced en and em rules, and the discussion in `comp.text.tex` which revealed some inconsistencies [3], so now some confirmations!

This came up again earlier this year, this time in TYPO-L, where Yateendra Joshi sought confirmation that unspaced em dashes were standard in US publishing, noting

> In the UK and Europe, I often see spaced en dashes when they occur in pairs but an unspaced em dash when it occurs singly.

Leila Singleton confirmed the US practice, deriving it from the MLA Handbook conventions [5] used by book and journal publishers, but noting that the Associated Press Stylebook (used by newspapers and sometimes magazines) instead calls for spaces [1].

Finally, Chris Maden pointed out that Jan Tschichold's influential design for Penguin Books included spaced en dashes instead of em dashes, and that directive (and a few others) saw wide uptake throughout British typography.

So now we know.

Discord for the LaTeX community

Discord is a messaging system handling text, video, and voice, originally set up for online gaming communities but now used by all kinds of groups. Topic-based discussions are supported by persistent 'servers' a little like chatrooms. Discord is available on all platforms and in-browser. There has been a LaTeX support server since April 2019.[3]

[1] `www.myfonts.com/pages/whatthefont`
[2] `www.identifont.com/`
[3] Named "LaTeX Support": `disboard.org/server/570670498309210112`

doi.org/10.47397/tb/44-3/tb138inn

It has been illuminating reading. Some of the queries are short and simple: ‘Hi, I can’t figure out how to make [*description*]’ (often homework, as the same problems reoccur) and only need pointing in the direction of the right package. Some are much more complex, where the user has gotten themselves and their thesis formatting tangled deep in a nest of irrelevant complexities suggested by Googling the perceived problem.

I have learned some useful lessons:

- New migrants from Word lack the vocabulary to express their difficulty; this is especially a problem with STEM students referring to packages, environments, and macros alike as ‘functions’;
- Users of the in-browser TeX editor *Overleaf* call classes ‘templates’, and find difficulty distinguishing *Overleaf* features from LaTeX;
- Many newcomers have been misled by the expectations of their peers, who have taught them incorrectly (see ‘sitting by Nellie’ in [4, p. 71] as a teaching method);
- Many are baffled by the idea of reading documentation, which is an alien activity to them.

I’ll be campaigning for a pinned post with a succinct one-para explanation of LaTeX and links to CTAN and MWEs.

Afterthought

A user on Discord asked why his embedded image (using wrapfig) wasn’t working. It turned out to be a conflict with lipsum, but I noticed that he was trying to place a circular logo on the (uncommon) *right-hand* side of the paragraph.

Sure enough, the next day a former colleague commented on the image in Figure 1 (posted on what’s left of Twitter/X) with a roundel embedded at the right-hand side of each paragraph, with a figure number for each one, outlined in red.

Elsewhere I have previously used a page from Vesalius from 1543 as an example of a formal figure [2, p. 38], with all the modern paraphernalia of caption, number, cross-references, and legend;[4] but it was interesting to see the beginnings of this in the manuscript era.

I’m hoping to get a transcription of the page to make a typographic reconstruction with wrapfig, bringing 21st century formatting to 15th century scholarship.

[4] He was by no means the earliest, but he was notoriously picky about how and where his figures should be placed.

Figure 1: 15th century astrological treatise [By kind permission of The Bodleian Libraries, University of Oxford. MS. Rawl. D. 1220, f. 7v]

References

[1] Associated Press. *Associated Press Stylebook 2022–2024*. Associated Press, New York, NY, 56th ed., Aug. 2022.

[2] P. Flynn. *Human interfaces to structured documents: The usability of software for authoring and editing.* Ph.D. thesis, University College Cork, Aug. 2014. `cora.ucc.ie/bitstream/handle/10468/1690/Human-Interfaces-to-Structured-Documents.pdf#page=38`

[3] P. Flynn. Typographers’ Inn: Dashing it off. *TUGboat*, 37(3):264, 2016. `tug.org/TUGboat/tb37-3/tb117inn.pdf`

[4] P. Flynn. Typographers’ Inn: No time to learn. *TUGboat*, 41(1):71, 2020. `tug.org/TUGboat/tb41-1/tb127inn.pdf`

[5] Modern Language Association of America. *The MLA Handbook.* Modern Language Association of America, New York, NY, 9th ed., May 2021.

◇ Peter Flynn
Textual Therapy Division, Silmaril Consultants
Cork, Ireland
`peter (at) silmaril dot ie`
`blogs.silmaril.ie/peter`

A short history of Greek type design before the digital era

George D. Matthiopoulos

The art of Typography transformed the book into a mass cultural tool, and it has been recognized as the midwife of modernity. The field of Typography includes punchcutting, matrix making, typecasting, typesetting of a given text and layout-pagination for printing in multiple copies. Gutenberg conceived and realized his idea of reusable metal type sorts by the middle of the 15th c., when the textura Gothic hand was the principal choice of the scribes and readers alike (pl. 1). The simplicity of the prevailing dense, vertical strokes produced by the pen informed the conceptualization of his solution and the success of its implementation. In effect, the printed book was born when the rising social demand for multiple copies of any text merged with the uncomplicated, almost mechanical motion of the prevailing writing hand of the era. This is the crux of the issue in terms of design and technological innovation, and it should be well understood, before we begin our probing into the peculiarities of the Greek types.

During the long Byzantine era, the ancient Greek alphabet had evolved gradually from majuscule to minuscule form (pl. 2–7), just like the Latin script. The Byzantine hand had accumulated an influx of ligatures, abbreviations, flourishes and other decorative nuances, which defined its flamboyant cursive character. After a quick look at any Byzantine script, anyone would surmise that it bears little resemblance to the simple, rhythmic vertical strokes found in texts written in Gothic hand.

As soon as the new art was brought south of the Alps by two German monks, Conrad Sweynheim and Arnold Pannartz, to serve the increasing pace of the new Italian humanism, the need to incorporate Greek passages in printed books immediately became apparent. The ancient Greek texts on Philosophy, Theater, History, Science and Theology were the core pillars of the awakening intellect in Europe and many scholars had come to realize that the few existing Latin translations were a poor substitute for the wealth of ideas and the linguistic beauty of expression that the originals contained. Having access to the Greek writings and learning to speak and write in the ancient language were deemed to be the only secure way to achieve textual accuracy and deeper knowledge. Therefore, it is not coincidental that the second written language to make it to the cutting-edge printing technology was Latin's older sister, Greek, and that its letterforms therefore needed to be cut and reproduced in metal types.

In order to avoid the baffling complexity of the Greek writing, the early type cutters threw out the visual history of the Byzantine scripts and isolated, as best as they could, the 24 minuscule letters for short Greek passages in Latin treatises (pl. 8, 9). This route, which Robert Proctor named as Greco-Latin, can be seen in many early editions and it lingered on as late as the first decades of the 16th c., when it was the choice of a new type cut and used in the first Polyglot Bible, which was published at the University of Alcalá de Henares in Spain (pl. 10).

However, new experiments and innovations concerning the shape of the Greek types were attempted since the mid-1470s in books entirely in Greek texts, including Byzantine capitals and diacritics for grammars, dictionaries, short classical texts, etc. (pl. 11). In 1486, two Cretan men of the cloth, Laonikos and Alexandros, were the first to attempt and, more significantly, to realize the unthinkable: A full-scale Byzantine hand composed of no less than 1,300 different sorts, of either single Greek characters or ligatured combinations (pl. 12). Their two small editions were not a publishing success and they slipped back to oblivion, but their idea was the catalyst for the future, pointing to alternative solutions as opposed to the current practices. They blazed a new path for Greek printing and their precedent opened the way for the aspiring Aldus Manutius to gamble with their paradigm in achieving his publishing ambitions with a team of dedicated Cretan scholars.

Aldus' herculean goal of publishing the first definitive edition (editio princeps) of all the known Greek classical texts proved to be very successful and his books were in great demand. Thus, the decision to imitate the contemporary ligatured style of writing became the new standard in Greek publishing and soon it spread from Venice to the rest of Italy and to Western Europe (pl. 13).

In the late 1490s, the first publishing effort made by Greeks also appeared in Venice. Nikolaos Vlastos, a stationer, and Zacharias Kalliergis, a scribe, joined forces to edit, print and publish a large Byzantine Lexicon and three classical texts. The type used was skillfully cut by Kalliergis himself, again in full Byzantine ligatured fashion, but the success of the Aldine editions did not allow it to exert any lasting influence (pl. 14).

By the third decade of the 16th century the geopolitical dominance of France had established Paris as a rival cultural centre of Europe, and King

George D. Matthiopoulos
doi.org/10.47397/tb/44-3/tb138matthiopoulos-greek

Francois I, under the guidance of a circle of erudite scholars, embarked on a major publishing programme of Greek editions for the Royal Library and Printing House. The indefatigable scholar and royal printer Robert Estienne invited from Rome Angelos Vergikios, a renown Cretan scholar–calligrapher, to collaborate with Claude Garamont, the best type cutter of his generation, for the production of a new Greek typeface in three sizes, which came to be known as Grecs du roi (pl. 15). The universal admiration for the new type marked indelibly the development of Greek types for the next two centuries; consecutive generations of distinguished type cutters, such as Robert Granjon (pl. 16), Pierre Haultin, Christopher van Dyck (pl. 17), Nicholas Kis (pl. 18), William Caslon (pl. 19), Joan Fleischman, François Rosart, Jean-Louis de Boubers, Pierre-Simon Fournier (pl. 20, 21), etc., largely followed its fashion.

Since the 18th century an increasingly scientific utilitarianism was introduced during the Age of Reason when Dutch and German printers first attempted abandoning the numerous ligatures in Greek classical texts; an iconic typeface of this expanding trend was the Homer Greek type (1756), by the Scottish type cutter Alexander Wilson for the Foulis brothers, publishers for Glasgow University (pl. 22).

By the turn of the 19th century the simplification of the Greek type case was, more or less, completed in European publishing and with it the last precarious ties with the Byzantine past were irrevocably severed. The renewed and comprehensive interest in classical studies and archaeological expeditions in Greece led to a significant and sustained increase of Greek texts for the expanding education systems in Europe. At the same period, the "one nation — one state" geopolitics which resulted after the Napoleonic wars also had an effect in the emergence of new, "nation-specific" Greek typefaces in Italy, France, England and Germany: Giambattista Bodoni produced several neo-classical Greek fonts in Parma (pl. 23), Firmin Didot designed a new upright Greek text type in Paris (pl. 24), Richard Austin cut the inclined Porsonic Greek (pl. 25) used in most English publications, and Karl Tauchnitz entered Greek publishing in Leipzig with his own inclined type (pl. 26). Since their introduction in the early 1800s, these "ethnic paths" in typestyle dominated the Greek editions of each respective country, until well into the middle of the 20th century.

An important advantage of the drastically reduced Greek type case was increased literacy — as it made reading and writing much simpler — , which allowed the quick dissemination of the emerging democratic ideals in Western Europe among the enslaved Greeks and paved the way for the national emancipation in the 1820s against the Ottoman Empire. During the Greek revolution and after the formation of the new Greek Kingdom, the Didot Greek typeface was used almost exclusively in every book, newspaper or periodical for more than a century hence.

Until the end of the 19th century there were some attempts at new Greek designs in England and Germany, mostly following a romantic notion to "revive" an ideal model of the ancient Greek alphabet (pl. 27–35). Of these new propositions, only an upright version of the Tauchnitz model was destined to become a serious rival to the Didot's supremacy in Greek publications (pl. 36).

From 1900 onwards, the new technological developments in mechanical typesetting quickly established Linotype's and Monotype's hegemony in publishing, and both companies quickly adapted versions of the most popular Greek designs in their type library (pl. 37, 38), while leading type foundries in Leipzig (Schelter & Giesecke), Turin (Società Nebiolo) and Paris (Deberny & Peignot) continued to export most of the types used in Greece. When industrialization finally started to take hold in Greece after the Great War, two competing type foundries by A. Karotsis–V. Karidis (pl. 39) and by Em. Karpathakis (pl. 40) were formed during the inter-war period.

In Western Europe, some new ideas were introduced: a monoline font appeared in Paris for the classical publications of the Société Les Belles Lettres (1920s, pl. 41); Willi Wiegand cut an upright calligraphic Greek font for Bremer Presse (Bremen, 1923, pl. 42), while at the British Library, Victor Scholderer revived the Renaissance model of Johannes Rubeus (1487). It was cut by Monotype Ltd and became commercially available as New Hellenic (1927), providing some competition to the long-prevailing Porsonic model (pl. 43).

Monotype also produced Perpetua Greek (1928), an experimental hybrid design by Eric Gill, in an attempt, with Stanley Morison, to shoe-horn in the Greek letters as mere subordinates of Latin fonts (pl. 44).

Jan van Krimpen first designed the calligraphic Antigone Greek for Joh. Enschedé (1927, pl. 45) and then he introduced Romulus Greek (1928) at Monotype, following Gill's and Morison's aesthetic obsessions (pl. 46).

In the 1930s, some scholars in Greece started advocating for the simplification of the polytonic system (diacritics above the vowels) that was used in Greek writing since the Hellenistic period, but had no real use in reading. The opposition by the educational establishment and the Orthodox church, however, reacted strongly and the issue was quickly suppressed.

At the same period, the Modernist movement ushered in the sans-serif vogue in texts and soon similar attempts were applied to the Greek alphabet from Deberny & Peignot (pl. 47), Nebiolo (pl. 48), Linotype (pl. 49), and Em. Karpathakis in Greece (pl. 50).

After World War II, Monotype wanted to increase its sales of typesetting machines internationally and to this end the Type Drawing Office (TDO) introduced two significant additions to their Greek type collection: Greek Gill Sans (pl. 51) and Greek Times (pl. 52). Later, Greek Univers was also added to the collection (pl. 53).

Hermann Zapf designed a number of Greek typefaces in the 1950s for the D. Stempel foundry: two sans serifs (Artemis and Attika), the upright Heraklit, the calligraphic Frederika and later the humanistic sans, Optima (pl. 54).

The two old Greek type foundries were replaced by younger rivals; since the mid 1950s Th. Paraskevopoulos established his company (“PAP”, pl. 55), and I. Sarasitis his (“Victoria”) in the 1960s (pl. 56). For the next twenty years both companies produced all the staple typefaces for everyday use nationwide, while their numerous new designs focused primarily into display fonts for use in headlines of newspapers, periodicals or ads.

Since the 1960s, Linotype was involved in the fast-developing world of phototypesetting machines. Trying to support its prospective sales, the company also tried to extend renowned Latin typefaces into Greek. The first attempt was to introduce Caledonia Greek with little success (pl. 57), and later Matthew Carter was commissioned to “convert” several other best-sellers into Greek, during the early 1970s: Helvetica Greek, Century Schoolbook Greek, Baskerville Greek, etc. (pl. 58). These were much more successful as phototypesetting quickly replaced all text production in Greek publishing and advertising.

Another field of new Greek type design activity was the emergence of the letter-transfer sector, which Letraset Ltd invented and introduced during the 1960s. The great demand for easy-to-use display lettering soon became a necessity for designers and architects, so the Greek type catalogues of Letraset and Mecanorma, its French rival, quickly expanded to include many exotic designs (pl. 59).

Starting in the 1960s, the newspaper industry raised again the issue of the polytonic writing practice, which was considered very costly in typesetting and proofing, and pressed in favour of a much simpler monotonic system; It took another 20 years for the long-awaited reformation to be ratified in 1981.

In the late 1970s and in the 1980s, as Linotype and Monotype, the two behemoths, were quickly losing their market share, new electronic companies proliferated in the industry. The result was that various Latin type designs were hastily “hellenized”, with varying market acceptance, and used as a means to entice prospective equipment buyers: typesetting services, publishers, advertising agencies etc. (pl. 60). The brief phototypesetting period ended with a last attempt to introduce original type designs in 1989; Takis Katsoulidis, a Greek artist-engraver, designed two type families, Apollonia and Katsoulidis, for Agfa Corp., but by then it was too late (pl. 61). Desktop publishing and the new digital revolution swept away the old practices and ushered in new type applications and ideas.

A note on the plates. Many of the fonts illustrated are displayed with both a sample of the type in use and a separate showing of the upper- and lowercase letters at an enlarged size, on a shaded background indicating the x-height and baseline of each character in relation to the capitals of each font.

◇ George D. Matthiopoulos
Dept. of Graphic Design and
Visual Communication
School of Applied Arts and Civilization
University of West Attica, Athens
Greece
`https://www.greekfontsociety-gfs.gr`
`gmat (at) uniwa dot gr`

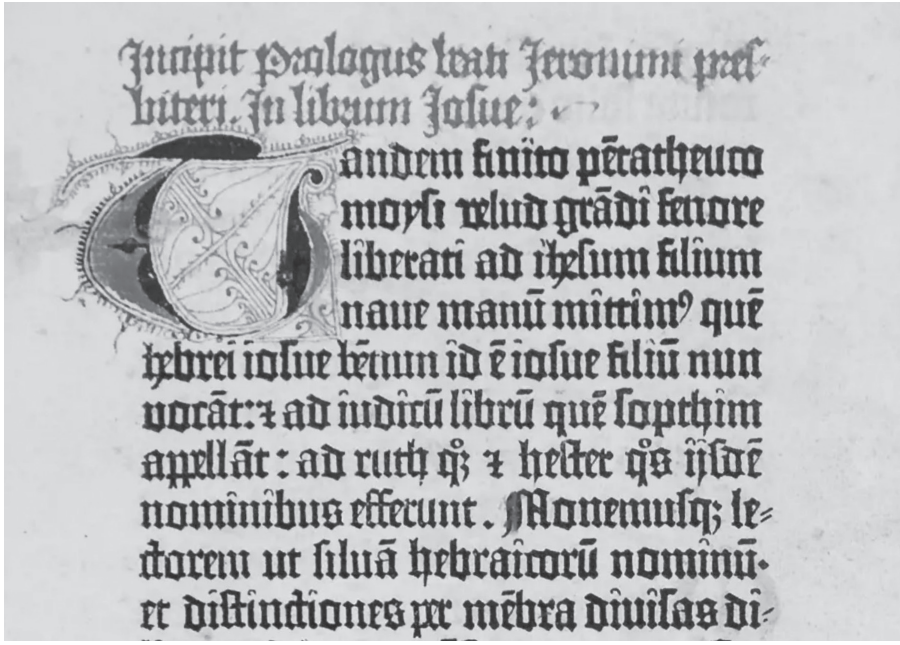


Plate 1: Johannes Gutenberg's Bible, Mainz, 1455.

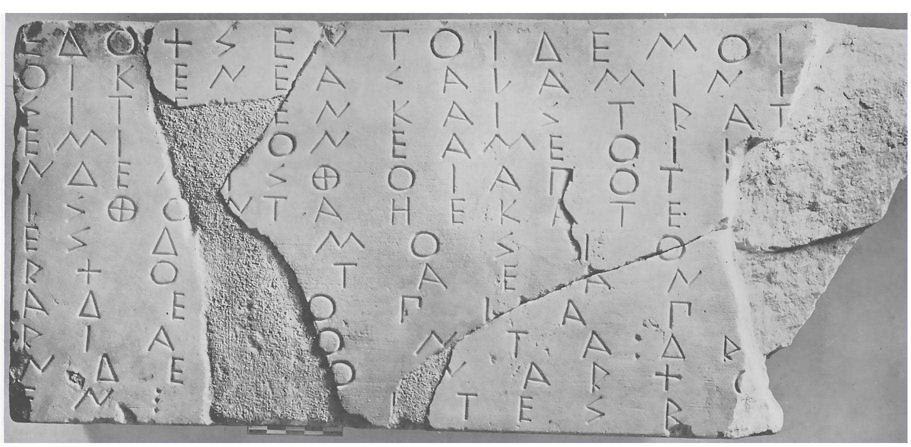
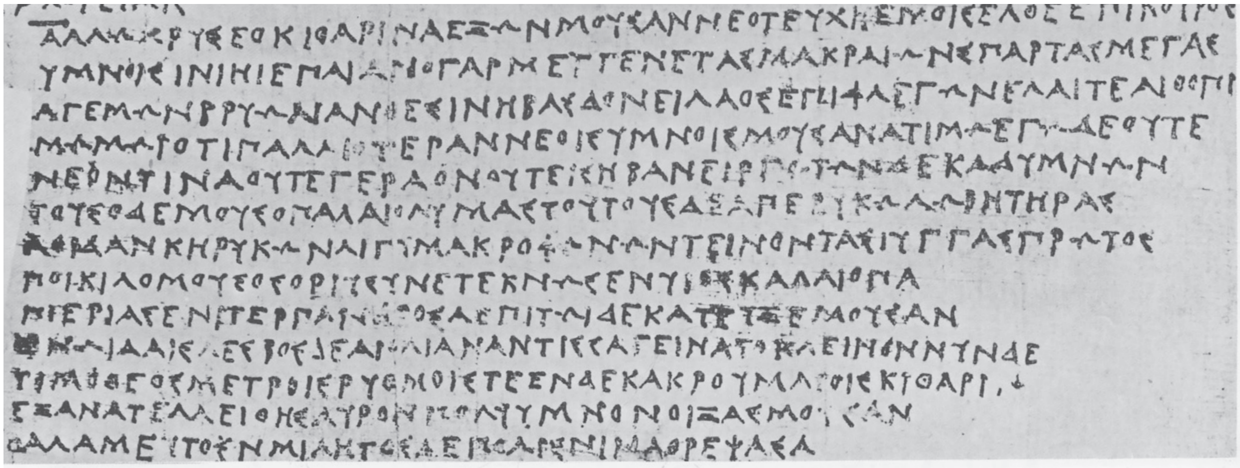
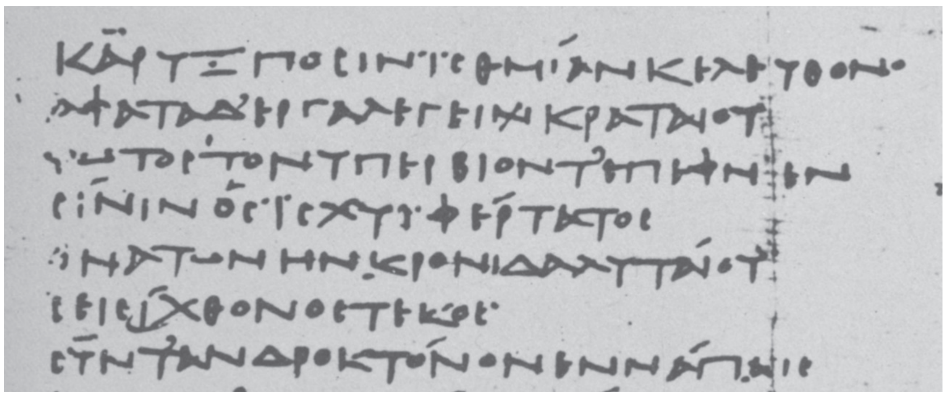


Plate 2: Greek majuscule writing on stone, 5th–4th c. BCE.





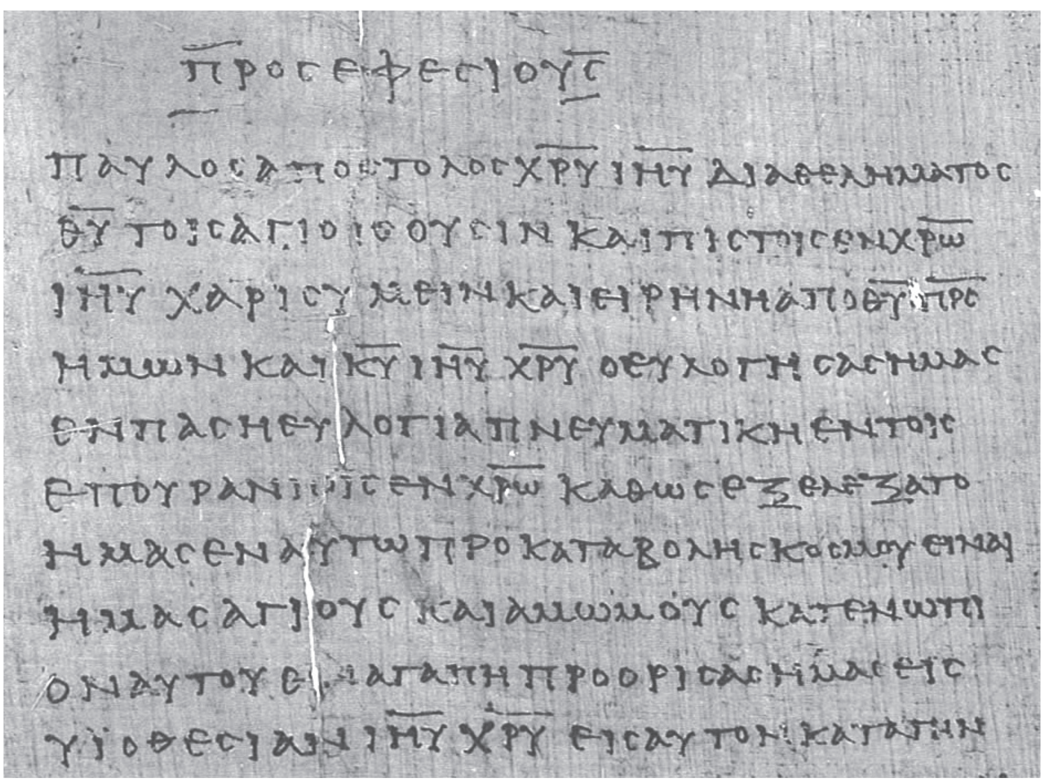


Plate 3: Greek majuscule writing on papyrus, 4th c. BCE–3rd c. CE.

Plate 4: Greek majuscule writing on parchment, 7th–9th c.

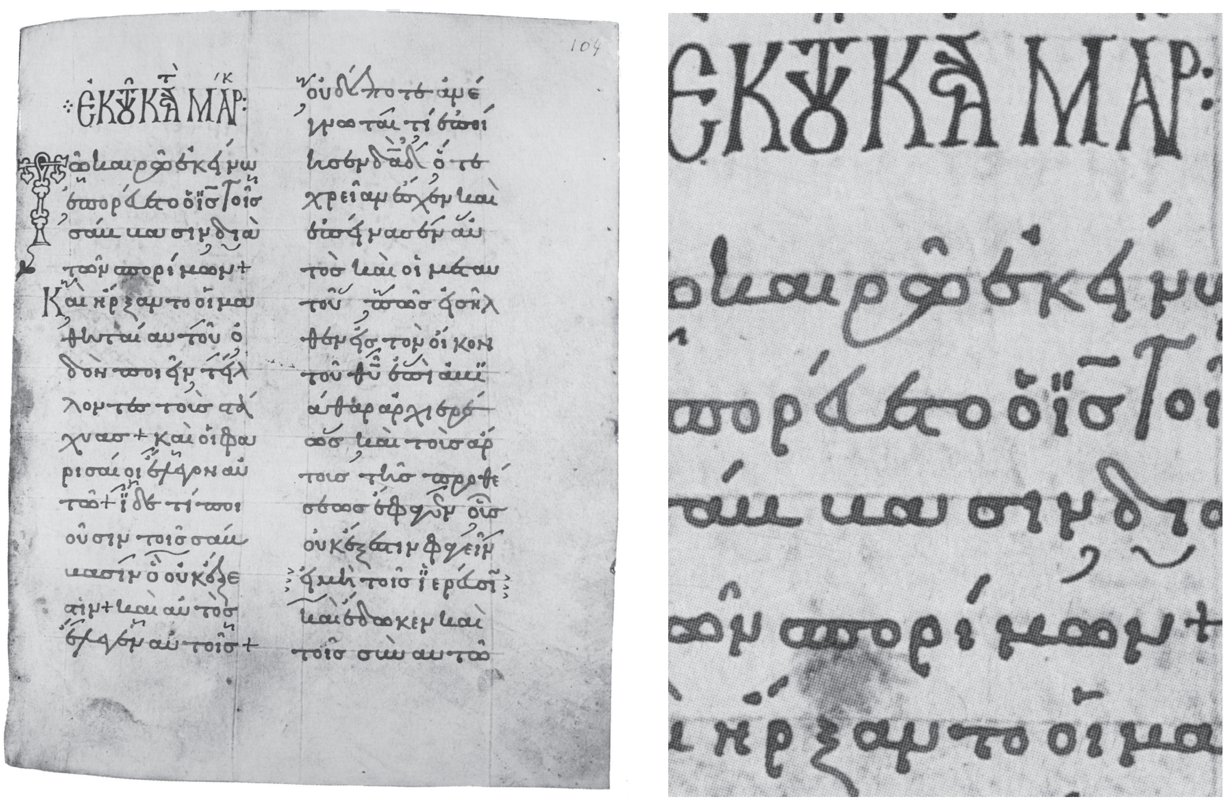


Plate 5: Greek minuscule writing on parchment, 12th c.

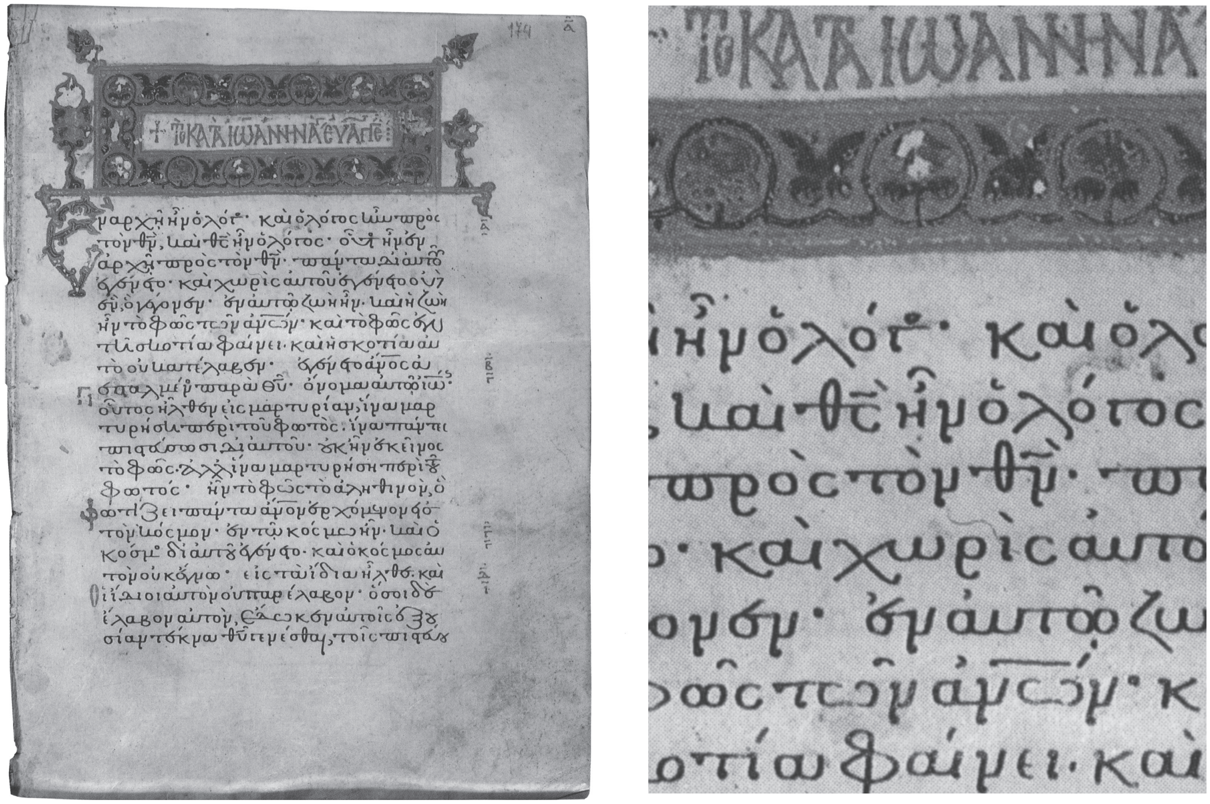


Plate 6: Greek minuscule writing on parchment, 13th c.

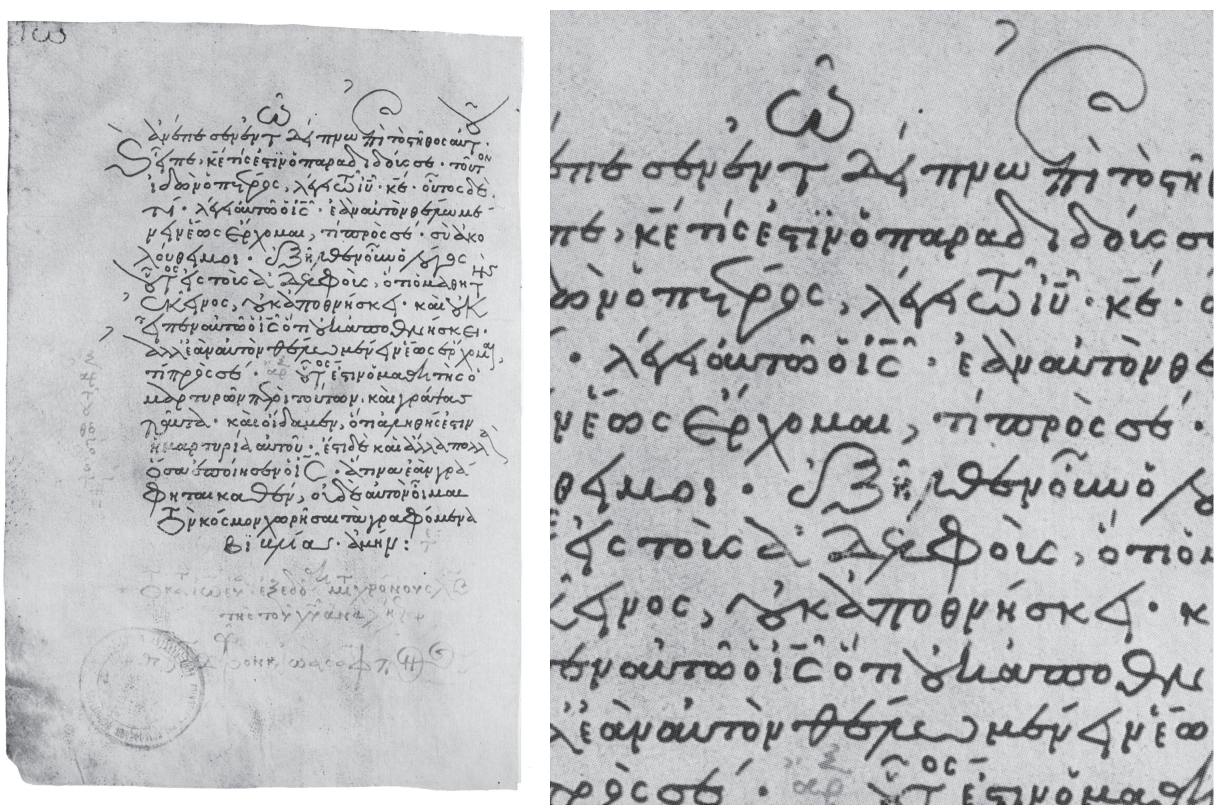
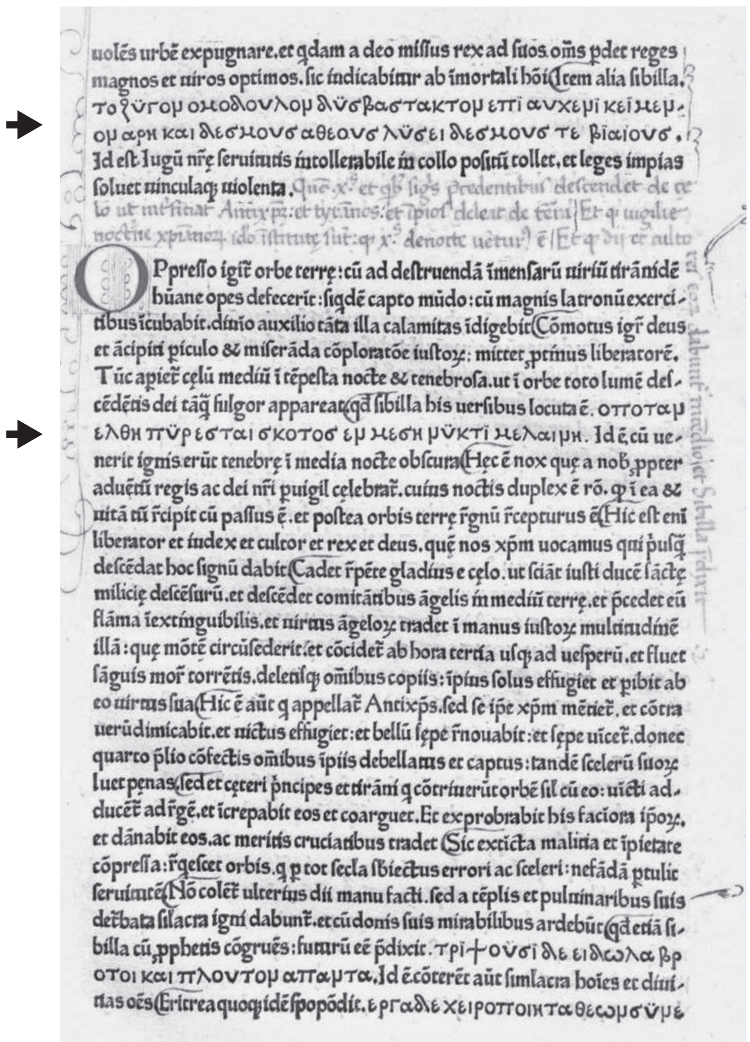


Plate 7: Greek minuscule writing on parchment, 14th c.



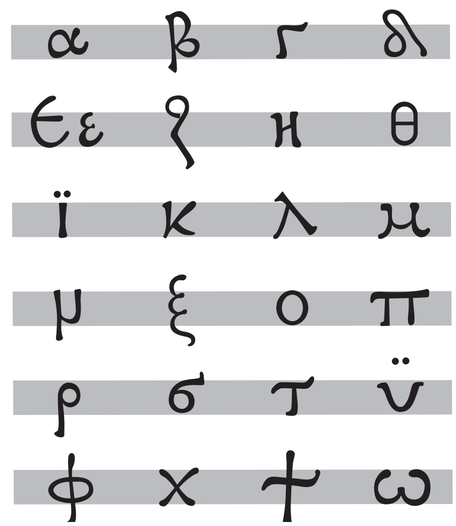


Plate 8: Lactantius, *Divinae Institutiones*, Subiaco, Conrad Sweynheim–Arnold Pannartz, 1465.

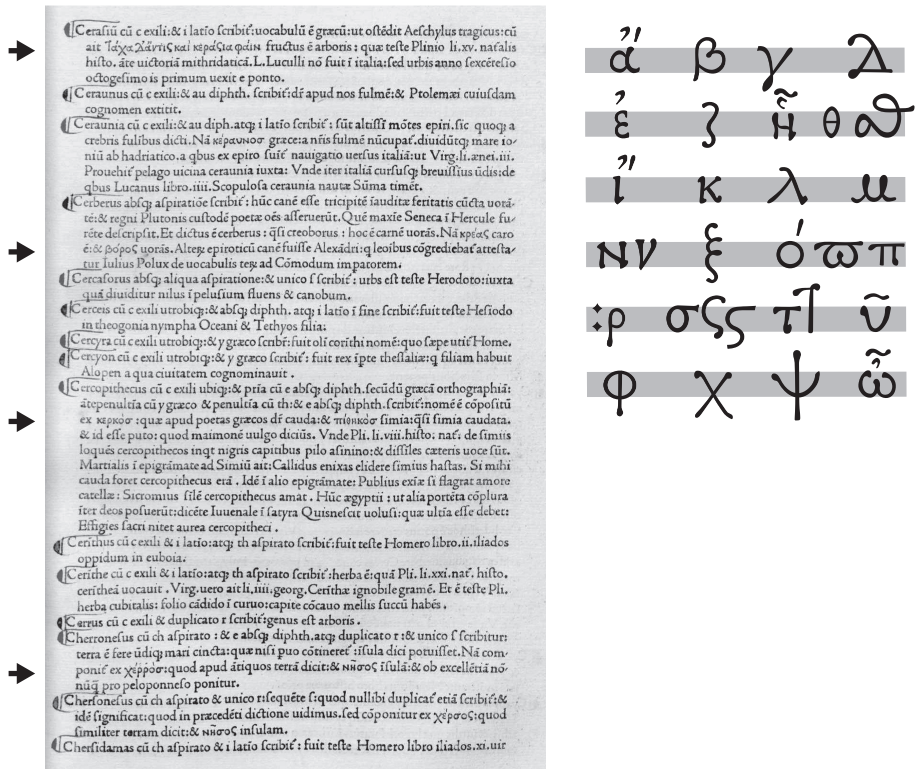


Plate 9: Tortellius, *De orthographia dictionum e Graecis tractarum*, Venice, Nicolas Jenson, 1471.

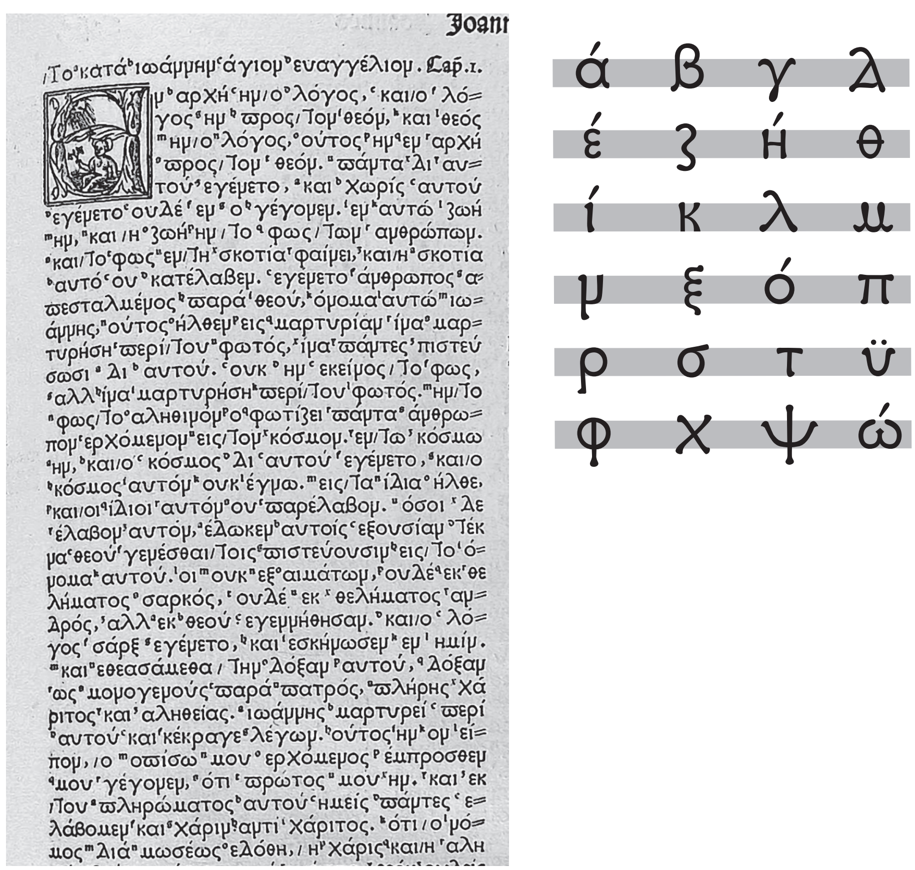


Plate 10: Tortellius, *Complutensis Biblia Sacra* (New Testament), Alcalá de Henares, A. Guillen de Brocar, 1512–1517.

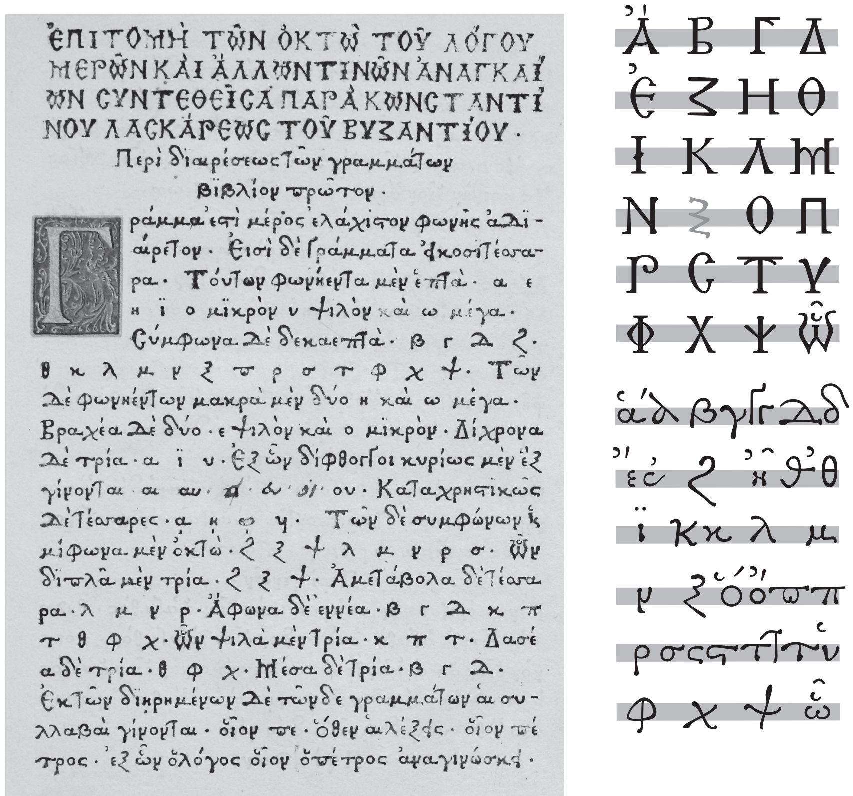


Plate 11: K. Lascaris, *Greek Grammar*, Milan, Demetrio Damilas–Dionigi Paravicino, 1476.

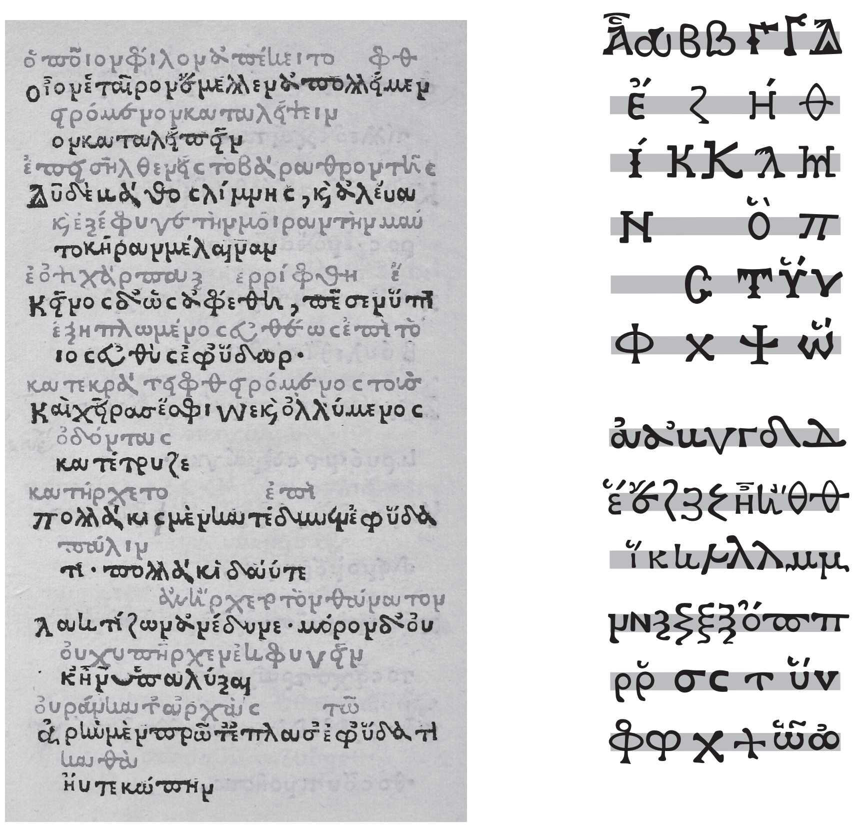


Plate 12: *Batrachomyomachia*, Venice, Laonikos–Alexandros, 1486.

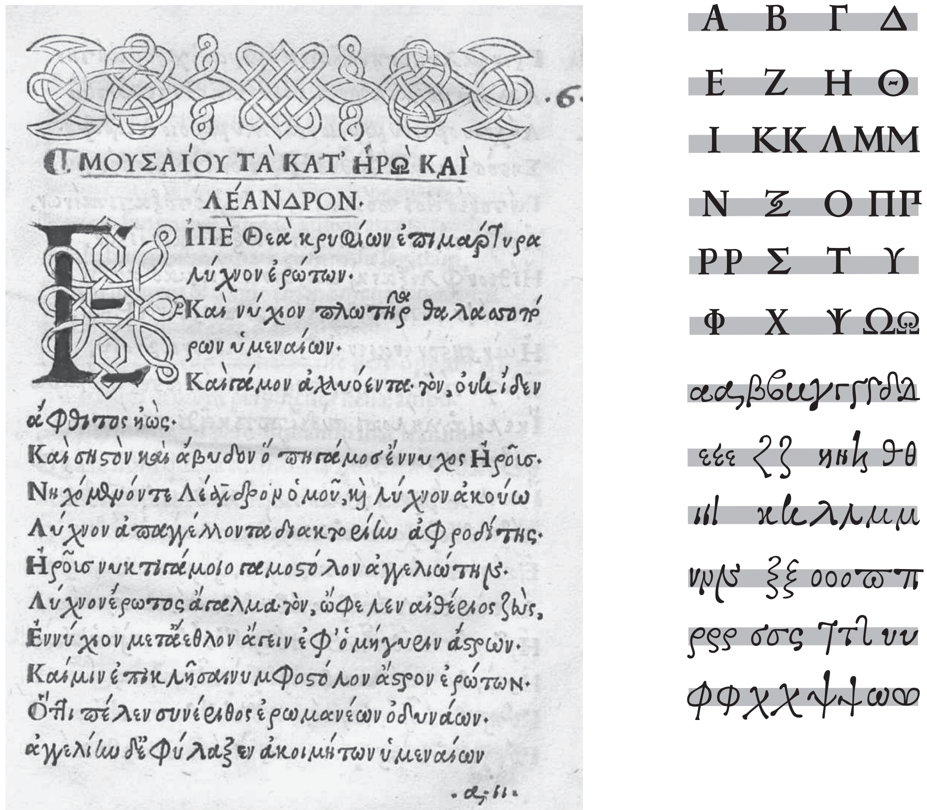


Plate 13: Musaeus, *De Hero & Leandro*, Venice, Francesco Griffo, Aldus Manutius, 1494.

Plate 14: *Etymologicum Magnum*, Venice, Nikolaos Vlastos–Zacharias Kalliergis, 1499.

Plate 15: Ioannis Xiphilinus, *Dionis Cassii Nicaei Romanae historiae*, Paris, Robert Estienne, 1551.

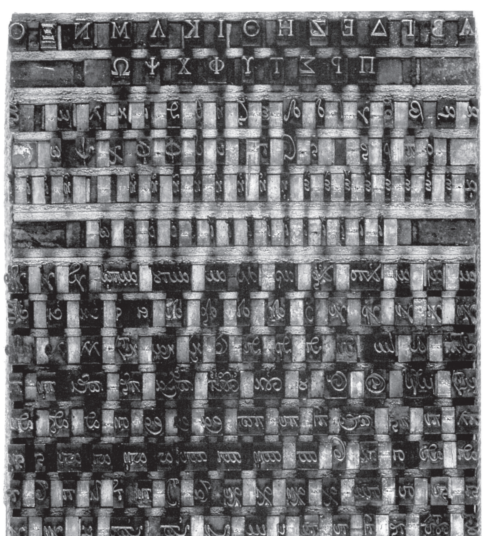
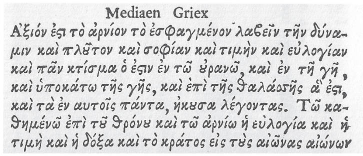
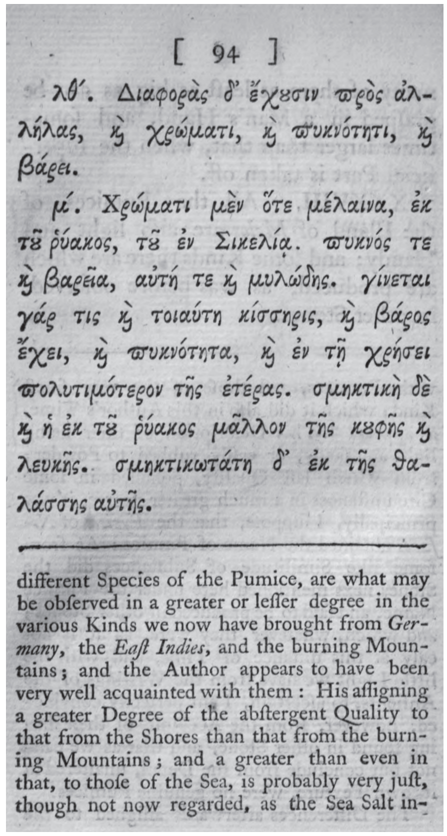


Plate 16, left: Greek font (size Parangon) by Robert Granjon, Antwerp, Christophe Plantin, 1565.
Plate 17, right: Greek font (size Garmont Grieks) by Christoffell van Dyck, Daniel Elsevier, 1681.


Mediaen Griex
Ἄξιόν ἐστι τὸ ἀρνίον τὸ ἐσφαγμένον λαβεῖν τὴν δύναμιν καὶ πλοῦτον καὶ σοφίαν καὶ τιμὴν καὶ εὐλογίαν
καὶ πᾶν κτίσμα ὅ ἐστιν ἐν τῷ οὐρανῷ, καὶ ἐν τῇ γῇ,
καὶ ὑποκάτω τῆς γῆς, καὶ ἐπὶ τῆς θαλάσσης ἅ ἐστι,
καὶ τὰ ἐν αὐτοῖς πάντα, ἤκουσα λέγοντας. Τῷ καθημένῳ ἐπὶ τοῦ θρόνου καὶ τῷ ἀρνίῳ ἡ εὐλογία καὶ ἡ
τιμὴ καὶ ἡ δόξα καὶ τὸ κράτος εἰς τοὺς αἰῶνας αἰώνων


[94]

λβʹ. Διαφορὰς δ' ἔχουσιν πρὸς ἀλλήλας, καὶ χρώματι, καὶ πυκνότητι, καὶ βάρει.

μʹ. Χρώματι μὲν ὅτε μέλαινα, ἐκ τοῦ ῥύακος, τοῦ ἐν Σικελίᾳ. πυκνός τε καὶ βαρεῖα, αὐτή τε καὶ μυλώδης. γίνεται γάρ τις καὶ τοιαύτη κίσσηρις, καὶ βάρος ἔχει, καὶ πυκνότητα, καὶ ἐν τῇ χρήσει πολυτιμότερον τῆς ἑτέρας. σμηκτικὴ δὲ καὶ ἡ ἐκ τοῦ ῥύακος μᾶλλον τῆς κεφῆς καὶ λευκῆς. σμηκτικωτάτη δ' ἐκ τῆς θαλάσσης αὐτῆς.

different Species of the Pumice, are what may be obſerved in a greater or leſſer degree in the various Kinds we now have brought from *Germany*, the *Eaſt Indies*, and the burning Mountains; and the Author appears to have been very well acquainted with them: His aſſigning a greater Degree of the abſtergent Quality to that from the Shores than that from the burning Mountains; and a greater than even in that, to thoſe of the Sea, is probably very juſt, though not now regarded, as the Sea Salt in-

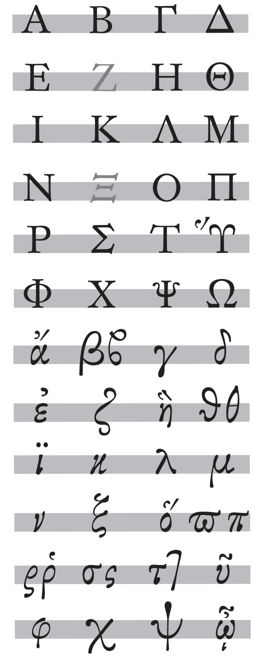
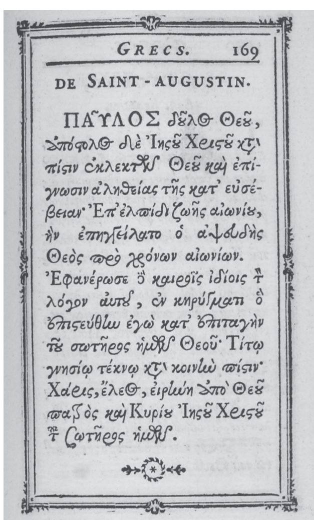


Plate 18, left: Greek font (size Mediaen Griex) by Nicholas Kis, Amsterdam, 1686.
Plate 19, right: John Hill, *Theophrastus*, London, William Caslon, 1774.


GRECS. 169

DE SAINT-AUGUSTIN.

ΠΑΥΛΟΣ δοῦλος Θεοῦ,
ἀπόστολος δὲ Ἰησοῦ Χριστοῦ κατὰ
πίστιν ἐκλεκτῶν Θεοῦ καὶ ἐπί-
γνωσιν ἀληθείας τῆς κατ' εὐσέ-
βειαν· Ἐπ' ἐλπίδι ζωῆς αἰωνίου,
ἣν ἐπηγγείλατο ὁ ἀψευδὴς
Θεὸς πρὸ χρόνων αἰωνίων.
Ἐφανέρωσε δὲ καιροῖς ἰδίοις τὸν
λόγον αὐτοῦ, ἐν κηρύγματι ὃ
ἐπιστεύθην ἐγὼ κατ' ἐπιταγὴν
τοῦ σωτῆρος ἡμῶν Θεοῦ· Τίτῳ
γνησίῳ τέκνῳ κατὰ κοινὴν πίστιν·
Χάρις, ἔλεος, εἰρήνη ἀπὸ Θεοῦ
πατρὸς καὶ Κυρίου Ἰησοῦ Χριστοῦ
τοῦ σωτῆρος ἡμῶν.

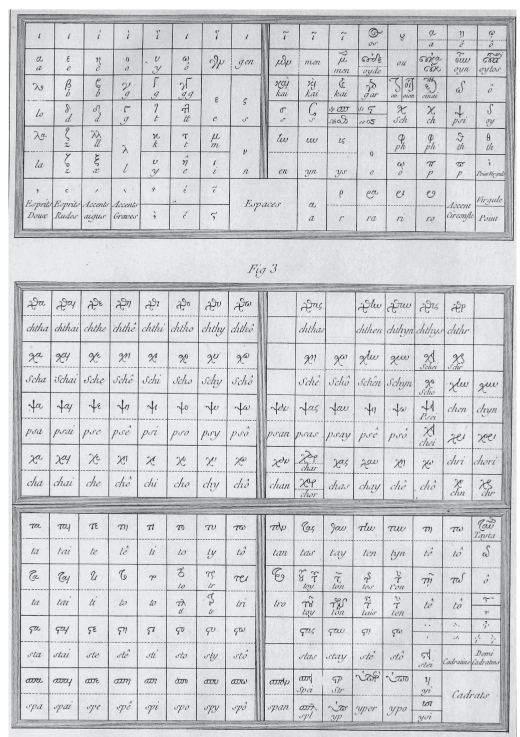


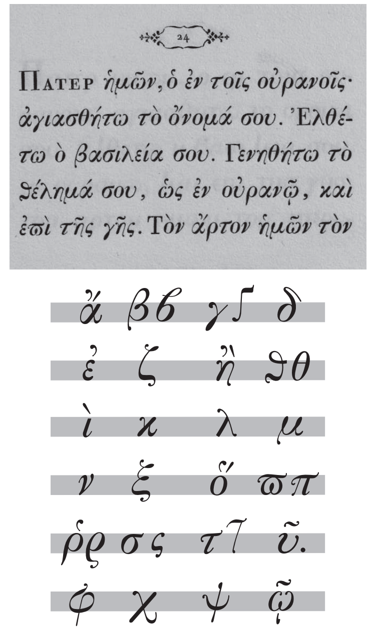
Plate 20, left: Greek font (size Saint Augustin) by Pierre Simon Fournier, *Manuel typographique*, Paris, 1764.
Plate 21, right: Greek typecases, Diderot–d'Alembert, *Encyclopédie...*, Paris, 1751.

ΤΟ ΤΗΣ

ΙΛΙΑΔΟΣ

Ρ Ω.

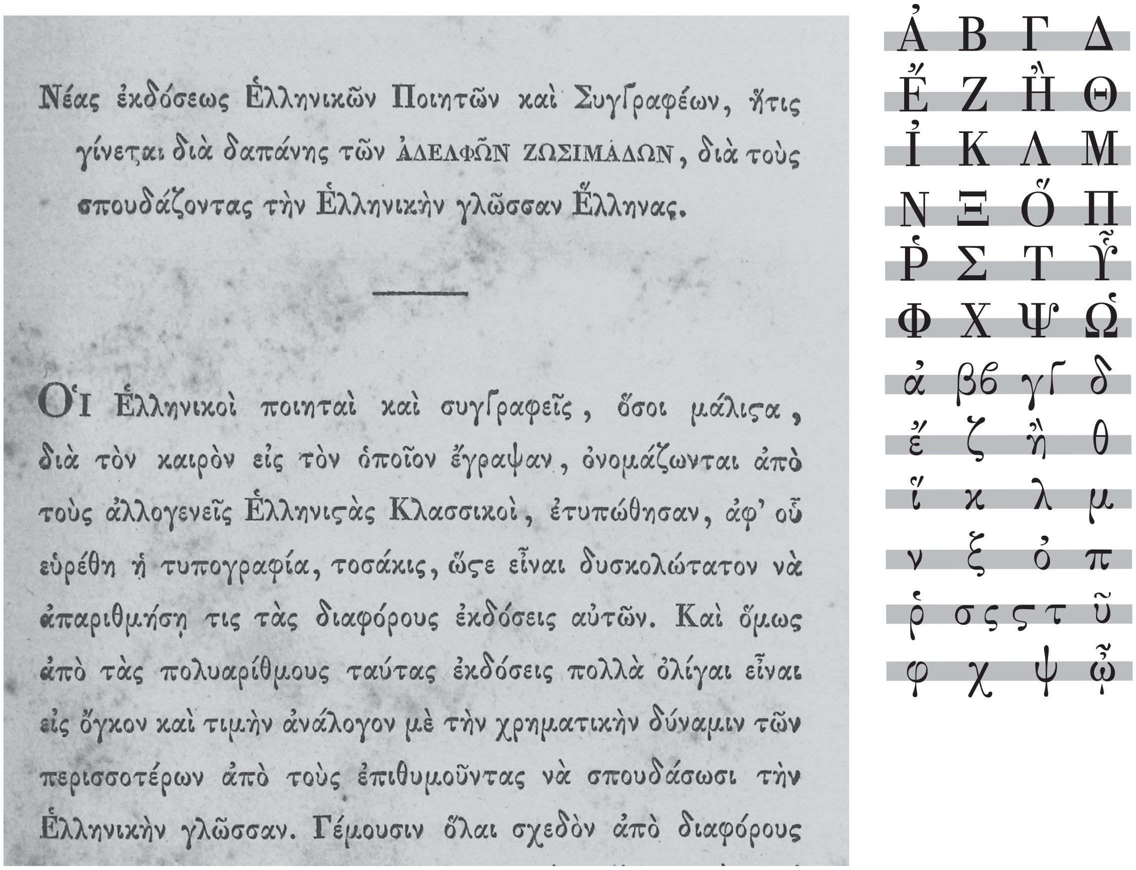
ΟΥ̓Δ' ἔλαθ' Ἀτρέος υἱὸν, ἀρηΐφιλον Μενέλαον,
Πάτροκλος Τρώεσσι δαμεὶς ἐν δηϊοτῆτι·
Βῆ δὲ διὰ προμάχων κεκορυθμένος αἴθοπι χαλκῷ·
Ἀμφὶ δ' ἄρ' αὐτῷ βαῖν', ὥς τις περὶ πόρτακι μήτηρ,
Πρωτοτόκος, κινυρὴ, οὐ πρὶν εἰδυῖα τόκοιο·
Ὣς περὶ Πατρόκλῳ βαῖνε ξανθὸς Μενέλαος·
Πρόσθε δέ οἱ δόρυ τ' ἔσχε καὶ ἀσπίδα πάντοσ' ἐΐσην,
Τὸν κτάμεναι μεμαὼς, ὅστις τοῦ γ' ἀντίος ἔλθοι.
Οὐδ' ἄρα Πάνθου υἱὸς ἐϋμμελίης ἀμέλησε
Πατρόκλοιο πεσόντος ἀμύμονος· ἄγχι δ' ἄρ' αὐτοῦ
Ἔστη, καὶ προσέειπεν ἀρηΐφιλον Μενέλαον·
Ἀτρεΐδη Μενέλαε, διοτρεφὲς, ὄρχαμε λαῶν,
Χάζεο, λεῖπε δὲ νεκρὸν, ἔα δ' ἔναρα βροτόεντα·
Οὐ γάρ τις πρότερος Τρώων κλειτῶν τ' ἐπικούρων
Πάτροκλον βάλε δουρὶ κατὰ κρατερὴν ὑσμίνην·



24

ΠΑΤΕΡ ἡμῶν, ὁ ἐν τοῖς οὐρανοῖς· ἁγιασθήτω τὸ ὄνομά σου. Ἐλθέτω ὁ βασιλεία σου. Γενηθήτω τὸ θέλημά σου, ὡς ἐν οὐρανῷ, καὶ ἐπὶ τῆς γῆς. Τὸν ἄρτον ἡμῶν τὸν

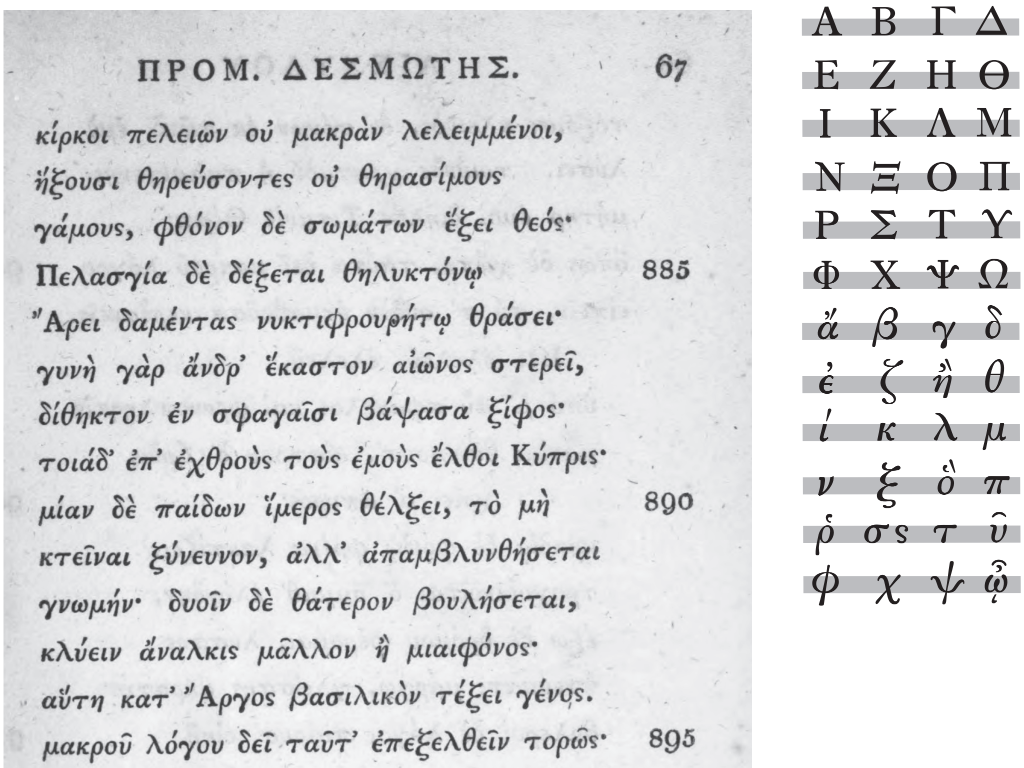
Plate 22, left: Greek font by Alexander Wilson, *Homer's Iliad*, Glasgow, Foulis Bros., 1755.
Plate 23, right: Greek font by Giambattista Bodoni, Parma, Greco 24–Testo/15–16p.



Νέας ἐκδόσεως Ἑλληνικῶν Ποιητῶν καὶ Συγγραφέων, ἥτις γίνεται διὰ δαπάνης τῶν ἈΔΕΛΦΩΝ ΖΩΣΙΜΑΔΩΝ, διὰ τοὺς σπουδάζοντας τὴν Ἑλληνικὴν γλῶσσαν Ἕλληνας.

ΟἹ Ἑλληνικοὶ ποιηταὶ καὶ συγγραφεῖς, ὅσοι μάλιστα, διὰ τὸν καιρὸν εἰς τὸν ὁποῖον ἔγραψαν, ὀνομάζωνται ἀπὸ τοὺς ἀλλογενεῖς Ἑλληνιστὰς Κλασσικοὶ, ἐτυπώθησαν, ἀφ' οὗ εὑρέθη ἡ τυπογραφία, τοσάκις, ὥστε εἶναι δυσκολώτατον νὰ ἀπαριθμήσῃ τις τὰς διαφόρους ἐκδόσεις αὐτῶν. Καὶ ὅμως ἀπὸ τὰς πολυαρίθμους ταύτας ἐκδόσεις πολλὰ ὀλίγαι εἶναι εἰς ὄγκον καὶ τιμὴν ἀνάλογον μὲ τὴν χρηματικὴν δύναμιν τῶν περισσοτέρων ἀπὸ τοὺς ἐπιθυμοῦντας νὰ σπουδάσωσι τὴν Ἑλληνικὴν γλῶσσαν. Γέμουσιν ὅλαι σχεδὸν ἀπὸ διαφόρους

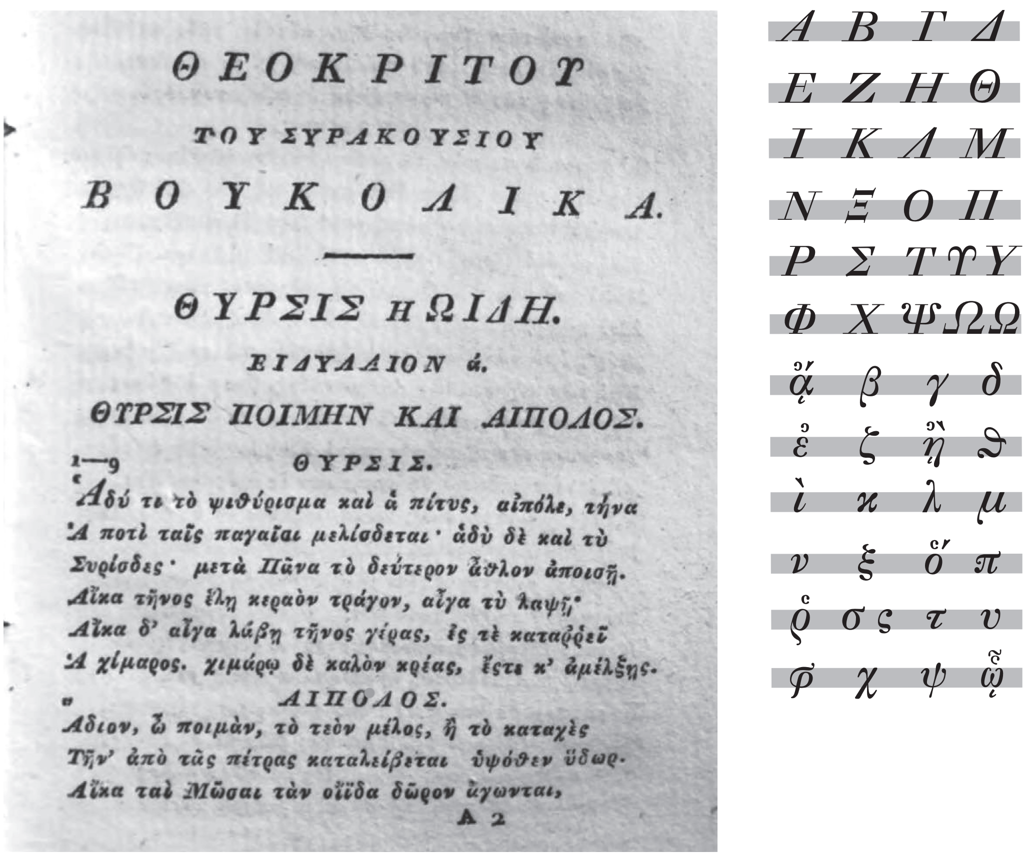
Plate 24: Greek font by Firmin Didot, Ad. Korai, *Πρόδρομος της Ελληνικής Βιβλιοθήκης*, Paris, 1805.



ΠΡΟΜ. ΔΕΣΜΩΤΗΣ. 67

κίρκοι πελειῶν οὐ μακρὰν λελειμμένοι,
ἥξουσι θηρεύσοντες οὐ θηρασίμους
γάμους, φθόνον δὲ σωμάτων ἕξει θεός·
Πελασγία δὲ δέξεται θηλυκτόνῳ
Ἄρει δαμέντας νυκτιφρουρήτῳ θράσει·
γυνὴ γὰρ ἄνδρ' ἕκαστον αἰῶνος στερεῖ,
δίθηκτον ἐν σφαγαῖσι βάψασα ξίφος·
τοιάδ' ἐπ' ἐχθροὺς τοὺς ἐμοὺς ἔλθοι Κύπρις·
μίαν δὲ παίδων ἵμερος θέλξει, τὸ μὴ
κτεῖναι ξύνευνον, ἀλλ' ἀπαμβλυνθήσεται
γνωμήν· δυοῖν δὲ θάτερον βουλήσεται,
κλύειν ἄναλκις μᾶλλον ἢ μιαιφόνος·
αὕτη κατ' Ἄργος βασιλικὸν τέξει γένος.
μακροῦ λόγου δεῖ ταῦτ' ἐπεξελθεῖν τορῶς·

Plate 25: Greek font after Richard Porson's hand, Charles James Blomfield, *Aeschyli Prometheus vinctus...*, Cambridge U.P., 1810.



ΘΕΟΚΡΙΤΟΥ

ΤΟΥ ΣΥΡΑΚΟΥΣΙΟΥ

ΒΟΥΚΟΛΙΚΑ.

ΘΥΡΣΙΣ Η ΩΙΔΗ.

ΕΙΔΥΛΛΙΟΝ α΄.

ΘΥΡΣΙΣ ΠΟΙΜΗΝ ΚΑΙ ΑΙΠΟΛΟΣ.

1—9 ΘΥΡΣΙΣ.

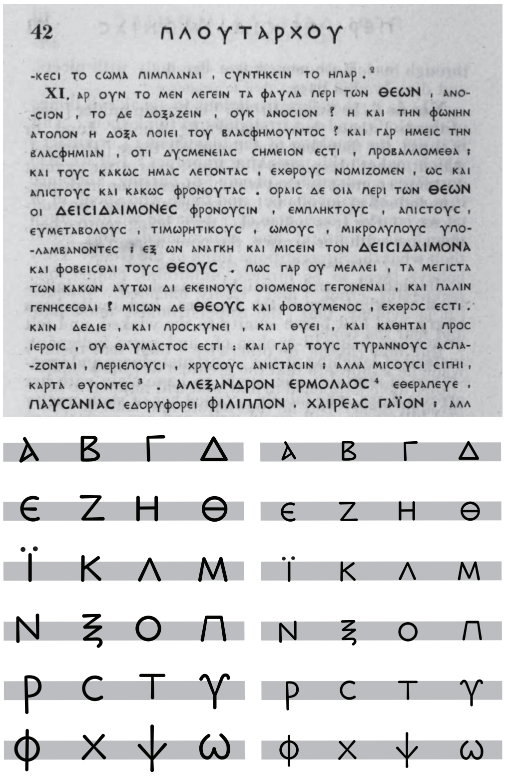
Ἁδύ τι τὸ ψιθύρισμα καὶ ἁ πίτυς, αἰπόλε, τήνα
Ἁ ποτὶ ταῖς παγαῖσι μελίσδεται· ἁδὺ δὲ καὶ τὺ
Συρίσδες· μετὰ Πᾶνα τὸ δεύτερον ἆθλον ἀποισῇ.
Αἴκα τῆνος ἕλῃ κεραὸν τράγον, αἶγα τὺ λαψῇ·
Αἴκα δ' αἶγα λάβῃ τῆνος γέρας, ἐς τὲ καταῤῥεῖ
Ἁ χίμαρος. χιμάρῳ δὲ καλὸν κρέας, ἔστε κ' ἀμέλξῃς.

ΑΙΠΟΛΟΣ.

Ἅδιον, ὦ ποιμὰν, τὸ τεὸν μέλος, ἢ τὸ καταχὲς
Τῆν' ἀπὸ τᾶς πέτρας καταλείβεται ὑψόθεν ὕδωρ.
Αἴκα ταὶ Μῶσαι τὰν οἶδα δῶρον ἄγωνται,

Α 2

Plate 26: Greek font by Karl Tauchnitz, G. Schäfer, Theocritus, *Bion et Moschus: ad optimorum librorum fidem...*, Leipzig, 1809.



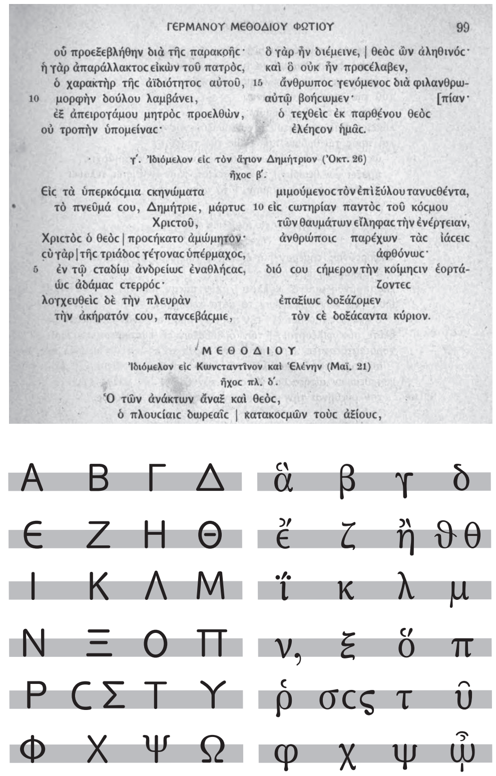
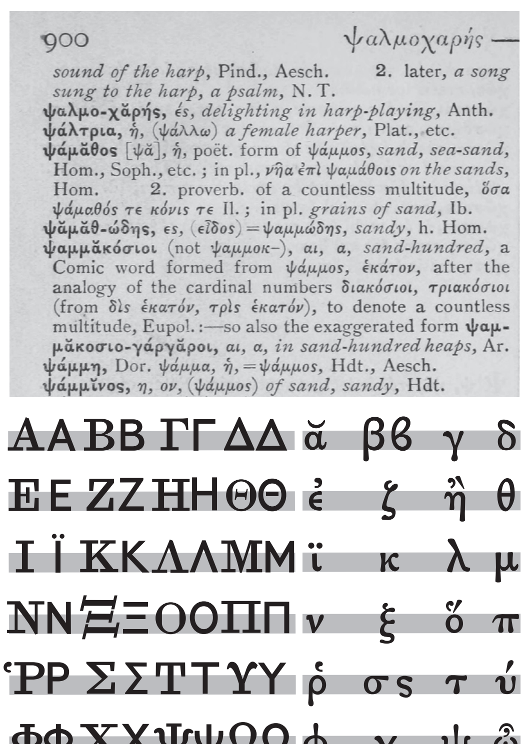


Plate 27, left: Greek font by Julian Hibbert, *Plutarchus and Theophrastus on Superstition*, London, 1828.
Plate 28, right: Greek font by Maurice Eduard Pinder, W. Christ–M.K Paranikas, *Anthologia Graeca...*, Leipzig, B.G. Teubner, 1866.



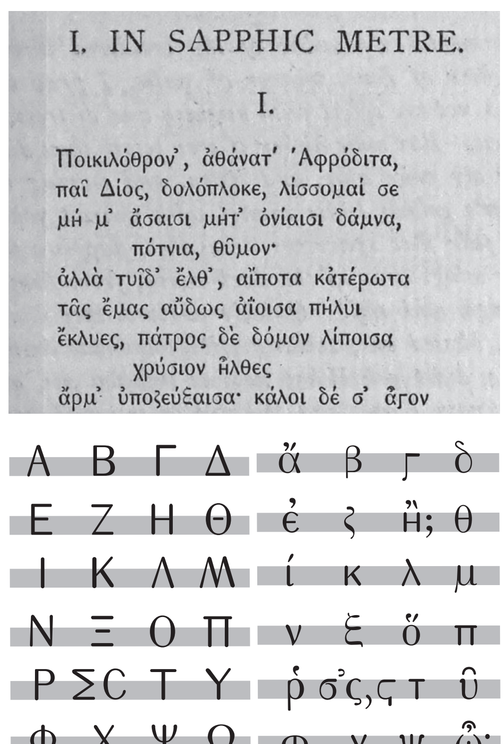
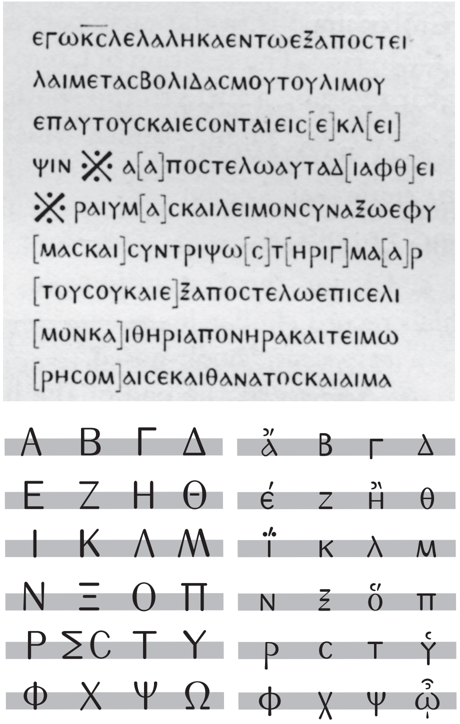


Plate 29, left: Bold Greek font by the foundries Figgins and Miller & Richard, mid-19th c.
Plate 30, right: Greek font by R. Decker, H.T. Wharton (ed.), *Sappho...*, London, David Stott, 1887.



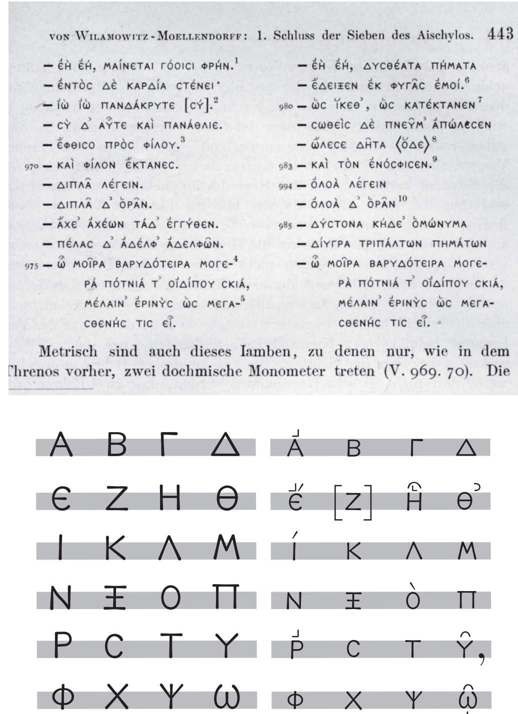


Plate 31, left: Greek font Klassiker Griechich by R. Decker, B.P. Grenfell, *An Alexandrian Erotic Fragment. . .*, Oxford, 1896.

Plate 32, right: Greek font for the Prussian Royal Library, *Sitzungberichte–Jahrgang* 1903, Berlin, 1903.

Plate 33, left: Greek font Mediäval, Genzsch & Heyse foundry, Hamburg, 1884(?).
Plate 34, center: Greek font by Selwyn Image for Macmillan Publishers, London, 1897.
Plate 35, right: Greek font Otter Greek by Robert Proctor, London, 1903.

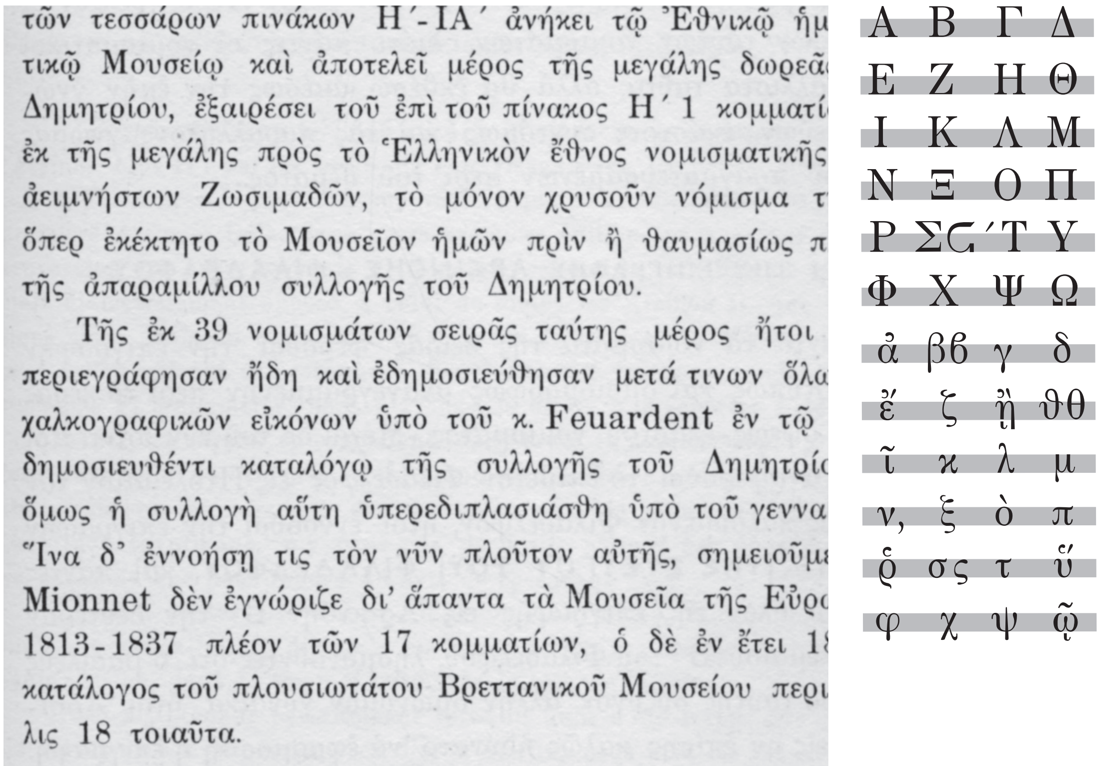
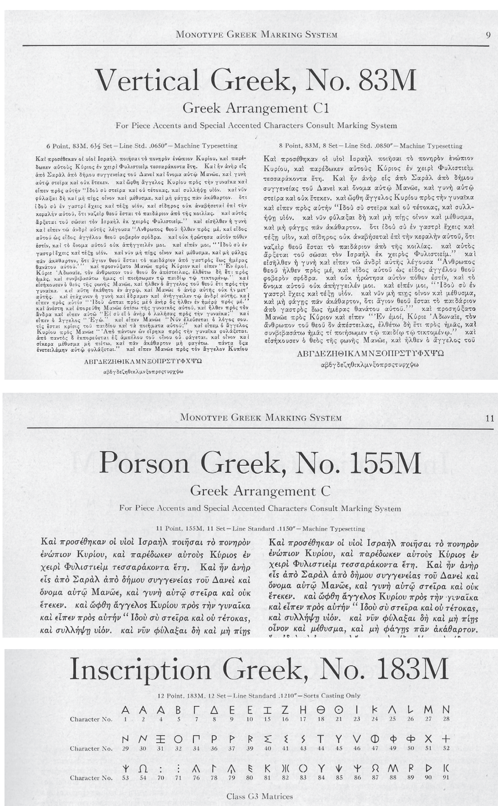


Plate 36: Greek font, Schelter & Giesecke AG, Leipzig, >1895.



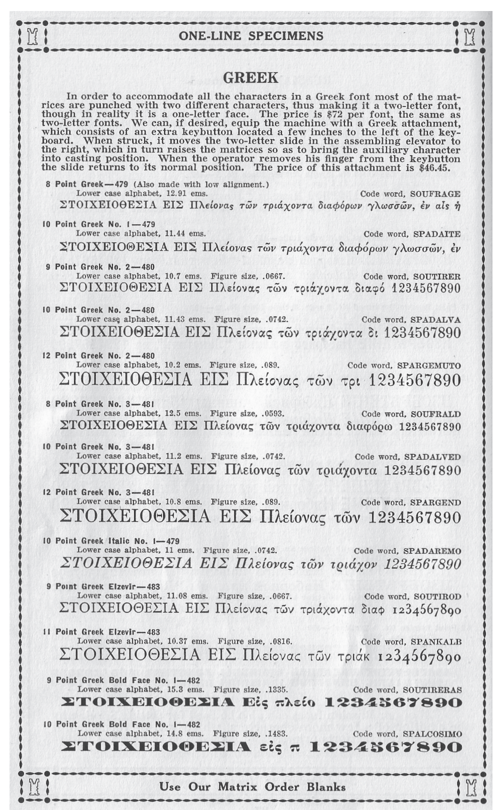


Plate 37, left: Lanston Monotype, US, 1910–1930.
Plate 38, right: Linotype, US, 1910–1930.

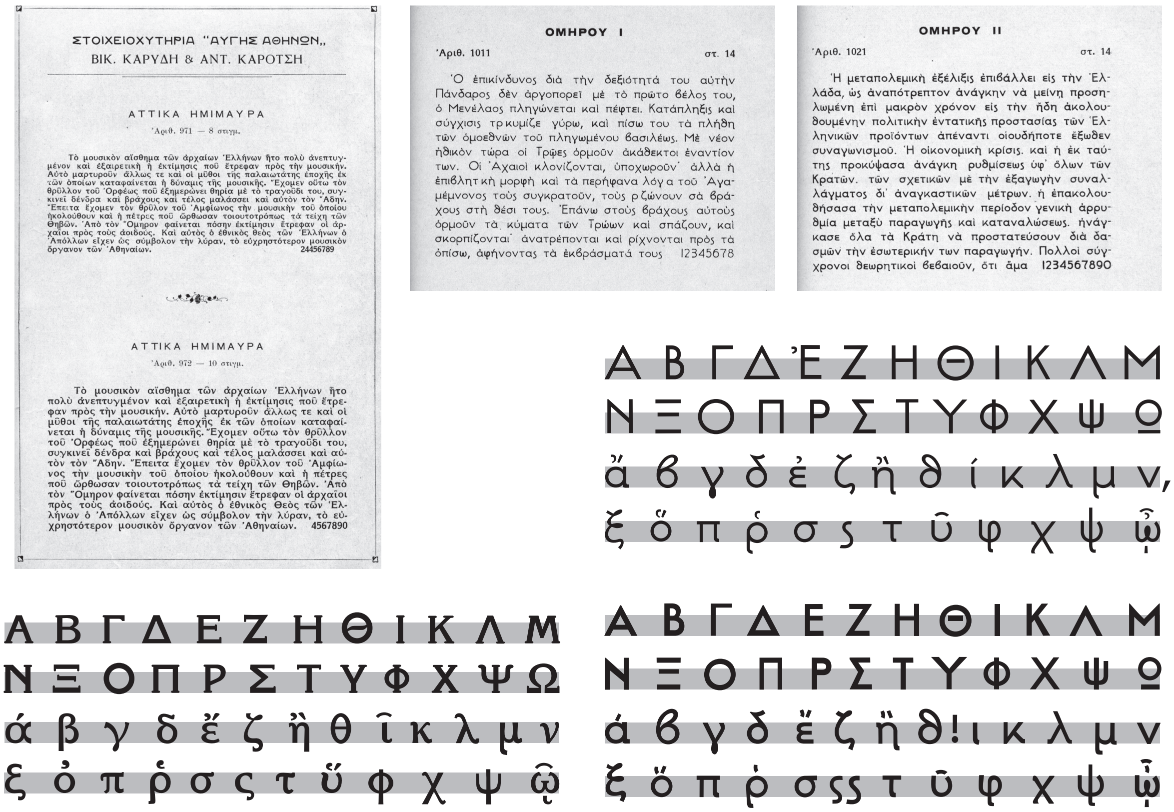


Plate 39: Greek type foundry A. Karotsis–V. Karidis, Athens, 1920s–1950s.

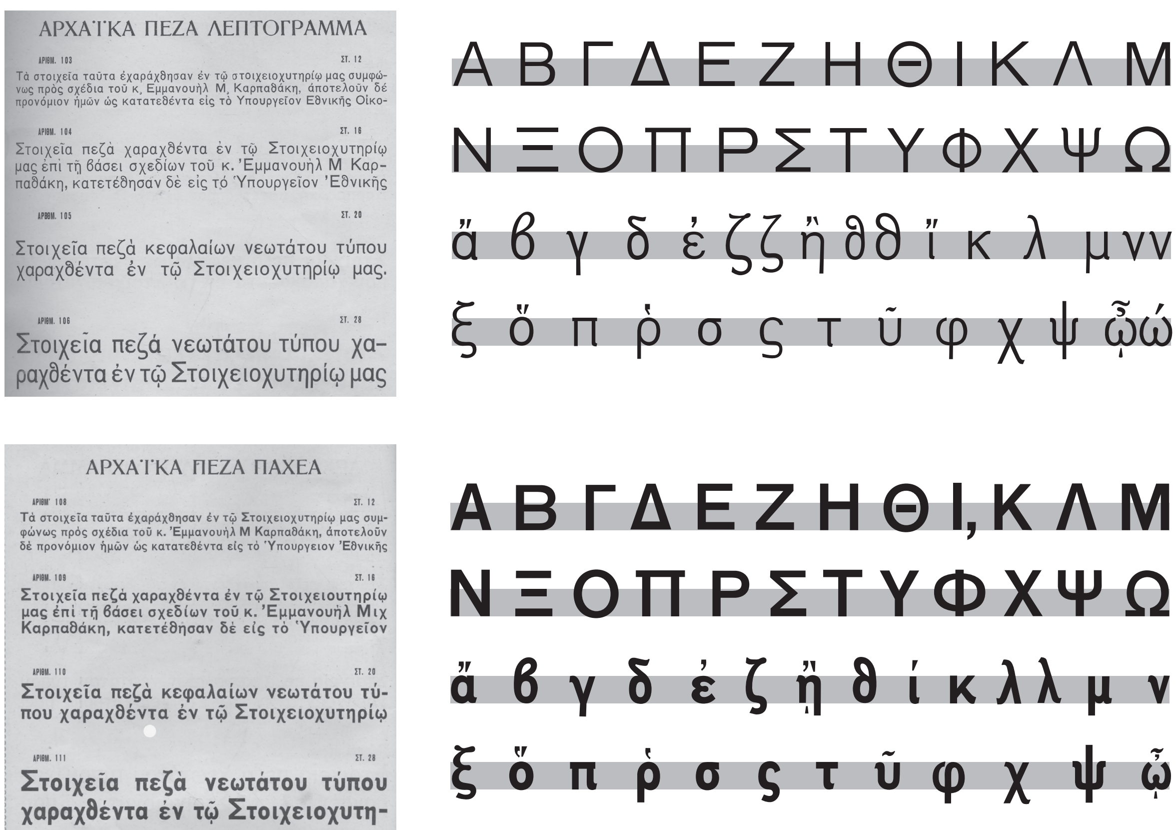
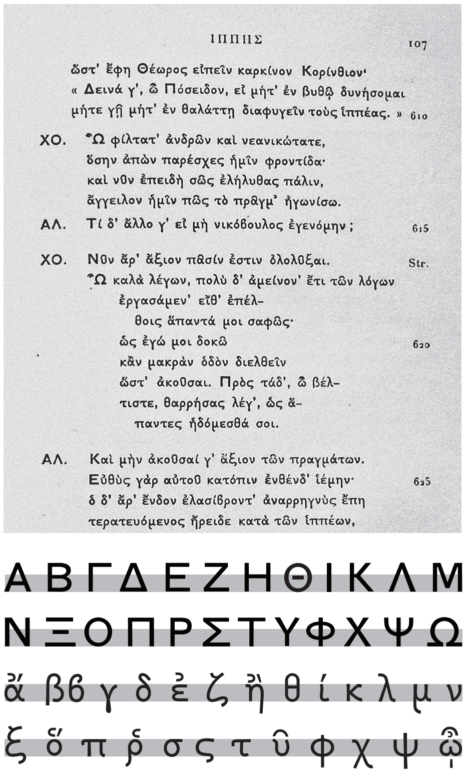


Plate 40: Greek type foundry Em. Karpathakis, Athens, 1920s–1960s. Right: Greek Archaic Light & Bold, 1920s.



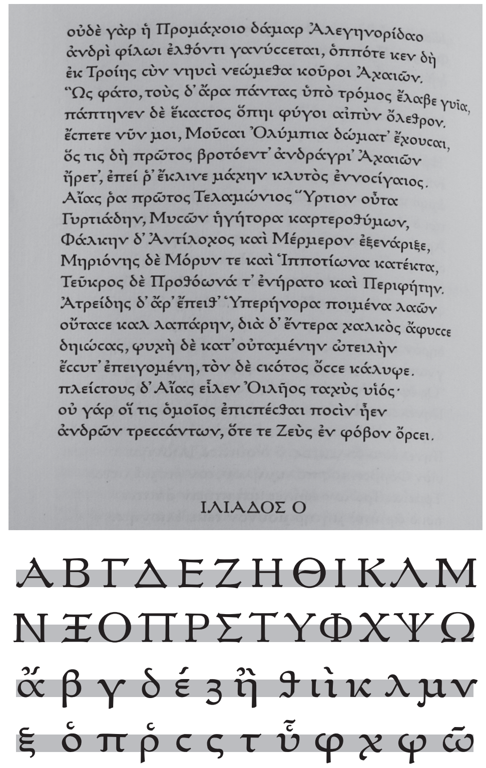


Plate 41, left: Greek font for the publisher Les Belles Lettres, Paris, 1920s.
Plate 42, right: Greek font by Willi Wiegand for the publisher Bremer Presse, Bremen, 1923.

Plate 43, left: New Hellenic 192, Monotype Ltd, 1927.
Plate 44, right: Greek Perpetua 283 by Eric Gill, Monotype Ltd, 1928.

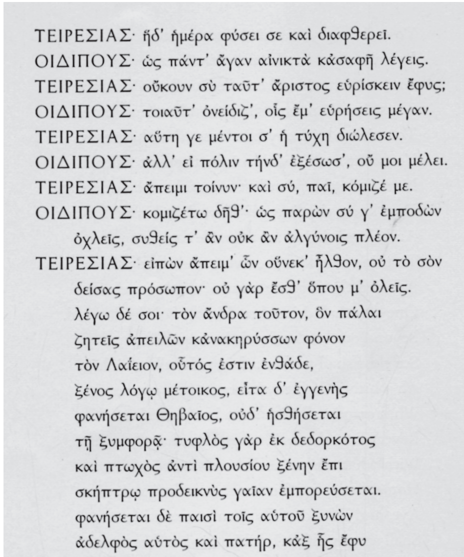
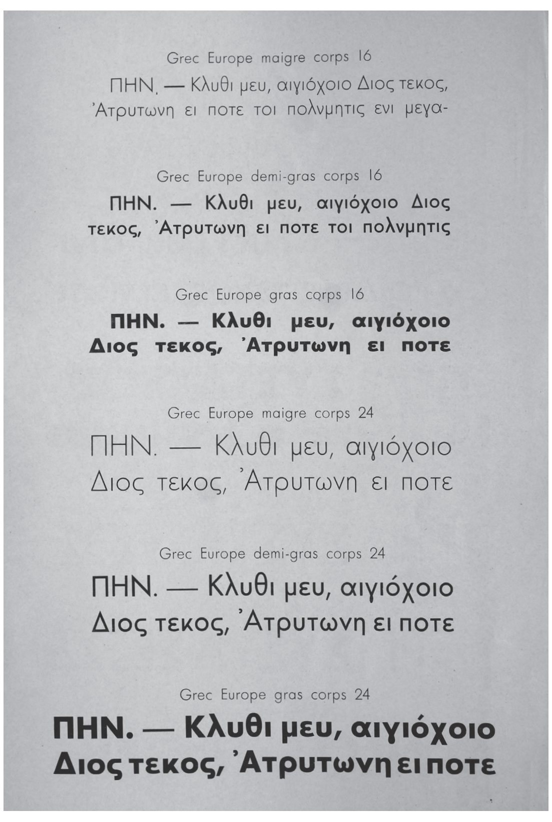

ΤΕΙΡΕΣΙΑΣ· ἥδ' ἡμέρα φύσει σε καὶ διαφθερεῖ.
ΟΙΔΙΠΟΥΣ· ὡς πάντ' ἄγαν αἰνικτὰ κἀσαφῆ λέγεις.
ΤΕΙΡΕΣΙΑΣ· οὔκουν σὺ ταῦτ' ἄριστος εὑρίσκειν ἔφυς;
ΟΙΔΙΠΟΥΣ· τοιαῦτ' ὀνείδιζ', οἷς ἔμ' εὑρήσεις μέγαν.
ΤΕΙΡΕΣΙΑΣ· αὕτη γε μέντοι σ' ἡ τύχη διώλεσεν.
ΟΙΔΙΠΟΥΣ· ἀλλ' εἰ πόλιν τήνδ' ἐξέσωσ', οὔ μοι μέλει.
ΤΕΙΡΕΣΙΑΣ· ἄπειμι τοίνυν· καὶ σύ, παῖ, κόμιζέ με.
ΟΙΔΙΠΟΥΣ· κομιζέτω δῆθ'· ὡς παρὼν σύ γ' ἐμποδὼν
ὀχλεῖς, συθείς τ' ἂν οὐκ ἂν ἀλγύνοις πλέον.
ΤΕΙΡΕΣΙΑΣ· εἰπὼν ἄπειμ' ὧν οὕνεκ' ἦλθον, οὐ τὸ σὸν
δείσας πρόσωπον· οὐ γὰρ ἔσθ' ὅπου μ' ὀλεῖς.
λέγω δέ σοι· τὸν ἄνδρα τοῦτον, ὃν πάλαι
ζητεῖς ἀπειλῶν κἀνακηρύσσων φόνον
τὸν Λαΐειον, οὗτός ἐστιν ἐνθάδε,
ξένος λόγῳ μέτοικος, εἶτα δ' ἐγγενὴς
φανήσεται Θηβαῖος, οὐδ' ἡσθήσεται
τῇ ξυμφορᾷ· τυφλὸς γὰρ ἐκ δεδορκότος
καὶ πτωχὸς ἀντὶ πλουσίου ξένην ἔπι
σκήπτρῳ προδεικνὺς γαῖαν ἐμπορεύσεται.
φανήσεται δὲ παισὶ τοῖς αὑτοῦ ξυνὼν
ἀδελφὸς αὑτὸς καὶ πατήρ, κἀξ ἧς ἔφυ

Plate 45, left: Antigone Greek by Jan van Krimpen, Enschedé, 1927.
Plate 46, right: Romulus Greek by Jan van Krimpen, Monotype Ltd, 1928.


Grec Europe maigre corps 16
ΠΗΝ. — Κλυθι μευ, αιγιόχοιο Διος τεκος,
Ἀτρυτωνη ει ποτε τοι πολνμητις ενι μεγα-

Grec Europe demi-gras corps 16
ΠΗΝ. — Κλυθι μευ, αιγιόχοιο Διος
τεκος, Ἀτρυτωνη ει ποτε τοι πολνμητις

Grec Europe gras corps 16
ΠΗΝ. — Κλυθι μευ, αιγιόχοιο
Διος τεκος, Ἀτρυτωνη ει ποτε

Grec Europe maigre corps 24
ΠΗΝ. — Κλυθι μευ, αιγιόχοιο
Διος τεκος, Ἀτρυτωνη ει ποτε

Grec Europe demi-gras corps 24
ΠΗΝ. — Κλυθι μευ, αιγιόχοιο
Διος τεκος, Ἀτρυτωνη ει ποτε

Grec Europe gras corps 24
ΠΗΝ. — Κλυθι μευ, αιγιόχοιο
Διος τεκος, Ἀτρυτωνη ει ποτε

Plate 47: Grec Europe, Deberny & Peignot, 1930.

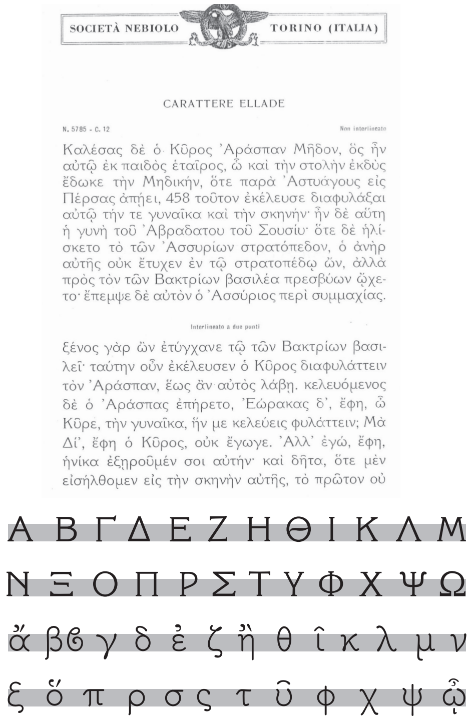


Plate 48, left: Ellade, Societa Augusta/Nebiolo, 1920s.
Plate 49, right: Metrolite Greek, Mergenthaler Linotype Corp., 1940s.

Plate 50: Olympia Greek (regular, medium, bold), type foundry Em. Karpathakis, Athens, 1939.

Plate 51: Greek Gill Sans (regular, bold, inclined), Monotype Ltd, 1950.

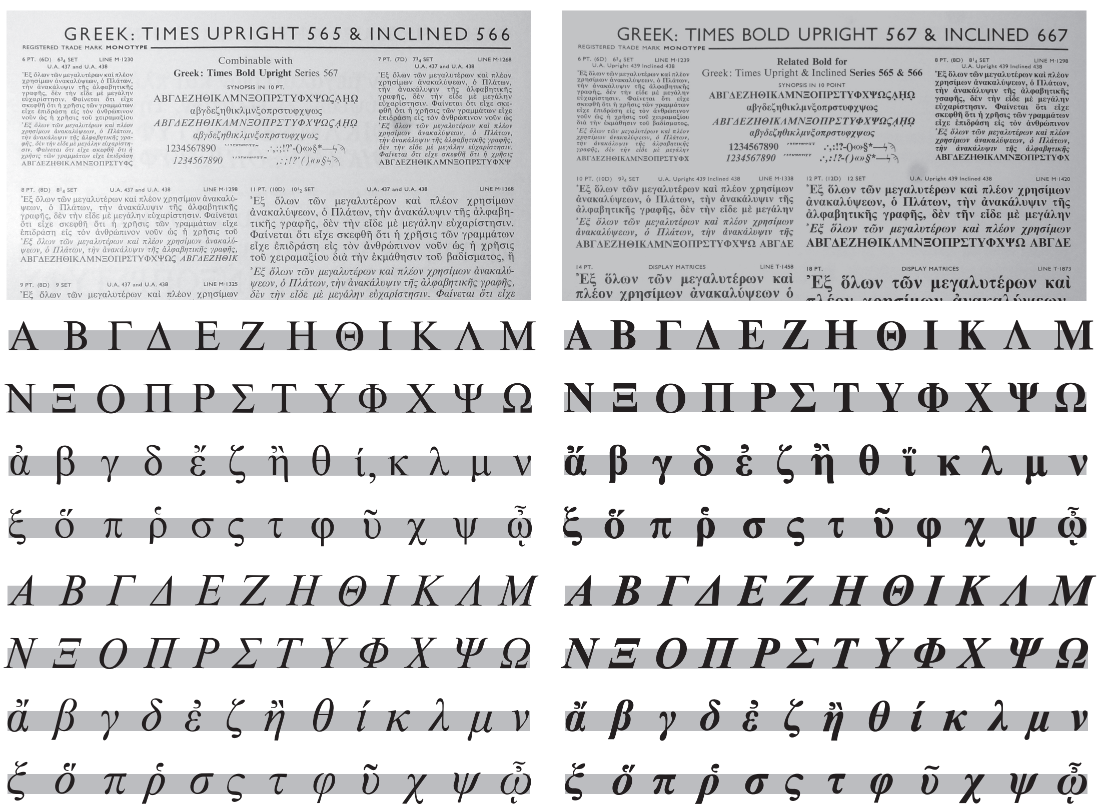


Plate 52: Greek Times, Monotype Ltd, 1950s.

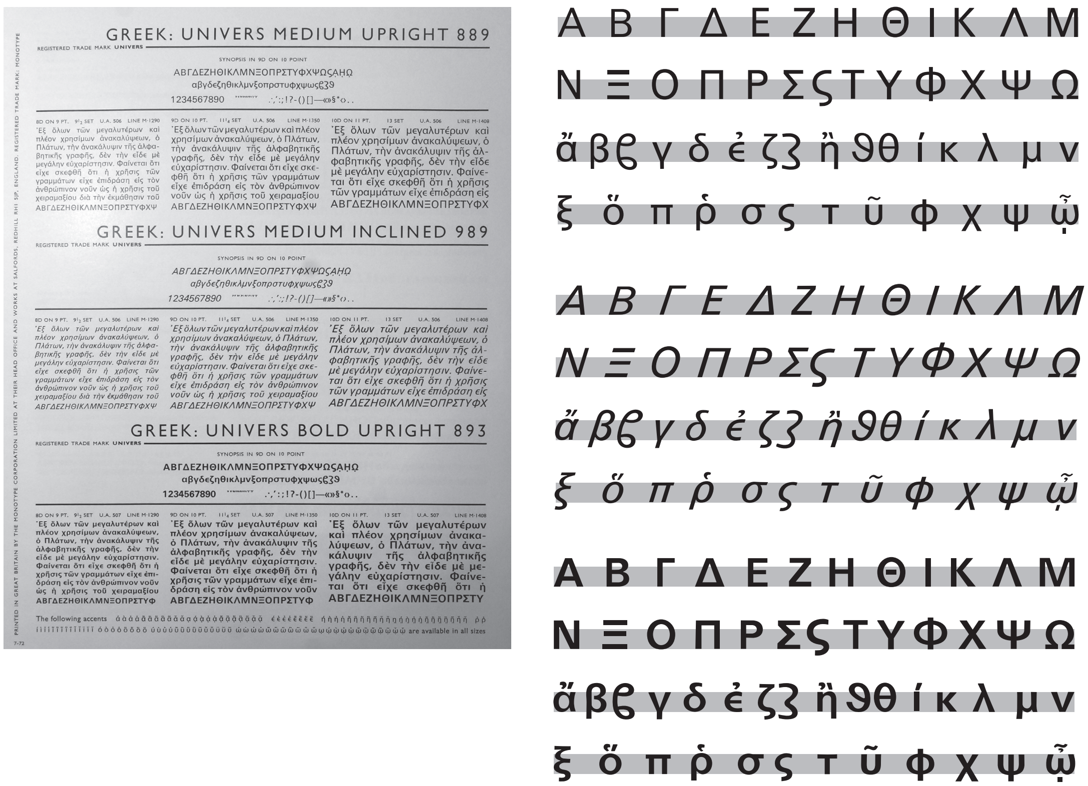


Plate 53: Greek Univers, Monotype Ltd, 1967

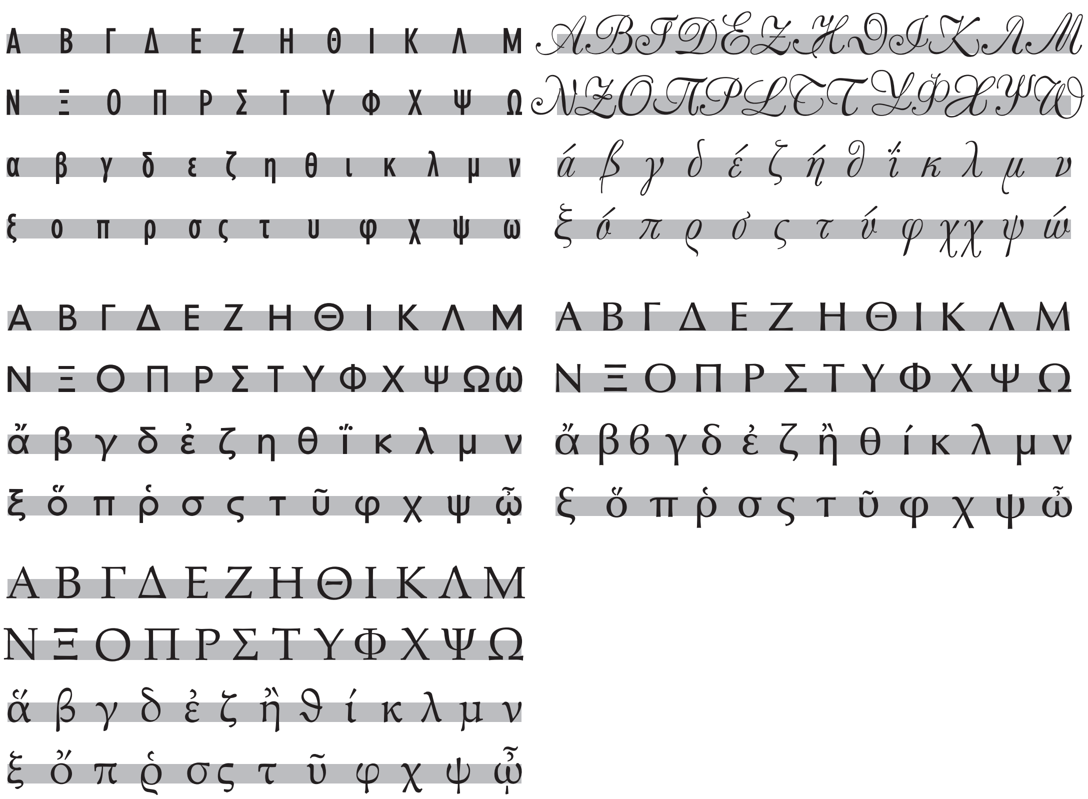


Plate 54: Greek types by Hermann Zapf, for J.D. Stempel foundry, 1950s. Top left: Artemis Griechische Enge Halbfette, 1953. Middle left: Attika Griechische, 1953. Bottom left: Heraklit Griechische, 1954. Top right: Frederika Griechische, 1953. Bottom right: Optima Greek, Mergenthaler Linotype Co., 1973.

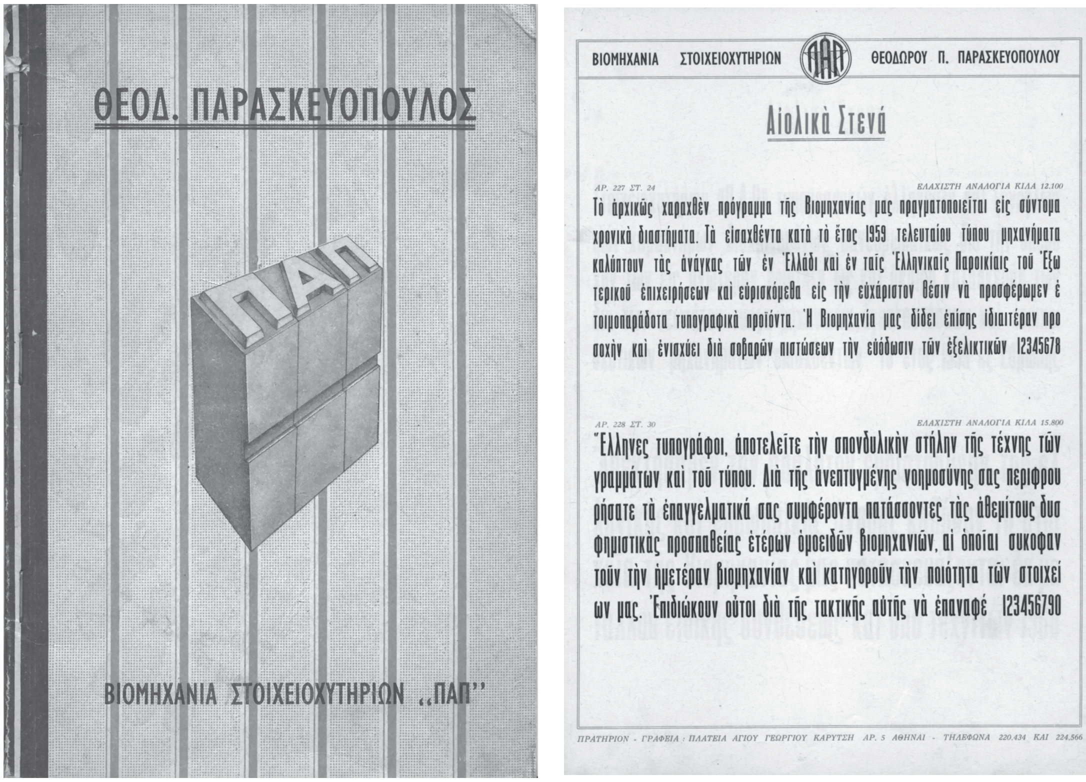


Plate 55: Type specimen from the Th. Paraskevopoulos (PAP) type foundry, 1950s.

Plate 56: Type specimens from the I. Sarasitis (Victoria) type foundry, 1960s. Right: Astir Greek, 1965.

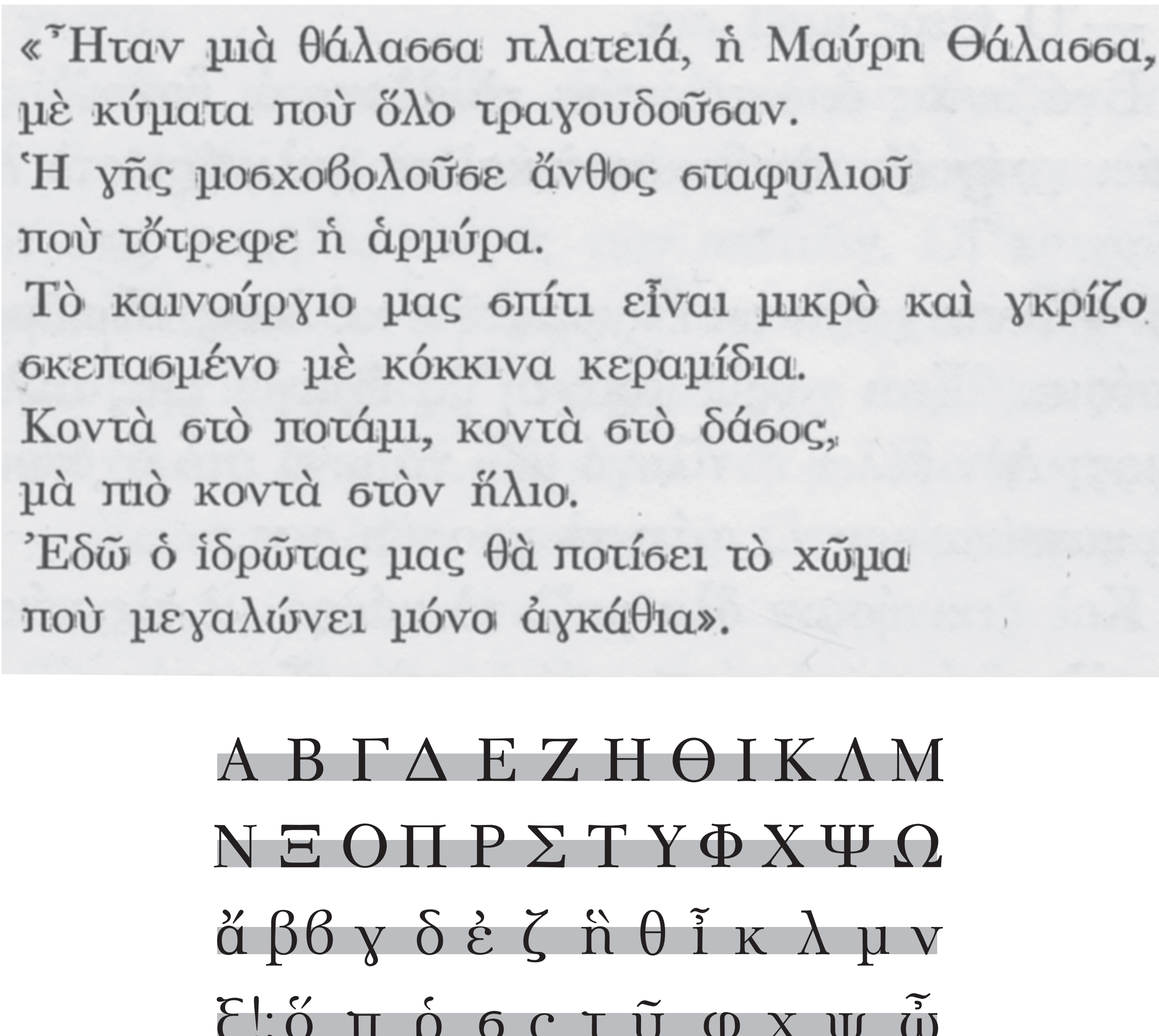

«Ἦταν μιὰ θάλασσα πλατειά, ἡ Μαύρη Θάλασσα,
μὲ κύματα ποὺ ὅλο τραγουδοῦσαν.
Ἡ γῆς μοσχοβολοῦσε ἄνθος σταφυλιοῦ
ποὺ τὄτρεφε ἡ ἁρμύρα.
Τὸ καινούργιο μας σπίτι εἶναι μικρὸ καὶ γκρίζο
σκεπασμένο μὲ κόκκινα κεραμίδια.
Κοντὰ στὸ ποτάμι, κοντὰ στὸ δάσος,
μὰ πιὸ κοντὰ στὸν ἥλιο.
Ἐδῶ ὁ ἱδρῶτας μας θὰ ποτίσει τὸ χῶμα
ποὺ μεγαλώνει μόνο ἀγκάθια».

Plate 57: Caledonia Greek, Mergenthaler Linotype Co., 1967.

Plate 58: Greek fonts by Matthew Carter for phototypesetting, Mergenthaler Linotype Co.
Left: Helvetica Greek, 1972; right: Century Schoolbook Greek, 1976.

Plate 59: Letter-transfer Greek fonts by Letraset and Mecanorma.

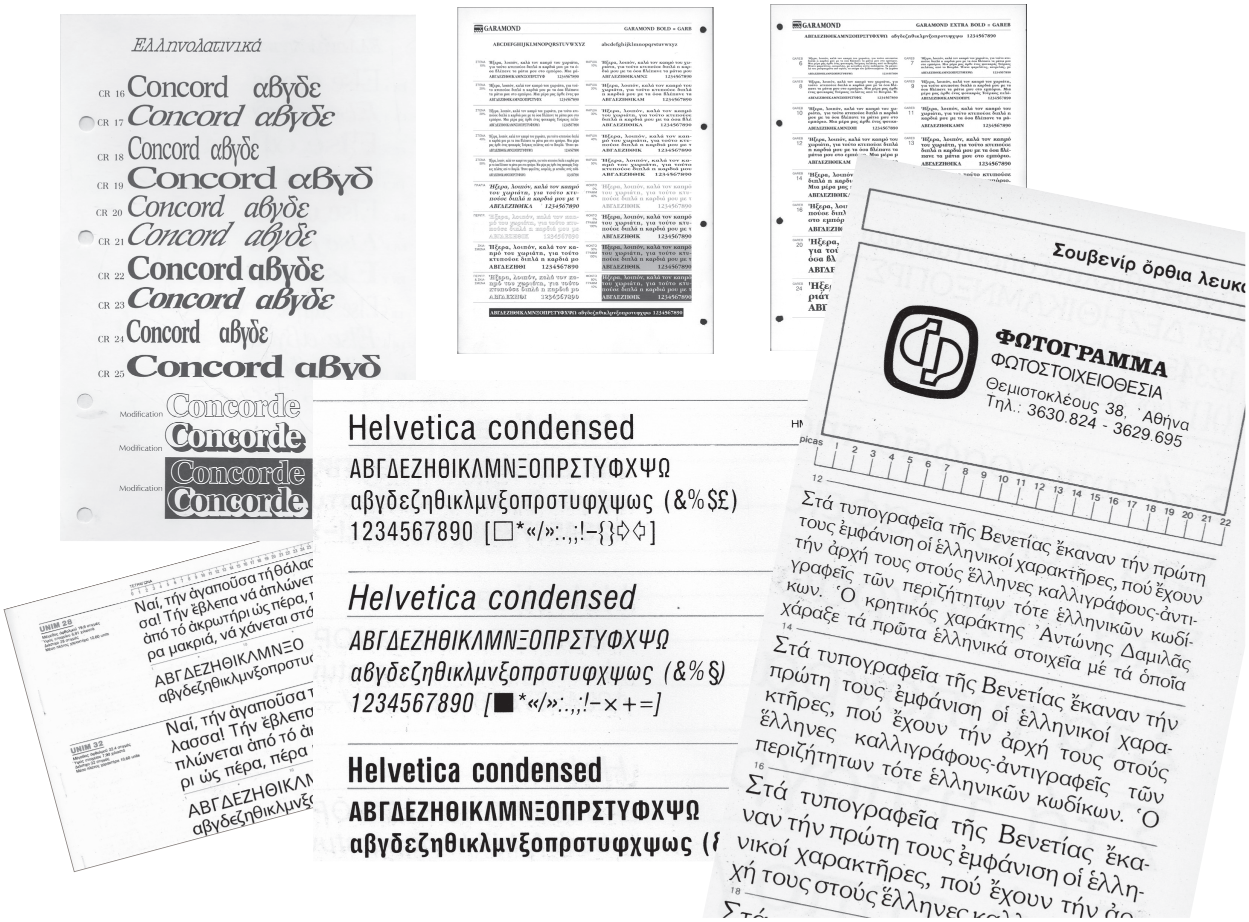


Plate 60: Phototypesetting specimen sheets of Greek fonts, 1970s–1980s.

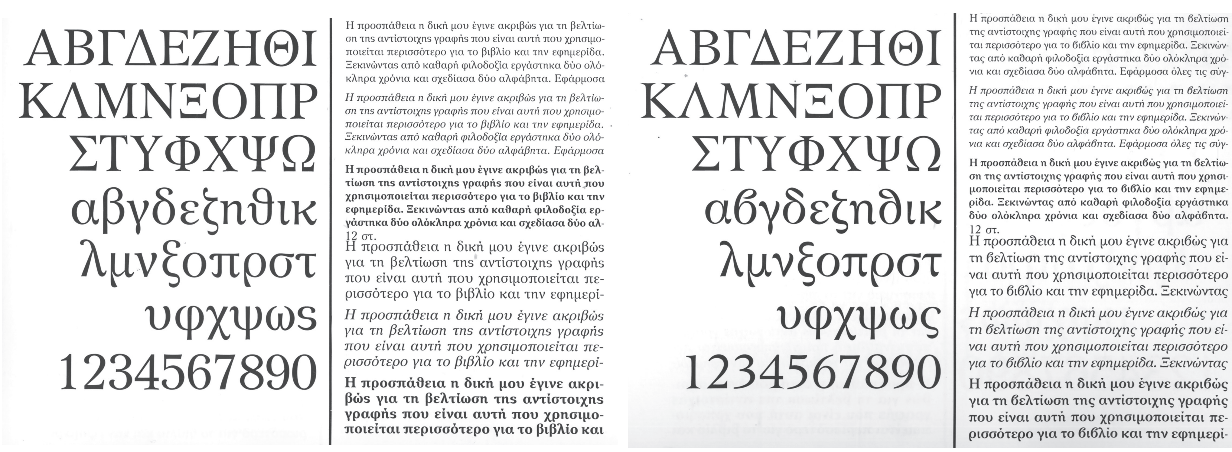


Plate 61: Two Greek typefaces by Takis Katsoulidis for AGFA SA, 1988–1990.
Left: Appolonia Greek; right: Katsoulidis Greek.

Towards an inventory of old print characters: Ungler's *Rubricella*, a case study

Janusz S. Bień

Abstract

The goal of the paper is twofold. One goal is to present a minor contribution to the task of describing the character inventory of a 16th century Polish printing house; for this purpose it is necessary to present also some theoretical aspects of the task. Another one is to discuss the use of XeLaTeX and other tools in the author's workflow.

1 Introduction

The idea to create the inventory of characters for Polish early printing houses dates back to 1920. It was proposed by Ludwik Bernacki [6],[1] probably inspired by Konrad Haebler's *Typenrepertorium der Wiegendrucke*, which started to appear in 1905.[2]

The 12 volumes of the series entitled *Polonia Typographica Saeculi Sedecim* published in 1936–1981 were an attempt to implement this concept. According to [14], the layout of the tables followed the publications of *Gesellschaft für Typenkunde*.[3] Since volume III, the fonts were documented not only with text samples, but also with tables of characters (Fig. 1); reportedly they were created by cutting out the characters from copies with a razor blade and pasting them together.

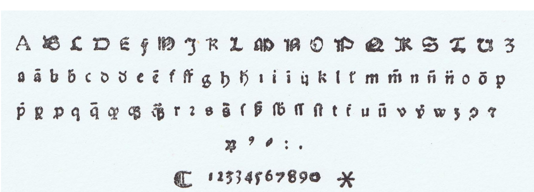


Figure 1: *Polonia Typographica III* ([13]), fragment of plate 112: font number 1

With the exception of the small samples included in the publications (probably photozincography was used), the authors of the series worked with original prints. In the meantime many of those prints have been digitized and are available on the Internet. Hence it is possible to compare the character sets provided in *Polonia Typographica* with the texts which are supposed to use them. The first (and for the time being the only) such attempt, described in [11], concerned the so-called font (in Polish paleotypographical terminology *pismo*) number 1 of Florian Ungler's first printing shop (1510–1516). It revealed some surprising discrepancies.

According to the editor of the third volume of *Polonia Typographica* [13], the font in question was used in only three publications,[4] all of them digitized by now:

1. *Algorithmus* [...] by Sacro Bosco (1511): [20],
2. *Almanach ad a. 1511* by Aurifaber: [22],
3. *Rubricella dioecesis Cracoviensis ad a. 1511* (author unknown): [5].

We will focus here on the third item, see Fig. 2 (the page format is *plano*, an unfolded sheet;[5] I don't know the exact size) which was used in [11] in a very limited way.

The print quality is low, the font is worn out, so for many letters the impression of the typeface is incomplete or distorted in some other way. Unfortunately there is no information about the earlier uses of the font (Ungler came to Poland from Bavaria).

2 Typemes

It is far from clear what the basic units of texts are (and how they should be called), especially when working with old texts.

To account for different user needs, in [18, p. 22] four levels of transcriptions were identified: graphic, graphetic, graphemic and regularized. A different classification is used in [19, p. 4] which, for so-called Ground-Truth texts, distinguishes diplomatic,[6] semi-diplomatic and hyper-diplomatic transcriptions. Indeed, talking about transcription levels is not precise; there is a whole multi-dimensional space of transcriptions differing in independent aspects of editing (e.g. are ligatures preserved? Are abbreviation marks resolved? Are misprints corrected?).

Our primary goal is the "typemic" transcription, representing a text as a sequence of typemes in the sense of Jacques André. He introduced the notion of typeme (in French, *typème*) in [3], and elaborated on it in [2], where the notion of typemic transcription was introduced. Specific typemes are listed in several draft technical notes by André, available at `jacques-andre.fr/PICA/`. The most recent [4] lists 1172 typemes. The typeme is described as the "cast character", considered regardless of the design and

[1] It is worth noting that the idea of an inventory has been proposed independently from time to time, e.g. [1].

[2] `tw.staatsbibliothek-berlin.de/`

[3] `worldcat.org/search?q=au=Gesellschaft%20fu%CC%88r%20Typenkunde%20des%20XV%20Jahrhunderts`

[4] I take this claim for granted, but it would be nice to verify it.

[5] `http://ihl.enssib.fr/paper-and-watermarks-as-bibliographical-evidence/the-shape-of-paper`

[6] The adjective is derived from *diplomatics* (`en.wikipedia.org/wiki/Diplomatics`), not *diplomacy*.

doi.org/10.47397/tb/44-3/tb138bien-rubricella

Figure 2: *Rubricella dioecesis Cracoviensis ad a. 1511*

size, in other words the typemes are the glyphs of characters present in the [metal] typefaces.[7]

In other words, we want to represent precisely the print types/sorts of the text transcribed.

The typemes are assigned the code points of the Unicode characters if appropriate ones exist, sometimes in the case of polysemic glyphs ([2, p. 129]) in an arbitrary way.

Usually it is not sufficient to provide users with only one type of transcription. If the document is available on a server, then usually the server software allows switching between various transcriptions. It is desirable to have a similar option for documents downloaded on a local computer. Theoretically PDF can be used for this, as multiple layers of text are allowed, e.g. for engineering and other applications. An example of a different idea for presenting multiple aspects of old texts in a single PDF document is discussed in [16]. In DjVu documents, it might be useful to use the hidden text layer for the typemic transcription and the annotations to provide another version of the transcription, but this is outside the scope of this paper.

3 Workflow

3.1 The sources

Latin *rubricella* (in Polish *rubrycela*[8]) is another name of the *Directorium Divini Officii*[9] or *Ordo Divini Officii*; the rubricellae discussed here have a very simple form.

In the digital library, the rubricella we're concerned with is accompanied by three pages of a typewritten text reconstructing the fragments missing (the page has been trimmed) or damaged; the reconstruction was based on another rubricella for this year printed by another printer, namely Jan (Johann) Haller, and preserved in the Jagiellonian Library as *Cim. vol. 4*. It is not digitized, but Jacek Patryka, the head of Old Prints Section, kindly provided me with a high quality photo.

The reconstruction was not based directly on Haller's rubricella, but on "Sawicki's copy". At first this remark was unclear to me, but later Mark Thakkar on the Facebook group of *The Paleography Society* pointed Sawicki's book out to me [21], which contains the full text of Haller's rubricella (following common editorial practice, almost all abbreviations have been resolved and expanded).

Although the content of the rubricella is in principle not relevant to the character inventory, I compared the sources, creating a parallel text, as shown in Fig. 3: Haller's rubricella in Sawicki's transcription, reconstructed Ungler's rubricella transcription, original Ungler rubricella typemic transcription. In particular, this allowed for resolving some non-obvious abbreviations. On the other hand the parallel text revealed some strange discrepancies, but this is outside the scope of this paper.

The parallel text was typeset using `expex`[10] by John Frampton; according to its author's description *The package provides macros for typesetting linguistic examples and glosses, with a refined mechanism for referencing examples and parts of examples.* The name is derived from the names of the two basic commands, `\ex` and `\pex`, which in turn are abbreviations of *example* and *parts example*.

3.2 Image preprocessing

Working with a very large page would be cumbersome even on a large display, so I decided to split the page into five parts, in reading order: the initial single column text, the left narrow column, the left half-page column, the right narrow column, the right half-page column. The digital library provides the document in the DjVu format (once very popular[11]) but there are no tools for such editing in this format. Hence the document was exported as a graphic file with `djview`[12] and split into PNG files with Gimp.[13]

The transcription process described below for unknown reason degrades the scan quality. Therefore they were also converted to a DjVu document with the `didjvu` program. The program was written by Jakub Wilk, who no longer maintains it, in Python 2. I am using now Friedric Foebel's Python 3 fork.[14]

3.3 The transcription

For transcription the so-called Expert Client to the Transkribus system[15] was used; see Fig. 4.

It is worth mentioning that the Optical Character Recognition feature, understood literally, is no

[7] From [4, p. 4]: *«caractère fondu» mais sans tenir compte ni du dessin, ni de la taille des caractères. On peut dire que ce sont les glyphes des caractères présents dans les polices de caractères au sens original d'inventaire (décompte des sortes).* Incidentally, the notion of typeme in other meaning has been used by a Swedish linguist Göran Hammarström, but this is not relevant to our purposes.

[8] `pl.wikipedia.org/wiki/Rubrycela`

[9] `en.wikipedia.org/wiki/Directorium`

[10] `ctan.org/pkg/expex`

[11] To make a long story short, the original reasons for creating it are no longer valid, but it is still the best format for some purposes; see, e.g. [8].

[12] `djvu.sourceforge.net/djview4.html`

[13] `gimp.org/`

[14] `github.com/FriedrichFroebel/didjvu`

[15] `readcoop.eu/transkribus/`. The Expert Client is no longer supported as it has been phased out by a Web interface. The formal status of the system is complicated. Some tools are open source and free (`github.com/Transkribus/`), while the use of other tools is charged per page.

Ungler 1-7 Ungler 2-3

(7)	Quadragesima	in	illa:	co	ra	tur.	Pascha:	san cti	que.	Rogationes:	ur	ban	ni.	Penthecosten:	iun	pri
	[Quad]ragesima	in	illa:	co	ra	tur.	Pascha:	sancti	que.	Rogationes:	ur	ban	ni.	Penthecosten:	iun	pri
[1-7]	⧺rageſiima.	In	illa:	Co	ra	tur.	Paſcha:	���	.	⧺tiões.	Ur	ban	in⚠.	Penthecoſten.	Iun	pri

mi.	Corporis	Christi	pro	dos	al.	Adventus	Domini:	andr	de cem
mi.	Coꝛporis	Christi	pro	dos	al.	Adventus	Domini:	andr	
mi.	Corp̄⚠is	xꝑi	pro	t⚠os	al.	Aduentus	ꝺn̄.	Anꝺr	

(8)	Item	vigilia	Epiphanaie	veniens	in	diem	dominicam	ibidem	teneatur
	[I]tem	vigi[lia	Epiphan]ie	veniens	in	diem	dominicam	ibidem	teneatur
[2-1]	⧺tem	vigi⧺	⧺ie	veniēs	in	ꝺiē	ꝺn̄icū⚠	ıbiꝺē	teneatur

(9)	ita	quod	sabbato	ad	vesperas	antiphonae	Tecum	principium
	[it]a	quod	sabbato	[ad]	[vesperas]	[antiphonae]	Tecum	principium
[2-2]	⧺a	qꝫ⚠	ſabbato	⋯	⋯	⧺phne⚠.	Tecū	pꝛincipium

(10)	cum	aliis	antiphonis	et	psalmis	et	ad	finem	prout	rubrum	viatici	docet.
	[cum?]	aliis	antiphonis	et	psalmis	et	ad	finem	prout	rubrum	viatici	do⸗
[2-3]	⧺i	alijs	an̄ijs	⁊	pſalmis	⁊	aꝺ	finē	ꝓut	rubꝛo	vratici	ꝺo⸗

Figure 3: The parallel text of the rubricella versions

longer in use. It was replaced by neural network tools, usually named Handwritten Text Recognition (HTR), which operate on the whole lines of texts. Although the results of HTR are often quite impressive, the segmentation into characters and even words is not needed for these algorithms. The tools for such detailed segmentations are not available easily, at least I did not find any such tool appropriate for my goals.[16]

The images created with Gimp and uploaded to Transkribus are segmented into text lines automatically, but the low quality of the print confused the algorithm quite often, so extensive manual corrections were needed.[17]

In the transcription I use ⧺ (U+29FA DOUBLE PLUS) to mark a missing beginning or ending of a word, and ⋯ (U+22EF MIDLINE HORIZONTAL ELLIPSIS) to mark a whole missing word. Moreover I use ⚠ (U+26A0 WARNING SIGN) to mark fragments which require special attention, and I use � U+FFFD REPLACEMENT CHARACTER for illegible characters.

I didn't aim at a perfect transcription; it is good to delay some decisions to a later stage.

Transkribus supports exporting the transcription in several formats. The ultimate format for me is the DjVu format mentioned earlier. I quite often exported a PDF file, with the *images plus text layer* option selected, until I encountered a problem with the (no longer maintained) pdf2djvu program by Jakub Wilk.[18]

So, this time I decided to export the transcription in the ALTO format (*Analyzed Layout and Text Object*)[19] with the option *Split Lines Into Words*.

[16] In particular, the Glyph Miner software package (`github.com/benedikt-budig/glyph-miner/`) looked quite promising, but its use appeared prohibitively difficult, primarily because of the lack of sufficient documentation (`github.com/benedikt-budig/glyph-miner/discussions/14`).

[17] For the reasons mentioned, Transkribus does not provide a tool for an automatic word segmentation. It is possible to do it by hand, but it is rather cumbersome, so I preferred to defer this operation to a later stage of the workflow.

[18] `github.com/jsbien/pdf2djvu/issues/1`

[19] `loc.gov/standards/alto/`

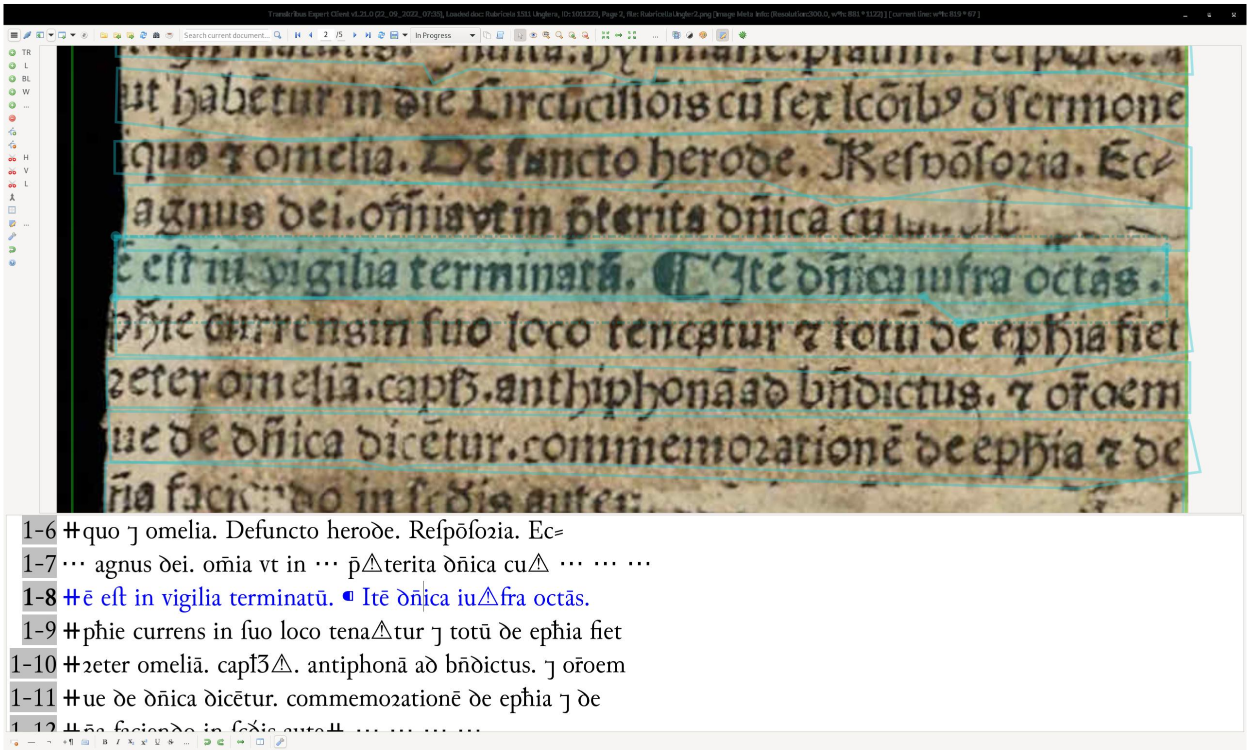


Figure 4: The transcription window of Transkribus Expert Client

It should be noted that recognizing words in old prints without understanding the text or using a dictionary is close to impossible, because of inconsistent use of the horizontal (interword/interletter) spaces. I don't know what algorithm is used by the ALTO export but the results, at least for *Rubricella*, were unfortunately not satisfactory and required manual adjustment, which was done during the indexing phase.

The ALTO files (one file for each page) were converted first to *hOCR* ([12]) with the ocr-transform tool.[20]

The next step was using the html2djvused program. Originally written by Jakub Wilk in Python 2 and distributed with ocrodjvu,[21] it is no longer maintained by him. Several forks exist with more or less complete conversion to Python 3, and I used one of these.[22] As the name suggests, the output of html2djvused was used by djvused, one of the tools of the DjVuLibre library,[23] to import the transcription as the hidden text layer into the DjVu document mentioned earlier in section 3.2.

[20] github.com/UB-Mannheim/ocr-fileformat
[21] github.com/jwilk-archive/ocrodjvu
[22] github.com/rockclimb/hocr2djvused
[23] djvu.sourceforge.net/

Last but not least, the basic metadata has been inserted into the document, providing in particular the link to the digital library containing the original version. This is the document which is used in the later stages of the workflow.

3.4 Indexing and annotating

The next step in the workflow intensively used the djview4poliqarp program[24] (designed by myself and programmed by Michał Rudolf). At first it was just a fork of djview intended to facilitate the use of DjVu corpora; see, e.g. [9]. Later, the program was extended to create and browse simple indexes to DjVu documents.[25]

From the technical point of view the indexes are just simple CSV files (using semicolon as the separator). Every line of an index file consists of three or four fields:

1. The text used for sorting and incremental search.
2. The reference to the relevant image fragment in the form used by the djview4 viewer mentioned earlier, namely a Universal Resource Locator. Some fully-fledged examples can be found in the

[24] github.com/jsbien/djview-poliqarp_fork
[25] E.g. github.com/jsbien/iLindeCSV, github.com/jsbien/Zaborowski-index4djview

indexes to Linde's dictionary, in particular in the tiny index of the planet symbols.[26]

In the indexes discussed here the scheme and authority parts are absent, and the path is limited to the file names; this means in practice that djview4poliqarp has to be called with the index directory as the default one. The fragment part is also missing, and the query part contains the dimensions and the coordinates of the image fragment in the djview4 specific form; it can also contain the specification of a color used for highlighting. The referenced image fragment has to be rectangular, but a single url can reference several such fragments.

3. A description: text displayed for the current entry in a small window under the index.
4. An optional comment displayed after the entry; we precede it by ※ (U+203B REFERENCE MARK) for a more distinctive display. It was absent in the first versions of the program and was added later, primarily to distinguish homographs.

The initial index is created by a quick-and-dirty modification of the above-mentioned djvused program.[27] Here is an example of a line created by the program for *Rubricella*; it is composed from the following fields (below split into several lines for editorial reasons):

1. ɔꝺuctus (the entry),
2. `file:Rubricela_1511_Unglera.djvu?` `djvuopts=&page=0003` `&highlight=0237,2075,0117,0019` (the url, with no color specification; as we use a rudimentary form of the path, the file has to be present in the current directory),
3. `Rubricela_1511_Unglera.djvu p=3 l=19` `tl=258 w=39 tw=530` (the description: the file name, the page number, the line number on the page, the line number in the document, the word number on the page, the word number in the document; especially useful when several indexes are merged),
4. ※ ɔꝺuctus (the comment, initially equal to the entry but prefixed by ※ for the reason mentioned earlier, usually somehow edited later to contain the transcription, in this case, *conductus*).

The index, in addition to being browsed with the djview4poliqarp program, also has other applications,

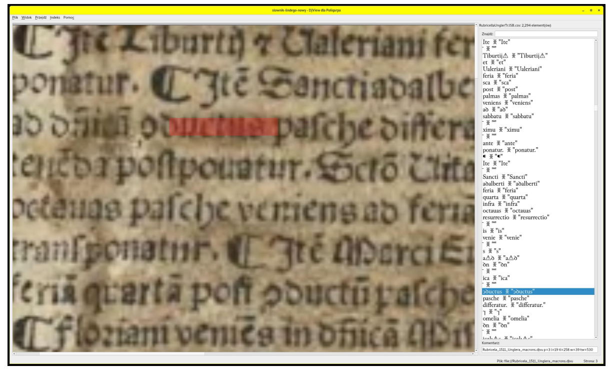


Figure 5: The incorrect bounding box of an original index entry (ɔꝺuctus, on the third line)

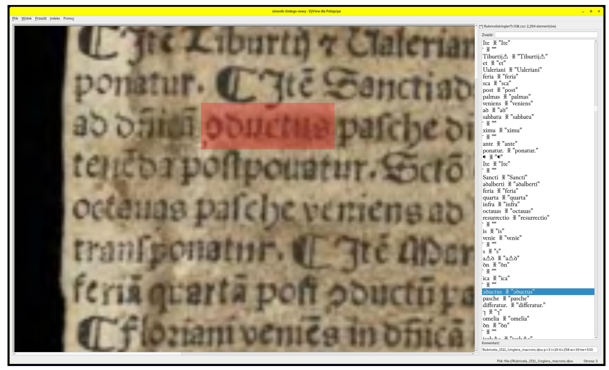


Figure 6: The manually corrected bounding box of the entry

e.g. a simple sed script creates a secondary index where the comments are entries and *vice versa*.[28]

Using djview4poliqarp, I corrected by hand the borders of practically all 1638 words of *Rubricella*; see Figs. 5 and 6. It would be nice to incorporate the changes into the hidden text layer. It can be done with a very simple program which is, however, yet to be written.

To double-check the index it is good to make a histogram of all the characters occurring in the text. For this, I have usually used unihistext,[29] a fork of Bill Poser's unihist, implemented some time ago by a student of mine following my suggestions. It provided the names for the Unicode and some private use characters and their combining sequences. Unfortunately, it was written in Python 2 and its adaptation to Python 3 does not seem trivial. Hence now I have to use the original unihist.

It should be noted that it is very easy to create various specialised subindexes by extracting the relevant lines of the CSV file with, e.g. grep. For

[26] `github.com/jsbien/iLindeCSV/blob/master/Linde-znaki/planety.csv`

[27] At present, I keep it in a private repository as I hope to replace it by something more elegant which I will make public. Help here would be greatly appreciated.

[28] See the sed directory at `github.com/jsbien/Zaborowski-index4djview`.

[29] `github.com/jsbien/unihistext`

```
sed "s/.\+;file:\(.*\)[.]djvu[?]djvuopts=&page=\([0-9]\+\)&highlight=\([0-9]\+\),\([0-9]\+\),\([0-9]\+\),\([0-9]\+\);.\+\'
/ddjvu -format=ppm -page=\2 -segment=\5x\6+\3+\4 \1.djvu imgtmp\/snippet_\1_page\2x\3y\4.ppm/"
input.csv > output.bat
```

Figure 7: Extracting graphic snippets (the script is split into several lines for editorial reasons)

```
U;file:RubricellaUnglerTrJSB.djvu?djvuopts=&page=1&highlight=8,55,259,313;RubricellaUnglerTrJSB.djvu p=1 l=2 tl=2 w=1 tw=1; * "U"
```

gets converted to

```
ddjvu -format=ppm -page=1 -segment=259x313+8+55 RubricellaUnglerTrJSB.djvu imgtmp/snippet_RubricellaUnglerTrJSB_page1x8y55.ppm
```

Figure 8: A graphic snippet example

```
convert imgtmp/*.ppm -set filename: "%t" imgtmp/%[filename:].png
```

Figure 9: Converting a directory of PPM files to PNG format

```
sed "s/\(.\+\);file:\(.*\)[.]djvu[?]djvuopts=&page=\([0-9]\+\)&highlight=\([0-9]\+\),\([0-9]\+\),\([0-9]\+\),\([0-9]\+\);.\+\'
/\\\\includegraphics[height=3ex]{snippets\/snippet_\2_page\3x\4y\5} % \1/"
input.csv > output.tex
```

Figure 10: Creating LaTeX source snippets (the script is split into several lines for editorial reasons)

```
U;file:RubricellaUnglerTrJSB.djvu?djvuopts=&page=1&highlight=8,55,259,313;RubricellaUnglerTrJSB.djvu p=1 l=2 tl=2 w=1 tw=1; * "U"
```

gets converted to

```
\includegraphics[height=3ex]{snippets/snippet_RubricellaUnglerTrJSB_page1x8y55} % U
```

Figure 11: LaTeX source snippet example

this purpose some special tags can be added to selected fields (with djview4poliqarp or a favourite text or spreadsheet editor). Such subindexes can be used in particular for snippet extraction described below.

3.5 Snippets extraction

The main tool for snippet extraction is ddjvu, and calls to ddjvu are generated by a sed script; see Figs. 7 and 8. You might notice that the graphic snippet file name incorporates the information about its origin. I consider this very useful, but unfortunately due to a not-yet-fixed bug it is not always reliable.

To allow for including the snippets in a LaTeX document we convert them to PNG format; see Fig. 9.

Finally, another sed script (Fig. 10), creates appropriate LaTeX code snippets to facilitate creating figures; here, please note that the entry field is preserved as the comment (Fig. 11).

4 The inventory

The inventory is presented using the expex package mentioned earlier. The first line contains the graphic snippets, and the second the subsequent numbers for reference purposes. The third line contains the typemic transcription typeset with the Junicode Two font, and the last one contains a kind of high-level transcription for clarity, for which I don't have a name yet. I apply the following rules, roughly in the order listed.

1. Ligatures are split into the component letters, e.g. ‘ﬁ’ becomes ‘ſi’ and ‘ĳ’ becomes ‘ij’. The letters may be subject to further processing, e.g. ‘ſi’ becomes ‘si’ and ‘ij’ becomes ‘ii’ (there was/is no letter ‘j’ in Latin;[30] nevertheless ‘j’ was/is used in the so-called Ramist forms[31]).
2. Obsolete letters (or rather obsolete letter shapes) are replaced by their modern equivalents, e.g. ‘ſ’ becomes ‘s’ and ‘ꝺ’ becomes ‘d’.
3. Ambiguities of ‘u’ and ‘i’ are resolved and replaced by the Ramist forms, i.e. when appropriate ‘u’ is replaced by ‘v’ and ‘i’ by ‘j’,[32] e.g. ‘Aue’ becomes ‘Ave’.
4. Brevigraphs[33] are resolved; in most cases their meaning is obvious, but sometimes understanding the context is needed, e.g. isolated ‘ꝑ’ becomes ‘per’ and ‘ꝑmanente’ becomes ‘permanente’ but ‘coꝛꝑe’ becomes ‘corpore’.
5. Other abbreviations are resolved, which sometimes may require good understanding of the context, e.g. ‘an̄e’ means here ‘antiphonae’ but this is far from obvious; moreover there are special rules for *nomina sacra*, see below.

[30] medium.com/in-medias-res/theres-no-j-in-latin-your-holiness-5a331c3f7a06

[31] The name comes from Pierre de la Ramée (Petrus Ramus); see http://stmarys-parish.org/Latin/LatinSpelling.htm.

[32] Reportedly alatius.com/macronizer/ should be able to do this automatically.

[33] This term is used by me in a narrow sense to refer to a scribal abbreviation in the form of a single character.

6. Letter case is adjusted to be compatible with modern spelling.
7. Obvious mistakes are noted and corrected.

These rules describe the transcription of single words. For multiword texts, additional rules are needed for marking the proper segmentation, but we won't use them here. This clarifying transcription is provided only when it differs from the typemic transcription.

The typemic transcription uses the default character glyphs, which can be sometimes quite different from the original. In some fonts a better approximation of the shape can be achieved using OpenType features, e.g. in Junicode 'ꝉ' can be rendered as 'ꝯ' using `cv17` (`cv` stands for character variant).

4.1 Some problems of typemic transcription

Peter Robinson,[34] the editor of the manuscripts of Geoffrey Chaucer, wrote [17, pp. 186–187]:

> scribes, over and over, don't seem to care whether the two minims[35] are joined at the top (as in a modern printed n; [...]) or joined at the bottom (as in a modern printed u; [...]), or not joined at all. And then, what we call a macron takes a bewildering variety of forms. Sometimes it is indeed a single straight stroke over the letter. Sometimes it extends over several letters. Often it is curved. And very often it appears as a loop, beginning at the base of the last minim and arching back over the two minims of the u/n character and preceding letters. Attempting to devise transcription protocols in these circumstances is a complex dance with a collection of hydra. The intentions of the scribes appear increasingly opaque and distant, and we are left searching our own intentions. Exactly what are we trying to record; for who; and why?

To my surprise the problem with minims occurs in *Rubricella*, as illustrated in Fig. 12. Sometimes 'n' is used instead of 'u', cf. example (8) and *vice versa*, cf. examples (2) and (7). Sometimes 'm' is used in place of 'in' or 'ni', cf. examples (1) and (3), and sometimes 'un' is used as 'im', cf. example (6).

However, the most intriguing are the fragments where 'm', 'n' and 'u' look just like a sequence of dotless *ı*, cf. examples (4) and (9). I have no idea whether this is an effect of the type being worn down, or whether it is an intentional use of the dotless *ı*. In consequence it is not clear what the typemic representation of such fragments should be. For the time being I have no answer to this question. I also don't know whether this phenomenon occurs also in other prints of that time.

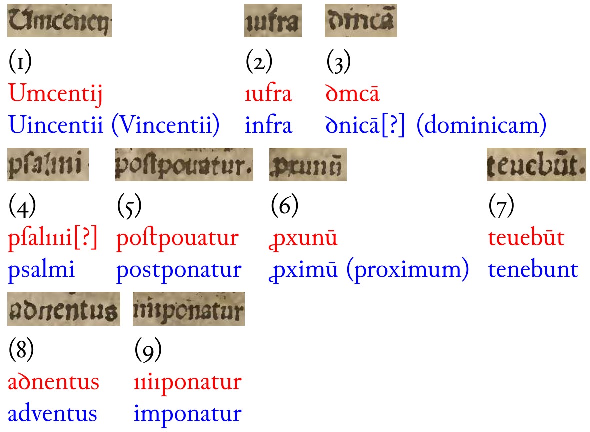
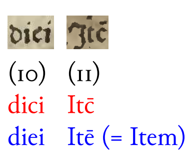


Figure 12: Examples of difficulties with minims

On the other hand the various shapes of the overline abbreviation mark definitely appear also in this and other publications (see some examples below). All of them are encoded, at least at present, just as a macron (in Unicode '◌̄' U+25CC COMBINING MACRON).

Another problem worth mentioning concerns the variants of letter *i*. There is no doubt that 'ī' (in Unicode U+012B LATIN SMALL LETTER I WITH MACRON) is an abbreviation, but the dotless *ı*,[36] the standard *i* with dot and *í* with acute seem to be used with the same function, When there is no doubt about the shape, they are encoded respectively as 'ı' U+0131 LATIN SMALL LETTER DOTLESS I, 'i' U+0069 LATIN SMALL LETTER I and 'í' U+00ED LATIN SMALL LETTER I WITH ACUTE.

In some cases 'c' is used instead of 'e'; some examples:



It is not clear whether they are just mistakes or intentional replacements due to technical limitations.

4.2 Majuscules

Not all majuscules listed in Fig. 1 occur in the text; see Fig. 13.

---

[34] `artsandscience.usask.ca/profile/PRobinson`

[35] Wikipedia (`en.wikipedia.org/wiki/Minim_(palaeography)`) states: *A minim is the basic stroke for the letters i, m, n, and u in uncial script and later scripts deriving from it*; the limitation to specific scripts does not seem necessary.

[36] By the way, this is the primary form of the letter; the dot, called technically a tittle, appeared in the Middle Ages; see, e.g. `quora.com/Why-do-"i"-and-"j"-posses-a-dot`.

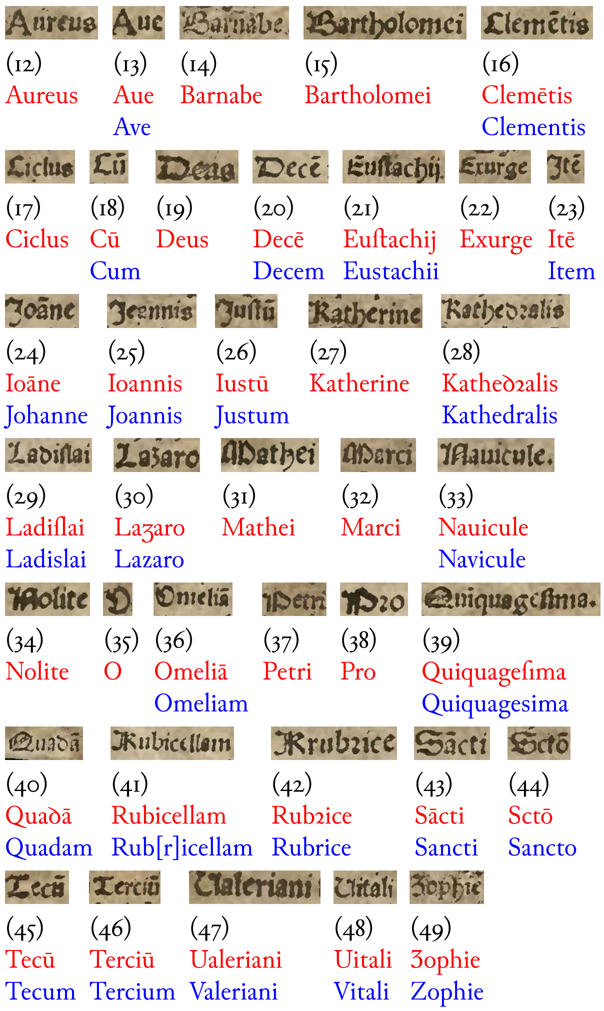


Figure 13: Majuscule (and minuscule) examples

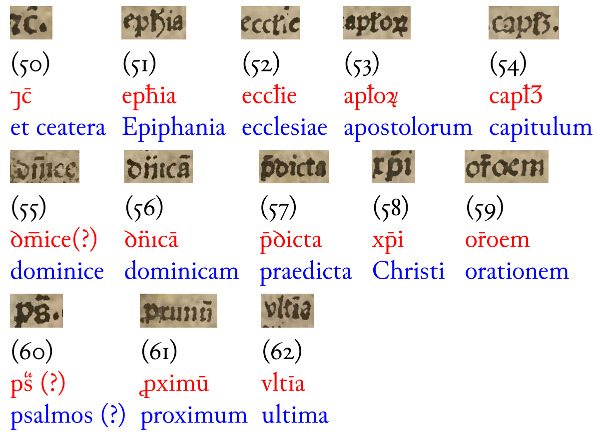


Figure 14: More miniscule examples

4.3 Minuscules

Fig. 1 lists minuscules, their ligatures and brevigraphs in one sequence; we prefer to describe them separately. For most of the minuscules the Unicode encoding is obvious.

The following minuscules occur in the examples in Fig. 13 (the number of a representative example is given in parentheses):

a (14)	ā (24)	b (14)	c (20)	ꝺ (29)	e (20)
ē (20)	ẽ (20)	f (2)	g (39)	ı (2)	i (25)
l (28)	m (39)	n (24)	o (24)	ō (44)	p (49)
q (39)	r (27)	ꝛ (28)	s (25)	ſ (4)	t (31)
u (26)	ū (26)	x (22)	ȝ (30)		

The remaining minuscules, namely ‘c̄’ (50), ‘ī’ (62), ‘ł’ (53), ‘n̄’ (56), ‘r̄’ (59), are illustrated in Fig. 14.

Some minuscules are supplemented by an abbreviation mark, usually a more or less horizontal line over the letter, conventionally encoded as a macron. Sometimes the mark resembles a tilde (example (57)), or a flattened circumflex (example (50)). Sometimes the abbreviation is marked by a diaeresis (example (46)). Sometimes the bar is so short that it looks like a dot (example (58)). The most intriguing is the abbreviation mark in example (60). Is this just a diaeresis? Or a little known character COMBINING LATIN SMALL LETTER FLATTENED OPEN A ABOVE proposed by the Medieval Unicode Font Initiative?[37] Or something else? I don't know the definite answer, though Susana Tavares Pedro supported my hypothesis in the Facebook Paleography Society group.[38] On the other hand, in Fig. 1 the diacritical mark over *s* looks rather like a macron.

It is interesting to compare the letter *l* in examples (4) and e.g. (15). Is this the same type/sort from the same font? I'm not sure.

A reader may be curious what the letter ‘3’, usually just the equivalent of ‘z’, is doing in the abbreviation of ‘capitulum’ (example (54)). This is an example of a homographic character; it should be interpreted, at some level of transcription, as Unicode U+A76B LATIN SMALL LETTER ET (in Junicode: ‘ꝫ’). The scribes used to write ‘m’ vertically to save the space, the letter written this way looked as ‘ꝫ’, so one of its meanings is just ‘m’.

A reader may be also curious what ‘x’ and ‘p’ are doing in an abbreviation of ‘Christi’ (example (58)). The answer is that ‘x’ stands for the Greek *chi*, and ‘p’ for the Greek *rho*. This is just an example of the special abbreviation rules for *nomina sancta*.

[37] mufi.info/q.php?p=mufi/chars/unichar/7635
[38] facebook.com/groups/7687162686/permalink/10159164467367687

4.4 Ligatures

Besides the ligatures classified here as brevigraphs and discussed below, Fig. 1 lists only ‘ﬀ’ (U+FB00 LATIN SMALL LIGATURE FF). However, in the text we can see also ‘ﬅ’ (U+FB05 LATIN SMALL LIGATURE LONG S T, example (5)), ‘ſl’ (M+EBA3[39] LATIN SMALL LIGATURE LONG S L [MUFI 4.0], example (29)), and ‘ſi’ (M+EBA2 LATIN SMALL LIGATURE LONG S I [MUFI 4.0], example (39)).

The ligature ‘ĳ’ (U+0133 LATIN SMALL LIGATURE IJ, example (10)), is just a variant of ‘ii’, as already mentioned.

4.5 Brevigraphs

Some popular brevigraphs have already occurred in the examples above, namely ‘⁊’ (U+204A TIRONIAN SIGN ET, in the Junicode font also ‘Ꝫ’; see example (50)), ‘ꝉ’ (U+A749 LATIN SMALL LETTER L WITH HIGH STROKE, in Junicode also ‘ꝉ’), which occurs with different meanings in examples (52), (53) and (54), and ‘ꝝ’ (U+A75D LATIN SMALL LETTER RUM ROTUNDA) (example (53)).

I prefer to classify the character encoded here as ‘h̵’[40] (U+0068 LATIN SMALL LETTER H followed by U+0335 COMBINING SHORT STROKE OVERLAY), in Junicode also ‘ƀ’[41], see also example (51), as a brevigraph, but the diacritic can be used also with other letters; see an example below.[42] However, because the diacritic touches the base letter, I prefer to treat them as a whole.

Essentially the same diacritic is used also with the letter ‘b’ which is in turn ligated with the long *s*; see example (63). David Baker suggested encoding it as the sequence of ‘ſ’, ‘b’ and ‘◌̵’ (U+0335 COMBINING SHORT STROKE OVERLAY),[43] which in the Junicode font is rendered as the ligature ‘ſƀ’. I prefer to treat the ligature as a single brevigraph.

Fig. 1 lists two brevigraphs based on the letter *p*, namely ‘ꝑ’ (U+A751 LATIN LETTER P WITH STROKE THROUGH DESCENDER) and ‘ꝓ’ (U+A753 LATIN SMALL LETTER P WITH FLOURISH). The first character is illustrated below in example (64), the second can be seen in examples (6) and (61).

[39] Using the M prefix for the characters from the recommendation of Medieval Unicode Font Initiative was proposed by me in [10]; I've spotted this convention used independently elsewhere, unfortunately I don't remember where.

[40] Its shape can be also approximated by ‘ꝁ’ (M+E8A3 LATIN ABBREVIATION SIGN AUTEM [MUFI 4.0]).

[41] github.com/psb1558/Junicode-font/discussions/134

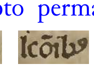
[42] In the Facebook Paleography Society group, Lisa Howarth wrote (facebook.com/groups/7687162686/permalink/10158299890607687) *When attached to an ‘h’, it usually means ‘er’ or ‘ab’ depending on the word. It can also stand as a more general abbreviation in longer words [...].*

[43] github.com/psb1558/Junicode-font/discussions/233

The figure lists three brevigraphs based on the letter *q*. The first two are well known: ‘ꝙ’ (U+A759 LATIN SMALL LETTER Q WITH DIAGONAL STROKE), see example (65),[44] and ‘ꝗ’ (M+E8BF LATIN SMALL LETTER Q LIGATED WITH FINAL ET [MUFI 4.0]), see example (66). The third is LATIN SMALL LETTER Q LIGATED WITH FINAL ET [MUFI 4.0] with a diacritic mark, which can be interpreted as the ‘◌ᷓ’ U+1DD3 COMBINING LATIN SMALL LETTER FLATTENED OPEN A ABOVE mentioned earlier, giving the interpretation ‘ꝗᷓ’. Another interpretation, used here, is just ‘◌̈’ (U+0308 COMBINING DIAERESIS). The brevigraph can be used as a separate word.

The figure lists two brevigraphs based on the long *s*. The first one is ‘ẜ’ (M+E8B7 LATIN SMALL LETTER LONG S WITH FLOURISH [MUFI 4.0], seen in example (68)); it can also be used as a separate word (its interpretations were suggested in the Facebook Paleography Society group by Gionata Brusa and Carolus Hrachowiczensis[45]). The second one, the ligature with the letter *h* and a diacritic mark, was discussed above.

					
(63)	(64)	(65)	(66)	(67)	(68)
ſƀſcripto	ꝑmanente	ꝙ	vſꝗ	ꝗ̈	ẜ
subscripto	permanente	quod	usque	quam	scilitet? sed?

	
(69)	(70)
ɔclufa	lcōibꝰ
conclusa	lectionibus

Further, example (69) illustrates the brevigraph ‘ɔ’ (U+2184 LATIN SMALL LETTER REVERSED C) and example (70) shows the brevigraph ‘ꝰ’ (U+A770 MODIFIER LETTER US).

Fig. 1 also lists brevigraphs in the form of an insular *d* with a diacritical mark[46] and a *v* with a diacritical mark, but they don't occur in the text of *Rubricella*.

4.6 Other characters

Other characters listed in the figure fall into two categories: digits, and (in a rather large sense) punctuation marks. Besides the full stop and the semicolon, we have here a hyphen, namely ‘⸗’ (U+2E17 DOUBLE OBLIQUE HYPHEN), an asterisk (not present in *Rubricella*) and a so-called rubric[47] which can

[44] The brevigraph is often used as a separate word.

[45] facebook.com/groups/7687162686/posts/10159250228377687

[46] github.com/psb1558/Junicode-font/discussions/133

[47] en.wikipedia.org/wiki/Rubric

be interpreted as '◖' (U+204C BLACK LEFTWARDS BULLET).[48]

5 Final remarks

The workflow presented here can be considered an exemplification of the rule *The best tool is the tool you know best* (author unknown to me). It is acceptable for hobbyists, but for serious research, subject to the principle *publish or perish*, it seems too cumbersome and time-consuming. A possible way to streamline the workflow would be to extend djview4poliqarp with some appropriate import and export facilities, but it is practically impossible because the program is orphaned.

Another extreme is illustrated by, for example, a complicated workflow presented in a recent paper [15].

Some time ago, I wrote *a printed or typed text is quite different from any other kind of utterance, because it is in fact a string of characters from a finite, well defined alphabet.* [7, p. 143]. As you can see, I was too optimistic, the "alphabet" of old prints is far from being well defined. There are still some open questions concerning the form (and the content) of Ungler's *Rubricella* and Ungler's font number 1 which deserve further investigation.

Last but not least the reader should be warned that I don't know Latin and my knowledge of paleo(typo)graphy is rather rudimentary. I tried to cross-check the statements made in the paper, but I could certainly have made errors. I will appreciate all comments and corrections.

Almost all the resources discussed here will be available, after additional verification, in a public repository at `github.com/jsbien/Rubricella`*; the repository is private at the time being, but individual access can be granted on request.*

References

[1] J. André. The Cassetin Project — Towards an inventory of ancient types and the related standardised encoding. *TUGboat* 24(3):314–318, 2003. `tug.org/TUGboat/tb24-3/andre.pdf`

[2] J. André, R. Jimenes. Transcription et codage des imprimés de la Renaissance. *Revue des Sciences et Technologies de l'Information — Série Document Numérique*, 16(3):113–139, 2013. `doi.org/10.3166/DN.16.3.113-139`

[3] J. André. Les typèmes de Garamont. À propos d'un projet de codage des caractères anciens. In *Passeurs de textes II. Gens du livre et gens de lettres à la Renaissance*, C. Lastraioli, I. Diu, C. Benevent, eds., Tours, France, 2012. Brepols. `jacques-andre.fr/japublis/ja-passeurs.pdf`

[48] `github.com/psb1558/Junicode-font/discussions/93`

[4] J. André. Inventaire des typèmes latins et français existant dans Unicode/MUFI ou à y faire entrer. Note de travail nt-2, Projet d'Inventaire des Caractères Anciens, 2022. `jacques-andre.fr/PICA/SIGMA-PICA.pdf`

[5] Author unknown. *Rubricella dioecesis Cracoviensis ad annum 1511.* Florian Ungler, Kraków, 1510. `www.wbc.poznan.pl/dlibra/show-content/publication/edition/422952`

[6] L. Bernacki. Monumenta typographica Poloniae XV et XVI ss. *EXLIBRIS*, Zeszyt III:89–90, 1920. `kpbc.ukw.edu.pl/publication/11872`

[7] J.S. Bień. Towards computer systems for conversing in Polish. In *Computational and mathematical linguistics. Proceedings of the International Conference on Computational Linguistics. Pisa, 27 VIII–1 IX 1973*, A. Zampolli, N. Calzolari, eds., vol. II of *Linguistica*, pp. 139–160. Leo S. Olschki Editore, Firenze, 1980. `aclanthology.org/C73-2015`

[8] J.S. Bień. Digitalizing dictionaries of Polish. In *Methods of Lexical Analysis: Theoretical assumption and practical applications*, K. Bogacki, J. Cholewa, A. Rozumko, eds., pp. 37–45. Wydawnictwo Uniwersytetu w Białymstoku, Białystok, 2009. `www.researchgate.net/publication/37686726`

[9] J.S. Bień. The IMPACT project Polish Ground-Truth texts as a DjVu corpus. *Cognitive Studies | Études Cognitives*, 14:75–84, 2014. `ispan.waw.pl/journals/index.php/cs-ec/article/view/cs.2014.008`

[10] J.S. Bień. Problemy kodowania znaków w korpusach historycznych. In *Semantyka a konfrontacja językowa*, D. Roszko, J. Satoła-Staśkowiak, eds., vol. 5, pp. 67–76. Instytut Slawistyki PAN, Warszawa, 2016. `www.researchgate.net/publication/338448462`

[11] J.S. Bień. Repertuar znaków pisma nr 1 pierwszej drukarni Unglera (1510–1516) na podstawie Polonia Typographica. *Acta Poligraphica*, pp. 1–20, 2021. `http://www.cobrpp.com.pl/actapoligraphica/uploads/pdf/AP2021_Bien.pdf`

[12] T.M. Breuel. The hOCR microformat for OCR workflow and results. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pp. 1063–1067. IEEE Computer Society, 2007. `www.dfki.de/fileadmin/user_upload/import/4373_The_hOCR_Microformat.pdf`

[13] H. Bułhak. *Pierwsza drukarnia Floriana Unglera 1510–1516 : tablice 61–120.* Polonia Typographica Saeculi Sedecimi : zbiór podobizn zasobu drukarskiego tłoczni polskich XVI stulecia. Zakład Narodowy im. Ossolińskich, Wrocław, 1959. `academica.edu.pl/reading/readMeta?cid=129206090&uid=129085566`

[14] A.F. Johnson. Polonia Typographica Saeculi Sedecimi. Warsaw, Polska Akademia Nauk, Instytut Badań Literackich, 1936–64. Fasc. 1–5, Pls. 1–245. *The Library*, s5-XX(4):330–331, 1965. `doi.org/10.1093/library/s5-XX.4.330`

[15] F. Kordon, N. Weichselbaumer, et al. Classification of incunable glyphs and out-of-distribution detection with joint energy-based models. *International Journal on Document Analysis and Recognition (IJDAR)*, 26:223–240, Sept. 2023. `doi.org/10.1007/s10032-023-00442-x`

[16] M. Ogrodniczuk, W. Gruszczyński. Embedding Transcription and Transliteration Layers in the Digital Library of Polish and Poland-Related News Pamphlets. In *Towards Open and Trustworthy Digital Societies*, H.R. Ke, C.S. Lee, K. Sugiyama, eds., pp. 54–60, Cham, 2021. Springer International Publishing.

[17] P. Robinson. The digital revolution in scholarly editing. In *Ars Edendi Lecture Series, vol. IV*, B. Crostini, G. Iversen, B. Jensen, eds., pp. 181–207. Stockholm University Press, 2016. `www.stockholmuniversitypress.se/site/chapters/10.16993/baj.h/download/464/`

[18] P. Robinson, E. Solopova. Guidelines for Transcription of the Manuscripts of The Wife of Bath's Prologue, 2006. `http://canterburytalesproject.com/pubs/transguide-MI.pdf`

[19] C.A. Romein, T. Hodel, et al. Exploring data provenance in Handwritten Text Recognition infrastructure: Sharing and reusing Ground Truth data, referencing models, and acknowledging contributions. Starting the conversation on how *We* could get it done, Nov. 2022. `doi.org/10.5281/zenodo.7267245`

[20] J. de Sacro Bosco. *Algorithmus Ioannis De Sacro Busto*. Florian Ungler, Kraków, 1511. `dbc.wroc.pl/publication/3586`

[21] J. Sawicki. *Statuty synodalne krakowskie biskupa Jana Konarskiego z 1509 roku.* PAU, 1945.

[22] Stanislaus Cracoviensis (Aurifaber). *Almanach ad annum incarnacionis 1511.* Florianum Ungleb[sic], Cracovie, 1511. `eod.vkol.cz/60812/60812.pdf`

◇ Janusz S. Bień
Warsaw, Poland
`jsbien (at) uw.edu.pl`
`sites.google.com/view/jsbien`
ORCID 0000-0001-5006-8183

A Ti*k*Z rendering of the Arecibo message

Travis Stenborg

Abstract

A new tutorial on the Ti*k*Z package for vector graphics in LaTeX is presented. Rendering binary-encoded data of the interstellar Arecibo message, into a digital image, was used as the learning exercise. Demonstration of data indexing, multidimensional lists, nested loops, string concatenation, variable declarations, and more, are given for budding Ti*k*Z enthusiasts. The material also has potential for adaptation to custom rendering 2D data matrices more generally.

1 Introduction

The Arecibo message is a well-known historical radio transmission, sent into space to spark extraterrestrial communication [1]. Sent from the Arecibo Telescope, the message contained pictograms of topics deemed likely to engage extraterrestrial intelligence: numbers, atoms, DNA, humanity, the Solar System and even the telescope itself [3, 15].

Modern audiences might liken the Arecibo message pictograms to pixel art. Its structure is rendered concisely here with Ti*k*Z, as a simple example of generating vector graphics from within a LaTeX document. More complex Ti*k*Z examples can be found elsewhere in *TUGboat* [2, 7, 8, 10, 11, 17, 18] and beyond [4, 6, 16].

2 The Ti*k*Z package

Ti*k*Z vector graphics use the `tikz` package, available with its underlying PGF (Portable Graphics Format) system, from CTAN in `graphics/pgf` (`ctan.org/pkg/tikz`). Invocation requires merely a standard document preamble inclusion:

```
\usepackage{tikz}
```

3 Message data

The Arecibo message comprised 1679 bits (a semiprime [9]). Arranged as 73 rows of 23 bits (both prime numbers), it encodes a raster image. A new macro `\data` was defined to emulate the message data and semantic structure. The macro is shown abbreviated, below, and in full in Figure 1.

```
% Data setup: 1679 bits.
\def \data{
{0,0,0,0,0 ... 0,0,0,0,0},   % row 1
...
{0,1,1,1,1 ... 1,1,0,0,0}}   % row 73
```

4 Configuring colors

The binary message data were rendered here in black and white. Custom color names, `pixel0` and `pixel1`,

`doi.org/10.47397/tb/44-3/tb138stenborg-arecibo`

```
\def \data{
{0,0,0,0,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0},
{0,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0},
{1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,1,0,1,1,0,0,1,0},
{1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{1,1,0,0,0,0,1,1,1,0,0,0,1,1,0,0,0,0,1,1,0,0,0},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0},
{1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0},
{1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},
{1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{1,1,0,0,0,0,1,1,0,0,0,0,1,1,1,0,0,0,1,1,0,0,0},
{1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
{1,1,0,1,0,0,0,0,1,1,0,0,0,1,1,1,0,0,1,1,0,1,0},
{1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0},
{0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1},
{1,1,1,1,1,0,0,0,0,1,1,0,0,0,0,0,0,0,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0},
{0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0},
{0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0},
{0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0},
{0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0},
{0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0},
{0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0},
{0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0},
{0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0},
{0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0},
{0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0},
{0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0},
{0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0},
{0,0,1,0,0,0,1,1,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0},
{0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,1,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
{0,0,1,0,0,0,0,1,0,1,1,1,0,1,0,0,1,0,1,1,0,1,1},
{0,0,0,0,0,0,1,0,0,1,1,1,0,0,1,0,0,1,1,1,1,1,1},
{1,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,1,1,0,1,1,1},
{0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,1,1,0,1,1},
{0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,1,1,1,1,1},
{0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,1,0,0,0,0},
{0,0,1,0,0,0,0,0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,1,1,1,0,1,0,1,0,0,0,1,0,1,0,1,0,1,0,1,0,1},
{0,0,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0},
{0,0,0,0,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0},
{0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0},
{0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0},
{0,1,1,0,0,1,1,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0},
{0,1,0,0,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0},
{0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,0},
{0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0},
{0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0},
{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,0,0,1,1,1,1,1,0,1,0,0,1,1,1,1,0,0,0}}
```

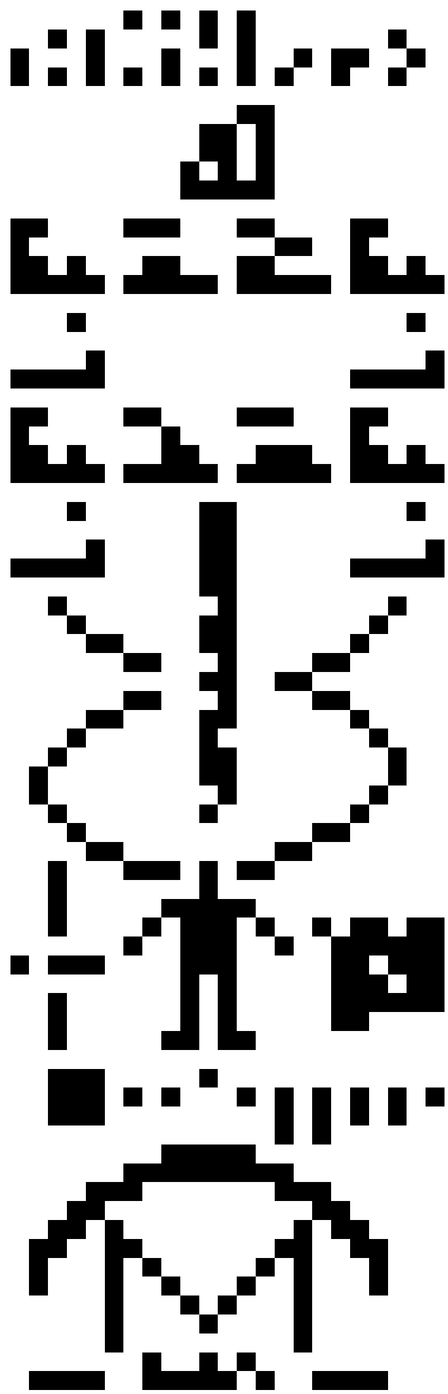


Figure 1: The Arecibo message, 1679-bit data stream TEX macro, and 2D Ti*k*Z render.

Travis Stenborg

were defined with `\definecolor`. Mapping the name suffixes to the possible data values (0, 1), facilitates convenient rendering later.

```
% Configure colors.
\definecolor{pixel0}{rgb}{1,1,1}   % white
\definecolor{pixel1}{rgb}{0,0,0}   % black
```

5 Message rendering

Ti*k*Z graphics may be specified via *path* and supporting commands inside the `tikzpicture` environment. Here is the code, with explanations following:

```
% Render message.
\begin{tikzpicture}
\foreach \row [count=\y] in \data {
  \foreach \bit [count=\x] in \row {
    \fill[color=pixel\bit] (0.25*\x, -0.25*\y)
      rectangle +(0.25, 0.25);
  }
}
\end{tikzpicture}
```

Message data were read here bit-by-bit via the loop command `\foreach`, using the syntax `\foreach` ⟨*variable*⟩ `[`⟨*options*⟩`]` `in` ⟨*list*⟩ `{`⟨*path commands*⟩`}`. The 2D structure of the data was traversed by nesting `\foreach`, using its `count` option to index rows along the image y-axis, and bits along the x-axis.

The `\fill` command rendered each bit on a Cartesian canvas as it was read. The rendering (x, y) position was a function of the `count`-based indexing. Square pixels were specified by a `rectangle` extending from the local render position by 0.25 units along the x and y axes, i.e. `+(0.25, 0.25)`.

Finally, bit values `\bit` were used to dynamically concatenate a target color name, leveraging the suffix-value mapping implemented earlier.

The end result appears in Figure 1.

6 Discussion

This material was intended to be an educational resource for Ti*k*Z. It has the side benefit however, of demonstrating a LaTeX-friendly method for custom vector graphics rendering of 2D data matrices. Many applications might be imagined. An example is visualizing hydrologic time-series data; the Arecibo message structure has already been noted as analogous to raster hydrographs [5].

For those interested in pixel-by-pixel image composition, packages such as `pixelart` [12], `pxpic` [14] or `pixelarttikz` [13] may also be of value.

Acknowledgements

This work was supported by the Australian Research Council Training Centre in Data Analytics for Resources and Environments (project ICI9010031).

References

[1] D. Atri, J. DeMarines, J. Haqq-Misra. A protocol for messaging to extraterrestrial intelligence. *Space Policy*, 27(3):165–169, 2011.

[2] I. Borja. An introduction to automata design with TikZ's automata library. *TUGboat* 44(1):102–107, 2023. `doi.org/10.47397/tb/44-1/tb136prado-automata`

[3] V.V. Demidov. *DNA Beyond Genes: From Data Storage and Computing to Nanobots, Nanomedicine, and Nanoelectronics*, ch. Hiding and Storing Messages and Data in DNA, pp. 7–23. Springer, Cham, 2020.

[4] G. Grätzer. *More Math Into LaTeX*. Springer, Cham, 5th ed., 2016.

[5] R.B. Koehler. *Raster-based Analysis and Visualization of Hydrologic Time-Series*. Ph.D. thesis, U. of Arizona, 2004.

[6] S. Kottwitz. *LaTeX graphics with TikZ*. Packt, Birmingham, 2023.

[7] C. Maggi. The DuckBoat: The Morse code of Ti*k*Z. *TUGboat* 39(1):21–26, 2018. `tug.org/TUGboat/tb39-1/tb121duck-tikz.pdf`

[8] C. Maggi. The DuckBoat: You do not need to be Neo to cope with a Ti*k*Z matrix. *TUGboat* 41(1):20–25, 2020. `tug.org/TUGboat/tb41-1/tb127duck-matrix.pdf`

[9] M.A. Makukov, V.I. shCherbak. SETI *in vivo*: testing the we-are-them hypothesis. *Int. J. Astrobiol.*, 17(2):127–146, 2018.

[10] A. Mertz, W. Slough. Graphics with PGF and Ti*k*Z. *TUGboat* 28(1):91–99, 2007. `tug.org/TUGboat/tb28-1/tb88mertz.pdf`

[11] A. Mertz, W. Slough. A Ti*k*Z tutorial: Generating graphics in the spirit of TeX. *TUGboat* 30(2):214–226, 2009. `tug.org/TUGboat/tb30-2/tb95mertz.pdf`

[12] L. Paternault. *The pixelart package*. `ctan.org/pkg/pixelart`

[13] C. Pierquet. *The PixelArtTikz package*. `ctan.org/pkg/pixelarttikz`

[14] J.P. Spratte. *The pxpic package*. `ctan.org/pkg/pxpic`

[15] Staff at the National Astronomy, Ionosphere Center. The Arecibo message of November, 1974. *Icarus*, 26(4):462–466, Dec. 1975.

[16] M.R.C. van Dongen. *LaTeX and Friends*. Springer, Berlin, 2012.

[17] Z. Walczak. Graphics in LaTeX using Ti*k*Z. *TUGboat* 29(1):176–179, 2008. `tug.org/TUGboat/tb29-1/tb91walczak.pdf`

[18] K. Wolcott. Three-dimensional graphics with PGF/Ti*k*Z. *TUGboat* 33(1):102–113, 2012. `tug.org/TUGboat/tb33-1/tb103wolcott.pdf`

◇ Travis Stenborg
Sydney, Australia
ORCID 0000-0002-2693-9628

A pattern from the Alhambra

Laurence Finston

Abstract

This article demonstrates the use of GNU 3DLDF for creating graphics based on a pattern from the Alhambra, a complex of fortifications and palaces outside Granada, Spain. It explains and demonstrates how to create a perspective drawing using 3DLDF and the technique of *color replacement*.

Introduction

The Alhambra is a complex of fortifications and palaces outside Granada, Spain, built from 1238 into the early 17th century, whereby the era of Islamic architecture there, as in the rest of Spain, ended in 1492 with the completion of the *Reconquista*. It is one of the world's best known architectural monuments and famous for its elaborate ornamentation in the Islamic style, often based on *plane tessellations* and *periodic tilings*, which have inspired many architects, designers and artists, notably M.C. Escher, the incomparable modern master of this métier.

Beginning in 1842, Owen Jones began publishing the monumental work *Plans, Elevations, Sections and Details of the Alhambra*, based on work he had done together with Jules Goury, who died of cholera during their stay at the Alhambra. Published in installments, this work has the distinction of being the first "significant" one to use the process of chromolithography for color reproduction [16].

Figure 1 shows Plate XLIX, No. 85, Mosaic in the Divan, Court of the Fishpond, from Goury and Jones, Vol. 2 [10]. It shows a single rapport of the pattern, so the latter is clearly unusually complex.

Figure 2 shows a version of this pattern which I created using GNU 3DLDF and GIMP with *color replacement* (explained below).

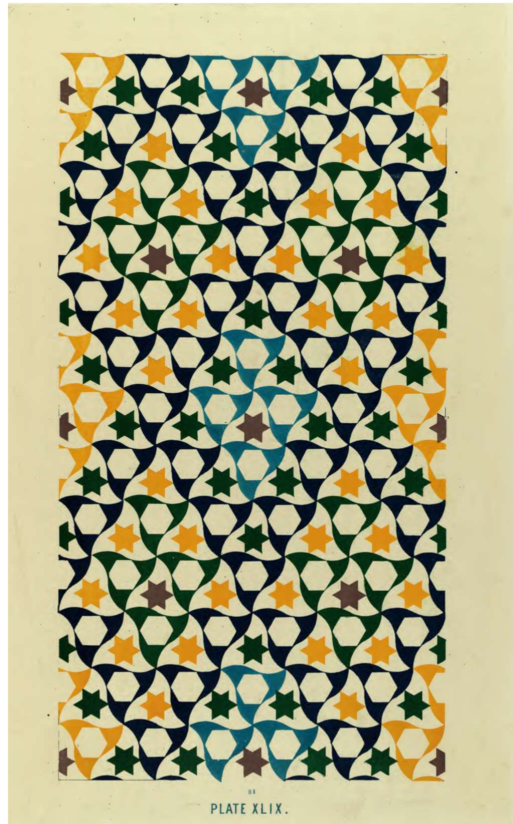


Fig. 1: Plate XLIX from [10].

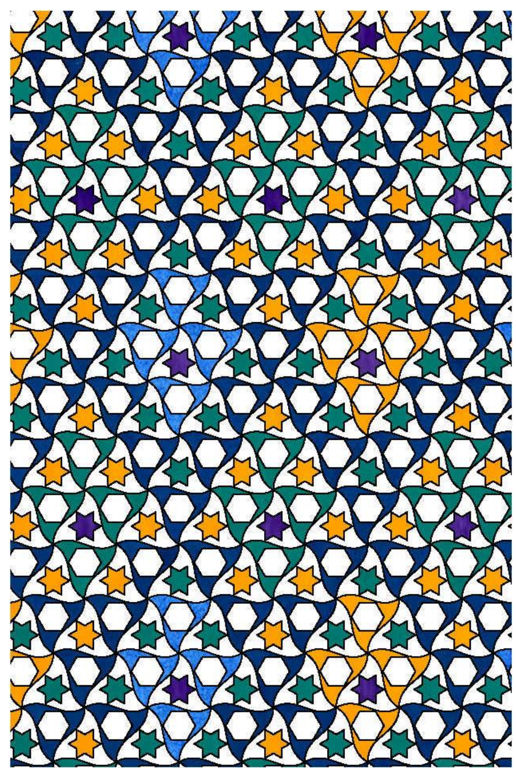


Fig. 2: 3DLDF version, color replaced.

Plane tessellations

"A tessellation or tiling is the covering of a surface, often a plane, using one or more geometric shapes, called tiles, with no overlaps and no gaps." [17]

Of the regular polygons, only three can form a plane tessellation, the equilateral triangle, the square and the regular hexagon (figures 3, 4, and 5 on the following page).

The pattern in question, referred to in the following as pattern 207, is based on the familiar "honeycomb" pattern consisting of regular hexagons.

doi.org/10.47397/tb/44-3/tb138finston-alhambra

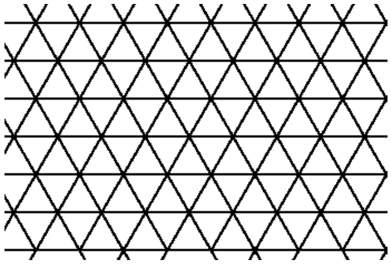


Fig. 3: Plane tessellation, equilateral triangles.

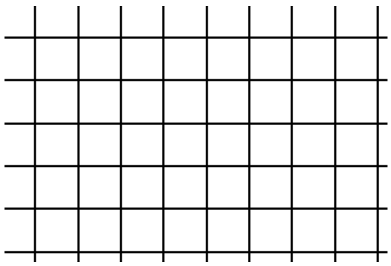


Fig. 4: Plane tessellation, squares.

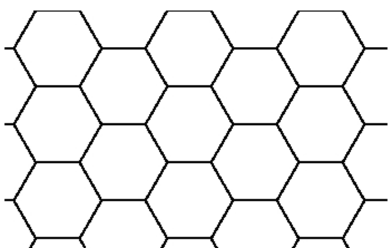


Fig. 5: Plane tessellation, regular hexagons.

Since each hexagon is divided into six equilateral triangles, one may say with as much justification that it is based on the plane tessellation using triangles. In pattern 207, the triangles in each hexagon alternately contain a smaller hexagon or a six-pointed star:

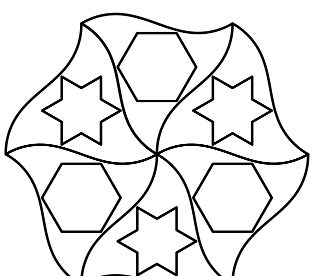


Fig. 6: Hexagon with inner figures.

One may thus consider such a hexagon the "basic unit" of the pattern. One way to fill the plane would be to copy the basic unit and shift it to the right and upward repeatedly, then to copy the original unit downward and repeat the procedure. If, however, we consider the "basic unit" to be two hexagons as shown in figure 7, then the plane may be filled using only translations (*shifts*) in the horizontal and vertical directions, which is more convenient from the point of view of programming.

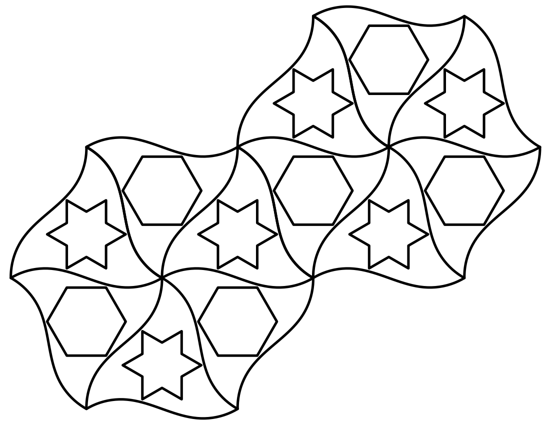


Fig. 7: Basic unit of the pattern.

In the hexagons in pattern 207, the straight edges of figure 6 have been replaced by undulating curves. However, since they are symmetrical, they don't affect the plane-filling property of the tessellation.

In the 3DLDF program, the curve is specified in this way:

```
point Z[]; path hex[];
for i = 0 upto 5:
  Z[i] := (2cm, 0) rotated (0, 0, i*60);
endfor;

path p[]; [...]
for i := .25 step .25 until .75:
  Z[6+j] := mediate(Z2, Z1, i) shifted (0, k*m);
  j += 1;
  m -= 1;
endfor;
p0 := Z2 .. {dir 0}Z6{dir 0} .. Z7
   .. {dir 0}Z8{dir 0} .. Z1;
```

This code would be similar in METAPOST except that the mediation operator `[]` would be used

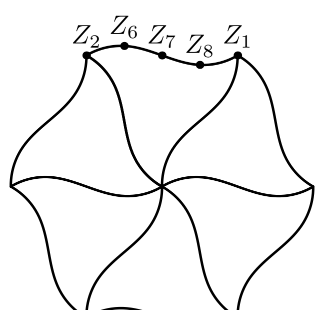


Fig. 8: Hexagon with wavy edges.

instead of 3DLDF's **mediate** command. `[]` doesn't exist in 3DLDF because it would cause conflicts in the Bison parser [8].

The direction specifiers used to define **p0** work fine when the drawing is projected onto the x-y plane, which is, in fact, equivalent to using METAPOST in the first place. However, they will produce erroneous results when it is projected using the perspective projection, for reasons explained at length in [8]. Therefore, in order to make **p0** projectable, it must be "resolved", i.e., enough points must be added to it along its length so that it won't "go out of shape" when projected using the perspective projection:

```
path_vector pv;
pv := resolve p0 to 80;
p0 := pv0;
```

Generally speaking, the more extreme the foreshortening, the more points are needed. So far, 80 has proven to be a sufficient number of points for this drawing.

After the first two hexagons with their inscribed figures have been drawn, they must be copied and translated as many times as necessary to create a single rapport, which consists of 6 rows of 4 hexagons, offset as described above. This is fairly simple. Following this, they must be colored, which is not. This task is performed by the macro **tessellate**, which is defined in the file `sub_alhambra_207_1.ldf`, which like all of the source files referred to in this article, and all of the images contained in it, is included in the distribution of 3DLDF.

For maximum flexibility, each of the paths representing the outlines of the triangles and the inscribed figures is assigned to a variable. This makes it possible to access each one individually. For example, the same pattern could be colored in a completely different way. (The outlines of the outer hexagons are not assigned to **path** variables.)

tessellate takes eight parameters, one for the pen used for drawing the outlines and seven for the colors in the pattern, including the colors for the outlines and the background. This is the call to **tessellate** for figure 9 (on the following page):

```
tessellate {medium_pen, black, white, blue,
   teal_blue, orange, cyan_cmyk, purple};
```

Using parameters for the colors makes it possible to use any combination of colors desired, and in particular, to make "masks" for *color replacement*, as described below. (In the following, most of the images have been rendered to JPEG bitmaps for ease in processing.) **tessellate** draws the outlines on the picture **v6** and fills the areas of color on picture **v5**. Labels are drawn on the picture **v105**. Again, for maximum flexibility, the paths for each individual row of triangles are also drawn and filled on separate pictures (**row_picture_draw0** ... **row_picture_draw12** and **row_picture_fill0** ... **row_picture_fill12**).

While it is most logical to consider the rows of the pattern as referring to the hexagons, it is more practical from the point of view of keeping track of locations within it for the purpose of coloring to consider the rows as referring to the triangles. Seen in this way, there are thus 12 rows in a rapport, offset to one another horizontally, but not vertically.

Creating a picture for each row of triangles makes it possible to use the pattern in drawings without necessarily using only complete rapports or having to clip the picture.

Color in 3DLDF

Color in 3DLDF works similarly to color in METAPOST, but there are a few differences. In METAPOST, **color** (i.e., **(rgb)color** and **cmykcolor**) are two different types; it's not possible to assign a value of one type to a variable of the other. In addition, "greyscale" colors are specified using *numeric expressions*. There is no type **greyscalecolor**.

In 3DLDF, there is only one type, namely **color**, for all three kinds of color and variables of this type can take on RGB, CMYK and greyscale values freely. All **color** objects contain all of the following "parts":

```
red_part green_part blue_part
cyan_part magenta_part yellow_part black_part
grey_part
```

In METAPOST, the spelling "grey" is mostly used rather than "gray". I prefer the spelling "gray", so it is used in 3DLDF except where consistency with METAPOST is desirable, e.g., **grey_part**. However, both spellings may be used for all commands and keywords in 3DLDF, since synonyms are defined for them all.

```
color c[];
c0 := (.5, .5, 0); % RGB
show c0;
>>
color:
name         == c[0]
type         == 3 (RGB_COLOR)
red_part     == 0.50000000
green_part   == 0.50000000
blue_part    == 0.00000000
cyan_part    == 0.00000000
magenta_part == 0.00000000
yellow_part  == 0.00000000
black_part   == 0.00000000
grey_part    == 0.00000000
```

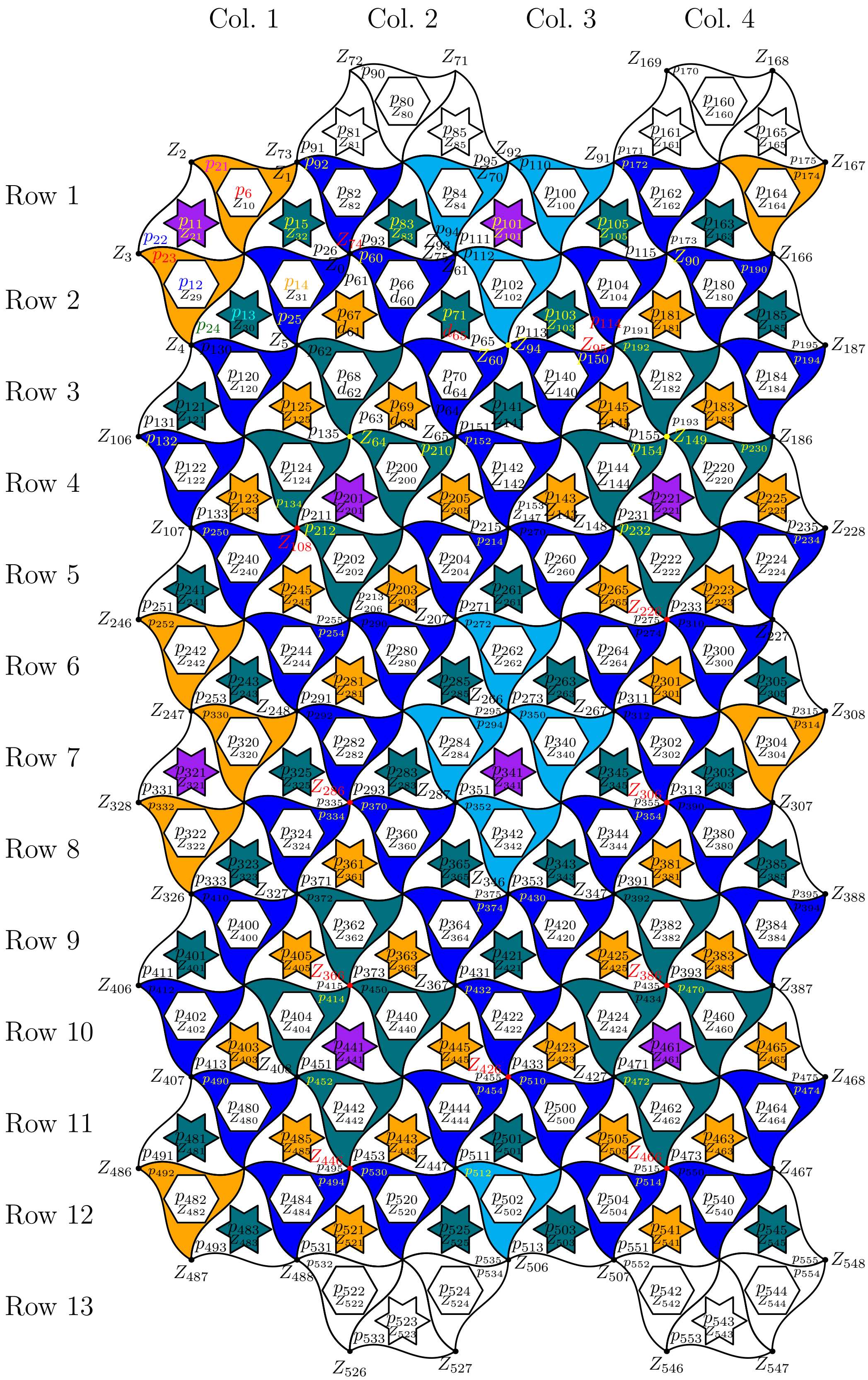


Fig. 9: Colored tessellation example with 3DLDF.

```
c1 := (.2, .3, .4, .5); % CMYK
show c1;
>>
color:
name         == c[1]
type         == 2 (CMYK_COLOR)
red_part     == 0.00000000
green_part   == 0.00000000
blue_part    == 0.00000000
cyan_part    == 0.20000000
magenta_part == 0.30000001
yellow_part  == 0.40000001
black_part   == 0.50000000
grey_part    == 0.00000000

c2 := .3; % Greyscale
show c2;
>>
color:
name         == c[2]
type         == 4 (GREYSCALE_COLOR)
red_part     == 0.00000000
green_part   == 0.00000000
blue_part    == 0.00000000
cyan_part    == 0.00000000
magenta_part == 0.00000000
yellow_part  == 0.00000000
black_part   == 0.00000000
grey_part    == 0.30000001
```

When a value is assigned to a **color** variable, 3DLDF "automatically" recognizes what kind of color it's supposed to be, RGB, CMYK or greyscale and the "parts" are set accordingly. If subsequently a value of a different type is assigned to it, it will take on the new type.

A single numeric value is used to specify a grayscale color, a ⟨*numeric list*⟩ with three elements specifies an RGB color and one with four elements, a CMYK color.

While 3DLDF and METAPOST support RGB and CMYK colors and there are other color models supported by other software, ultimately colors will be converted to the model required by the device that displays or prints them. Generally speaking, computer displays require RGB and printers CMYK. Please note that conversion is *not perfect*, nor can it be. Often the results are good, but sometimes the colors don't match well at all. It seems this occurs most often when CMYK containing a non-zero **black_part** is converted to RGB.

In addition, I have found that it doesn't work at all to import images from 3DLDF or METAPOST containing CMYK colors into the video editing software Flowblade for animations: they are simply not displayed. For this purpose, RGB colors must be used instead.

Color reproduction.

RGB colors. The topics of color in general and color reproduction in particular are complex and it is beyond the scope of this article to discuss them in detail. In short, red, green and blue are the *primary colors* for *additive mixing* and cyan, magenta and yellow are the primary colors for *subtractive mixing*. Additive mixing applies to light and subtractive applies to pigments that absorb light selectively.

Computer displays that could display red, green and blue and mix these colors were finally made possible by the invention of the blue LED. Before this time, the only form of additive mixing in general use was with theatrical gels (i.e., filters) and spotlights. White light, or more accurately, light which the human eye perceives as white, results from mixing red, green and blue light.

The pixels on a computer display, each of which consists of a triplet of tiny LEDs, one red, one green and one blue, do the same thing, but on a much smaller scale and at a much lower intensity.

Mixing red and green light produces yellow, red and blue magenta and green and blue cyan, so yellow, magenta and cyan are the *RGB secondaries* in addition to being the CMYK primaries.

CMYK colors. Schoolchildren are taught that the primary colors are red, blue and yellow and that mixing red and blue produces purple (violet), red and yellow produces orange and blue and yellow produces green. This isn't wrong, but the colors used (at least when I was in elementary school) are somewhat "off" compared to what is now considered correct.

Mixing any red and blue paints will produce a purplish color and the same applies to the other combinations. All of the possible combinations of a given set of a single red, blue and yellow paint, respectively, will produce a "color space". It has been determined, theoretically and/or by experiment, that the largest possible color space under real-world constraints can be achieved by using the colors cyan, magenta and yellow in the shades (not coincidentally) used as ink in offset printing.

What happens in practice when mixing two pigments is in effect equivalent to mixing two pure colors and adding gray, i.e., a mixture of black and white pigments. Since the "colorful" pigments, i.e., not black, white or gray, are more expensive than black, white or gray pigments, it is wasteful to use the colorful pigments for this purpose. It is far better, where possible, to use a pigment of the desired shade and to add gray to it.

In addition, while it is theoretically possible to produce black by mixing cyan, magenta and yellow, the result is unlikely to be satisfactory and these pigments are much more expensive than black, which is often simply soot produced by burning wood (charcoal), acetylene gas or animal bones (in the past, scraps of ivory were also used).

For these reasons, the CMY model (which also exists) is generally expanded to CMYK (where “K” stands for “black”).

Since working with pigments is subject to real-world constraints, a color space with more possible colors and gradations of color can be achieved by using more pigments, including for the CMYK secondaries and other combinations. There are pigments that have shades that are not reproducible by mixing other pigments.

The use of CMYK in offset printing and in laser and inkjet printers is called a “four-color” process. There are also seven-color and eight-color processes which use additional pigments for the sake of better color reproduction.

Built-in RGB colors. These RGB colors are built-in, i.e., they are defined within the C++ code for 3DLDF (in the file `sctpcrt.web`):

```
black white
red green blue
pink
yellow cyan magenta
orange violet purple
yellow_green green_yellow dark_green
blue_violet violet_red
brown
gray light_gray dark_gray
```

Additionally, `grey`, `dark_grey` and `light_grey` are defined as synonyms for `gray`, etc.

Note that cyan, yellow and magenta are defined as secondary RGB colors rather than primary CMYK colors; also, violet and purple are two different colors:

```
show cyan;
>>
color:
name         ==
type         == 3 (RGB_COLOR)
red_part     == 0.00000000
green_part   == 1.00000000
blue_part    == 1.00000000
```
(remaining elements zero)

```
show magenta;
>>
color:
name         ==
type         == 3 (RGB_COLOR)
red_part     == 1.00000000
green_part   == 0.00000000
blue_part    == 1.00000000
```
(remaining elements zero)

```
show yellow;
>>
color:
name         ==
type         == 3 (RGB_COLOR)
red_part     == 1.00000000
green_part   == 1.00000000
```
(remaining elements zero)

```
show violet;
>>
color:
name         ==
type         == 3 (RGB_COLOR)
red_part     == 0.93333334
green_part   == 0.50980395
blue_part    == 0.93333334
```
(remaining elements zero)

```
show purple;
>>
color:
name         ==
type         == 3 (RGB_COLOR)
red_part     == 0.62745100
green_part   == 0.12549020
blue_part    == 0.94117647
```
(remaining elements zero)

CMYK colors defined in plainldf.lmc. Numerous CMYK colors are defined in the `plainldf.lmc` source file, which is included in the distribution of 3DLDF. Versions of the CMYK primaries (including black) and secondaries (which, black aside, correspond exactly to the RGB secondaries and primaries):

```
cerulean_blue teal_blue dark_blue
dark_olive_green mauve turquoise
rose_madder lime_green
```

`plainldf.lmc` also defines RGB versions of these colors:

```
cerulean_blue_rgb teal_blue_rgb dark_blue_rgb
dark_olive_green_rgb mauve_rgb turquoise_rgb
rose_madder_rgb lime_green_rgb
```

In addition, `plainldf.lmc` defines CMYK and RGB versions of the colors defined in the `color.pro` and `colordvi.tex` files from the Dvips distribution (located under `/usr/share/texlive` in
`texmf-dist/dvips/base/color.pro`
`texmf-dist/tex/generic/dvips/colordvi.tex`
on my computer). These colors and their names are based on the box of 64 Crayola crayons (as of some past date). The versions from Dvips are CMYK colors. The RGB versions defined in `plainldf.lmc` have the suffix `_rgb`:

```
color GreenYellow; color GreenYellow_rgb;
color Yellow; color Yellow_rgb;
color Goldenrod; color Goldenrod_rgb;
color Dandelion; color Dandelion_rgb;
```
etc.

Computer-generated vs. painted colors. One problem with filling paths with computer-generated colors is that, leaving *antialiasing* (explained below) aside for the moment, they are entirely uniform. Neither the colors displayed on a monitor nor those produced by an inkjet or laser printer can compare with the appearance of fine artists' colors on high-quality paper. On the other hand, while they can't compare with the appearance of the originals either, the results of scanning drawings and paintings and displaying and printing them using standard office equipment are often surprisingly good.

Since these results are also just representations of pixels, it would, of course, be theoretically possible to simulate scanning a painted background, for example, using the computer only. In practice, however, it is most likely easier to just scan real painted backgrounds, since it would be difficult to program all of the many variations in appearance that in combination result in their characteristic appearance.

Figures 10 and 11 show scans of watercolor and gouache backgrounds, respectively, which I painted. With a few exceptions, they are in DIN A4 landscape format (297mm × 210mm) with a 15mm margin on each side. They have all been painted on watercolor paper, but not all of the same kind. For most I have used paints made by the company Schmincke from their Horadam lines of watercolor and gouache, respectively. Some of the gouache backgrounds were made using paints from Schmincke's HKS Designers' Gouache line and a couple of the watercolors were made using paints from the company Old Holland. Schmincke and Old Holland are both top-of-line manufacturers of artists' colors.

In figure 2, the computer-generated colors have been replaced by portions of a selection of these backgrounds using GIMP.

Paint characteristics. Gouache and watercolor are both made by combining one or more pigments with water and a small amount of gum arabic. However, unlike watercolor, gouache may contain additives to make it opaque, including chalk or white pigment. In traditional watercolor painting, translucent colors are preferred and white paint is never used; white areas in a watercolor painting are achieved by leaving the paper blank. Due to the translucency of the paint and the fact that watercolor doesn't lie on the surface of the paper but rather soaks into it

Fig. 10: Watercolor backgrounds.

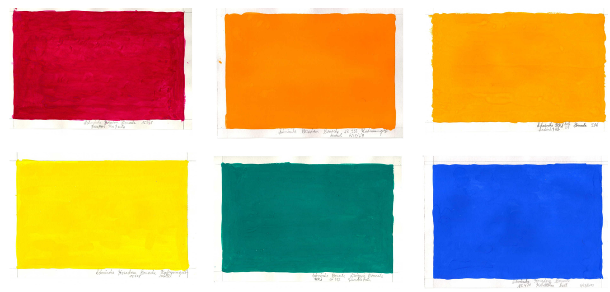


Fig. 11: Gouache backgrounds.

and stains the fibers, the structure of the paper will normally remain visible and contribute significantly to the total impression of the work. Watercolor paints consist almost entirely of pigment so that the colors are very pure. All of these factors contribute to the luminosity that is typical of watercolor paintings. In addition, they often reproduce extremely well.

Many techniques are possible with watercolors. One is to use multiple layers of diluted washes to create a glazed effect. Light colors are typically achieved not by mixing with white, but by diluting the paint. Accidental effects, where the pigment concentrates in particular areas, are often desirable in watercolors.

Gouache, in contrast, is always opaque and has a matte surface. It doesn't soak into the paper to any appreciable extent but rather forms a layer on top of it. Normally, each area of color, or each individual brushstroke in areas containing more than one color, is meant to be completely uniform with respect to color and opacity. In other words, a poster-like effect, but with a matte surface, is usually desired when using gouache. Light colors are achieved by mixing with white; dark colors may have to be lightened with white or they may be virtually indistinguishable from black.

While it is possible to dilute gouache and the results don't exactly look bad, there's normally not much point in doing this, because one might as well just use watercolor, which, unlike gouache, is intended to be used in this way.

The variations in the watercolor backgrounds I've been making for this and other projects cannot practically be achieved by using the computer. With gouache, the situation is different. Since areas of a single color in gouache paintings are usually intended to be entirely uniform, theoretically, the only advantage of using a gouache background is the difficulty of discovering, inventing or stumbling upon attractive colors just by using the computer. In practice, however, there are variations in the gouache backgrounds and the structure of the paper sometimes does show through, adding variability and interest to what appears at first glance to be a solid block of color.

Color replacement

Performing color replacement requires a number of steps. First, the image needs to be output multiple times so that the individual versions may be used as *masks*. Figures 12 through 17 show the parts of the pattern for each color individually with all of the other colors replaced by gray.

These images are generated with METAPOST, and must be imported into GIMP. METAPOST can produce output in the form of EPS, SVG or PNG files. It doesn't matter which format is used, but it is essential that the image be imported *without* antialiasing. Antialiasing is a procedure whereby the color of pixels near an edge where areas of two different colors meet may be altered slightly in order to improve the appearance of the edge.

However, color replacement depends on all areas of a given color being entirely uniform, so that they may be accessed by using GIMP's "Select by color" tool. When importing EPS files, at least, into GIMP a menu appears where antialiasing can be enabled or disabled. I usually have METAPOST output EPS files. When generating PNG output, antialiasing can be suppressed by using the corresponding option to **outputformatoptions**. In METAPOST:

```
outputtemplate := "%j%3c.png";
outputformat:="png";"
outputformatoptions :=
   "format=rgba antialias=none";
```

Or in 3DLDF:

```
verbatim_metapost
  "outputtemplate := \"%j%3c.png\";"
  & outputformat:=\"png\";"
  & "outputformatoptions := "
  & \"format=rgba antialias=none\";";
```

When the image is loaded into GIMP, it has a single layer. First, an *alpha channel* (for transparency)

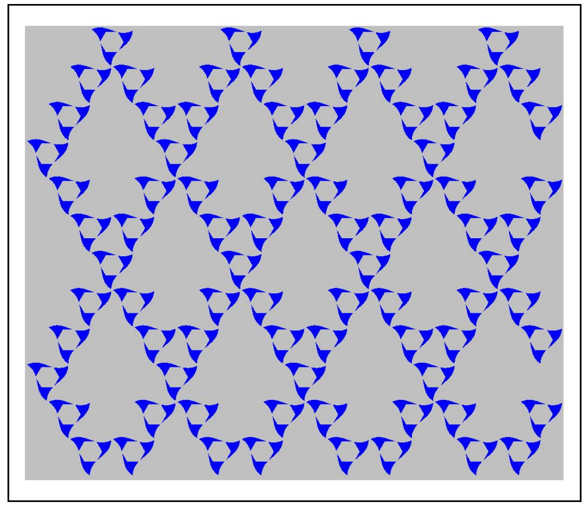


Fig. 12: Blue mask

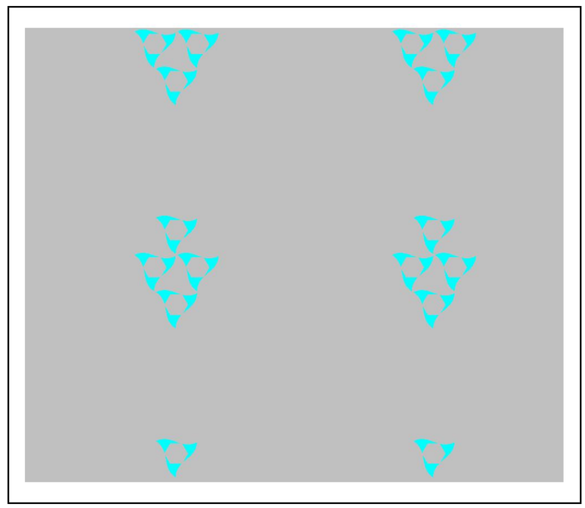


Fig. 13: Cyan mask.

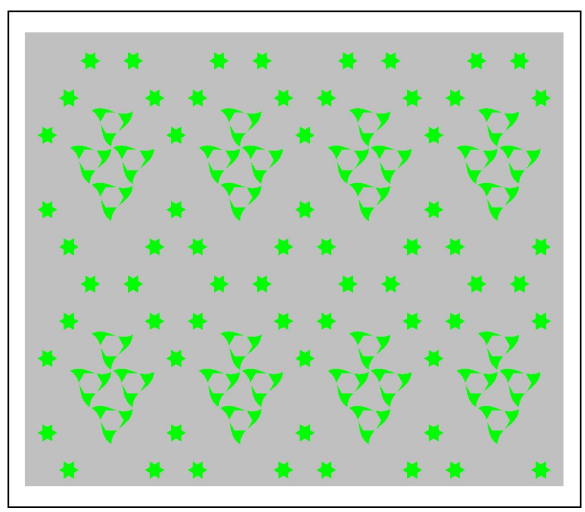


Fig. 14: Green mask.

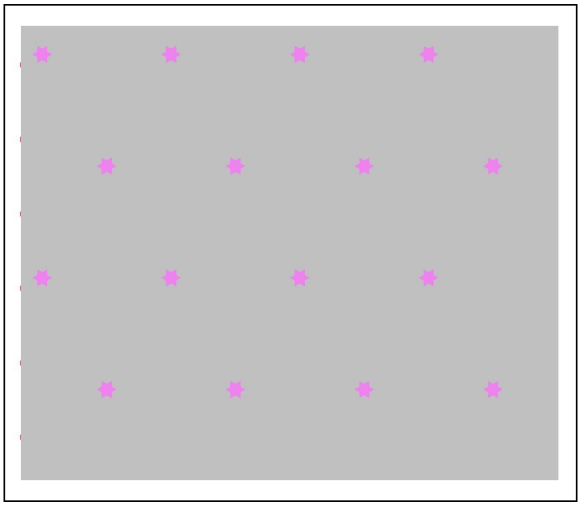


Fig. 15: Purple mask.

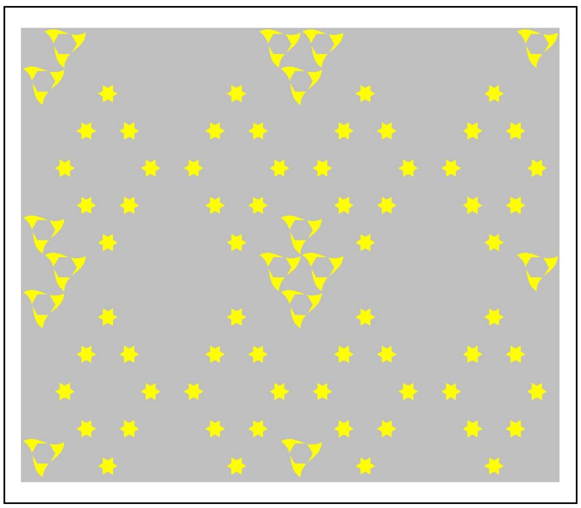


Fig. 16: Yellow mask.

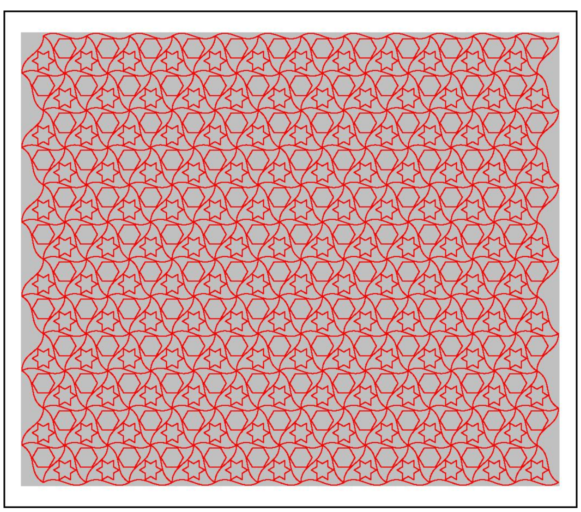


Fig. 17: Outlines mask.

must be added by calling the "Add alpha channel" command in the "Layer" menu. Then the layer is duplicated so that both a positive and negative mask may be created. Although a single mask would suffice, it is convenient to have two separate ones.

To make the positive mask, the blue areas are selected by using the "Select by color" tool and clicking on a spot where there are only blue pixels. This selects all of the blue pixels in the image. Then, the selection is inverted so that all of the other portions of the image are selected instead. These are then made transparent by pressing the "Delete" key.

To make the negative mask, the same procedure is followed, except that the selection isn't inverted, so that the blue areas are made transparent.

In order to fill the blue areas with colors from one of the painted backgrounds, the positive mask is made invisible by clicking on the "eye" symbol next to the name of the layer in the listing of layers in the GIMP window, the negative mask is duplicated and the original negative mask is also made invisible.

Then, the desired file is opened by using the "Open as Layers" command in the "File" drop-down menu. It is placed behind the layer with the copy of the negative mask and the "Merge visible layers" command is executed. The resulting layer contains the portions of the background image corresponding to the blue areas of the pattern, surrounded by gray. (Figure 18.)

Now the gray areas of this layer must be made transparent. However, this cannot be accomplished by simply selecting them by color: This only works when the layer contains areas of color that are completely uniform and distinct. The painted background image is likely to contain pixels of many different colors, including gray. If the "Select by color" tool is used to select gray pixels, pixels within the "blue" areas will also be selected, producing ugly and unusable results. In addition, there is no longer any way to select the areas where the color replacement has taken place directly, again, because these areas now are likely to contain pixels of many different colors and also because the areas aren't contiguous. Therefore neither the "Select by color" nor the "Fuzzy select" tool will select these areas correctly.

To select only the gray areas in the combined layer, first the layer with the original negative mask must be made visible and clicked on to make it the active layer. Then, the gray areas are selected and the combined layer is chosen as the active layer. When the "Delete" key is now pressed, the gray areas on the combined layer are made transparent.

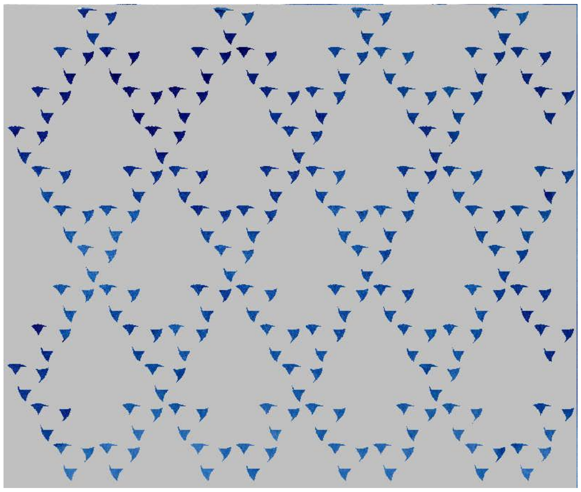


Fig. 18: Blue mask, color replaced.

When an area is selected using one layer and the selection is used to make changes to another, the first layer is called a "mask".

The procedure may be varied and the steps don't necessarily have to be performed in exactly the order given, but the principle remains the same and it's six-of-one, half-a-dozen of the other how it's done exactly, as long as the desired result is achieved.

Of course, it's possible to open multiple files containing background images as layers, combine them with copies of the negative mask, make them visible or invisible, use a single file to contain all of the masks, save them in separate files, etc.

When layers have been created for all of the masks, and with painted backgrounds replacing the computer-generated colors in the original image, they may be *composited* to form a "color replaced" image. (Figure 19.)

The perspective projection. For the parallel projection of pattern 207 onto the x-y plane alone, there would be no need to use 3DLDF: METAPOST would have sufficed. For making versions of it using the perspective projection, 3DLDF is required. Figures 20 and 21 show pattern 207 lying in the x-z plane and projected using the perspective projection, with the focus set as follows:

```
focus f;
set f with_position (5, 10, -40)
   with_direction (5, 10, 10)
    with_distance 45;
```

(The units are cm.)

Masks are created and color replacement is performed in GIMP in exactly the same way as with the version using the parallel projection, except that clipping is performed in GIMP rather than in 3DLDF. If desired, the image may be cropped using the "Crop to content" command in the "Image" drop-down menu.

3DLDF implements various commands for clipping, which all work by writing a call to METAPOST's **clip** ⟨*picture*⟩ to ⟨*path*⟩ operation. This works fine for parallel projections, but for reasons (as yet) unbeknownst to me, it does not for the perspective projection. I plan on debugging the relevant functions, although using a simple example rather than one containing over 300 paths.

There is no particular advantage to performing the clipping in 3DLDF (or METAPOST), especially since clipping in METAPOST is not completely bulletproof: Clipping doesn't actually remove any objects, it just hides them. They are still present and still affect the size of the bounding box of the image, which is important when including the image in a TeX file, for example. In addition, if a clipped image is included in a PDF file (via TeX or some other way), the areas that are supposed to have been hidden may be displayed anyway. I've had this problem with the Firefox web browser.

Bibliography

[1] Bongartz, Klaus et al. *Farbige Parkette.* Basel: Birkhäuser Verlag, 1988.

[2] Bonner, Jay. *Islamic Geometric Patterns.* New York: Springer, 2017.

[3] Brend, Barbara. *Islamic Art.* Cambridge, Massachusetts: Harvard University Press, 1991.

[4] Brentjes, Burchard. *Die Kunst der Mauren.* Köln: DuMont Buchverlag, 1992.

[5] Cundy, H. Martyn and Rollet, A. P. *Mathematical Models.* Oxford: Oxford University Press, 1961.

[6] Ernst, Bruno. *Der Zauberspiegel des M.C. Escher.* Berlin: TACO Verlagsgesellschaft und Agentur mbH, 1986.

[7] Escher, M. C. *Graphik und Zeichnungen.* 13. unveränderte deutsche Auflage. München: Heinz Moos Verlag GmbH & Co. KG, 1979.

[8] Finston, Laurence. *An introduction to GNU 3DLDF. TUGboat* 43(3):319–332, 2022. `tug.org/TUGboat/tb43-3/tb135finston-3dldf.pdf`

[9] Gimpl, Karoline. *Andalusien.* DuMont Kunst-Reiseführer. 2. Auflage. Ostfildern: DuMont Reiseverlag, 2012.

[10] Goury, Jules and Jones, Owen. *Plans, Elevations, Sections and Details of the Alhambra.* London: Owen Jones, 1842–1845.

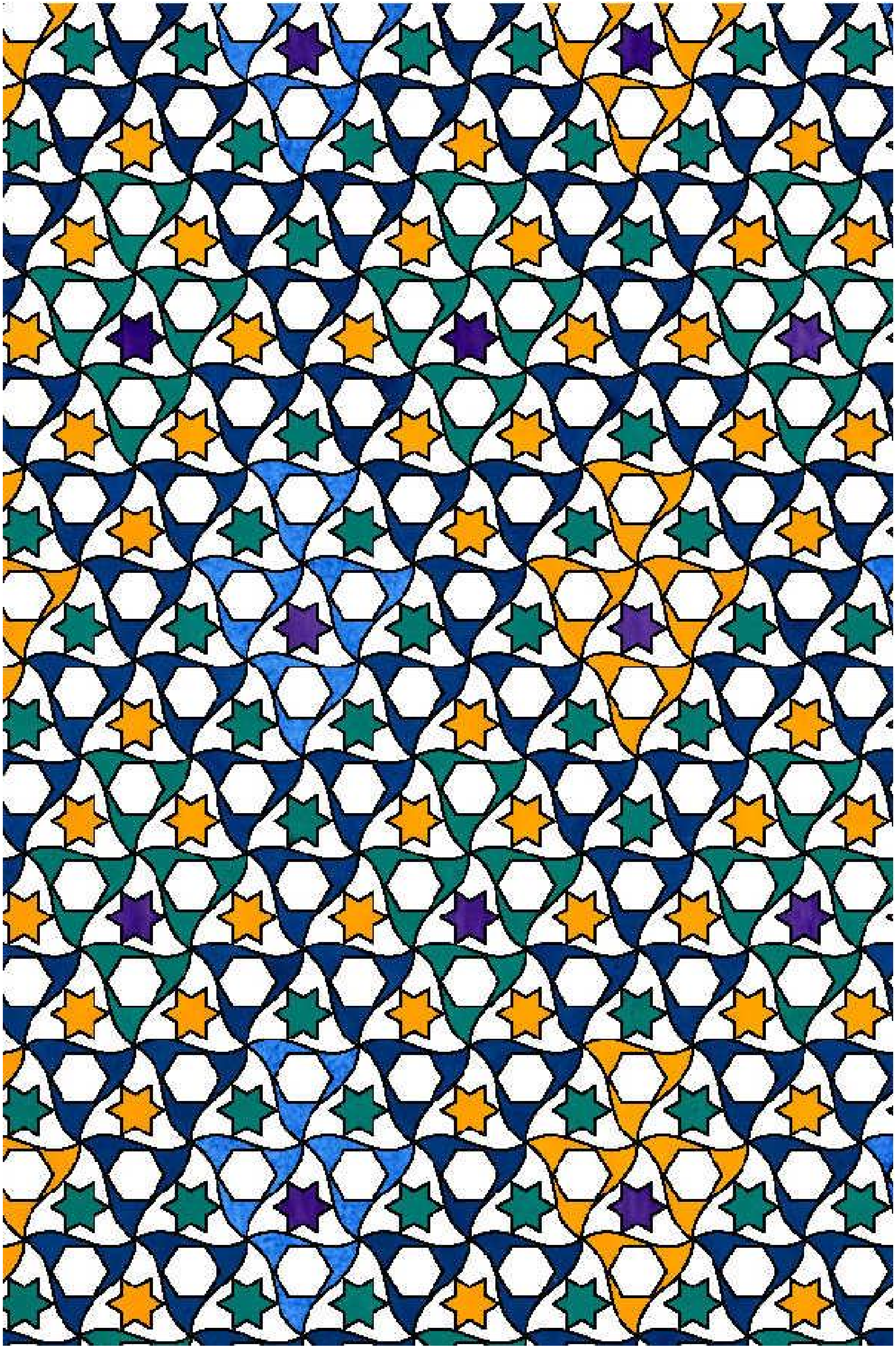


Fig. 19: Color replaced parallel projection

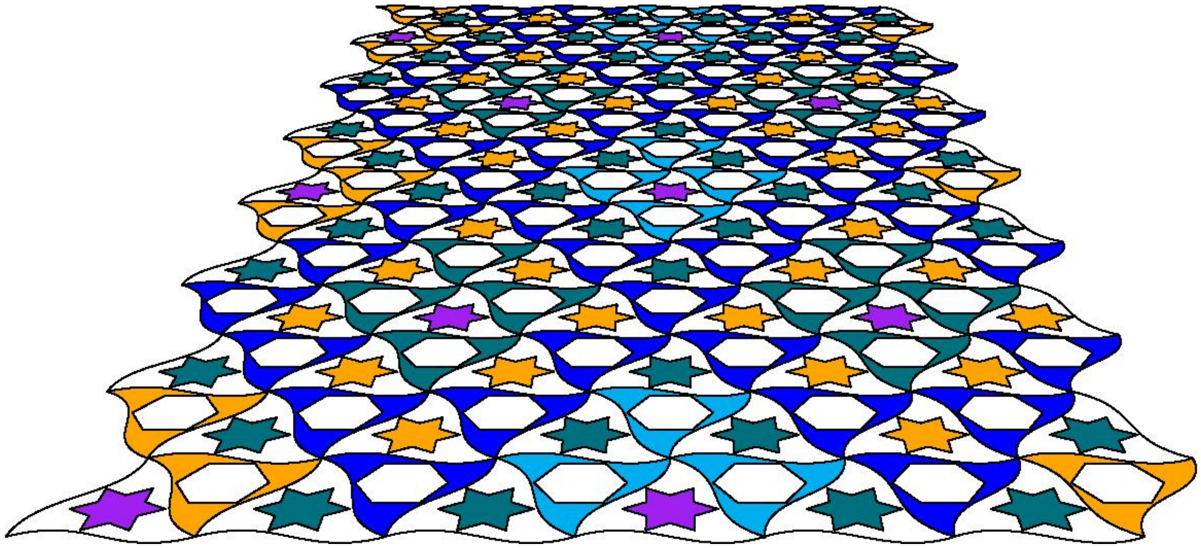


Fig. 20: Perspective projection.

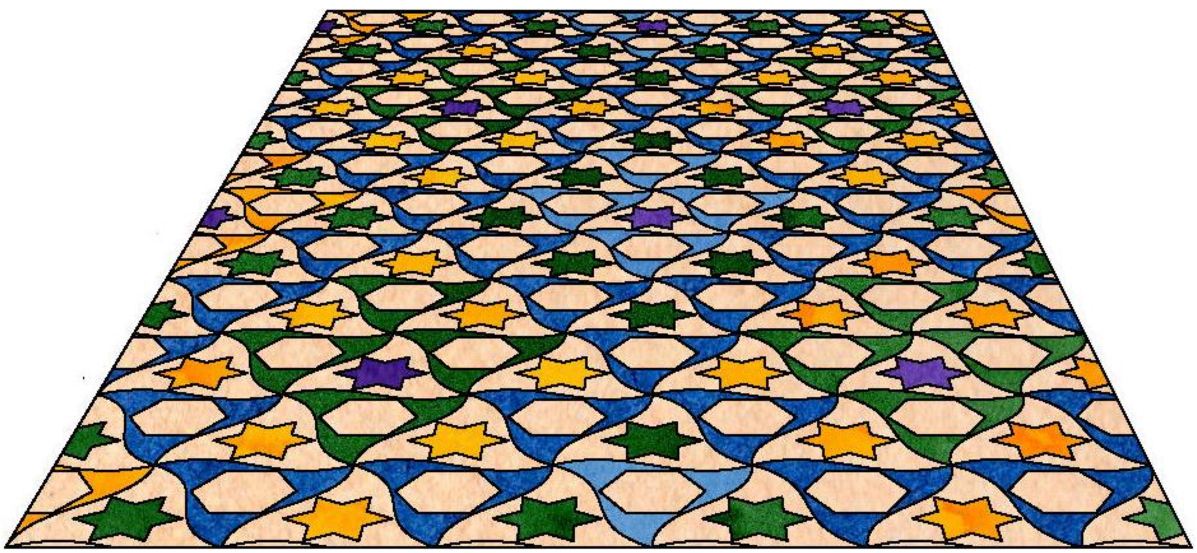


Fig. 21: Perspective projection, colors replaced.

[11] Hobby, John D. and the MetaPost development team. *METAPOST, A User's Manual*. 2020. `tug.org/metapost`

[12] Kühnel, Ernst. *Die Kunst des Islam*. Springers Handbuch der Kunstgeschichte. Stuttgart: Alfred Kröner Verlag, 1962.

[13] Küppers, Harald. *Schule der Farben*. Köln: DuMont Buchverlag, 1992.

[14] Lata, Sabine. *Die Alhambra. Geschichte—Architektur—Kunst*. Berlin: Elsengold Verlag GmbH, 2016.

[15] Wikipedia. *Alhambra*. `en.wikipedia.org/wiki/Alhambra`

[16] Wikipedia. *Owen Jones*. `en.wikipedia.org/wiki/Owen_Jones_(architect)`

[17] Wikipedia. *Tessellation*. `en.wikipedia.org/wiki/Tessellation`

[18] Wilson, Eva. *Islamic Designs for Artists and Craftspeople*. Mineola: Dover Publications, Inc., 1988.

[19] Wolfram Mathworld. *Regular Tessellation*. `mathworld.wolfram.com/RegularTessellation.html`

◇ Laurence Finston
Germany
`Laurence dot Finston (at) gmx dot de`

Using artificial-intelligence tools to make LaTeX content accessible to blind readers

Gerd Kortemeyer

Abstract

Screen-reader software enables blind users to access large segments of electronic content, particularly if accessibility standards are followed. Unfortunately, this is not true for much of the content written in physics, mathematics, and other STEM disciplines, due to the strong reliance on mathematical symbols and expressions, which screen-reader software generally fails to process correctly.

A large portion of such content is based on source documents written in LaTeX, which are rendered to PDF or HTML for online distribution. Unfortunately, the resulting PDF documents are essentially inaccessible, and the HTML documents vary greatly in accessibility, since their accessibility-standards compliant rendering is cumbersome at best.

This paper explores the possibility of generating standards-compliant, accessible HTML from LaTeX sources using Large Language Models. It is found that the resulting documents are highly accessible, with possible complications occurring when the artificial intelligence tool starts to interpret the content.

1 Introduction

Compared to paper printouts, electronic documents generally have the advantage of being accessible to blind readers using screen-reader technology [2]: the software reads the text on the screen using speech synthesis (often at amazing speeds!), and users can navigate the document using the keyboard by following hyperlinks or jumping from section to section. How a screen reader works, unfortunately, is hard to convey without observing users [22, 24].

Particularly in physics and mathematics, but also in other STEM disciplines, TeX (and for several journals particularly LaTeX) continues to be the dominant typesetting environment for scientific articles and reports, as well as for lecture scripts and other teaching materials. The LaTeX source is most frequently processed into a PDF-presentation document for distribution. While screen-reader technology generally works well for web pages, particularly if accessibility standards like the Web Content Accessibility Guidelines (WCAG) [26] are followed, it frequently fails to adequately make PDF documents accessible, which leaves large segments of scientific content out of immediate reach for blind users [17]. Particularly in Europe, where lectures are often accompanied by instructor-authored scripts rather than standard textbooks, these shortcomings of PDF also strongly disadvantage blind students while pursuing higher education [10, 21].

Currently, either particular LaTeX packages with some restrictions on syntax and later rendering need to be applied when writing the LaTeX source [1, 16], or specialized tools need to be used to later mark up the rendered PDF [6, 13]. In an alternative approach, the LaTeX source is rendered to HTML with embedded MathJax [3, 4] using tools like `htlatex`; unfortunately, this approach is also not compatible with all of LaTeX's syntactical variants. Given the current restrictions, some blind users prefer having the original LaTeX source available to their screen-reader software [23]; however, students in introductory courses are likely not proficient in LaTeX.

Narrating mathematical formulas requires either explicit semantic markup, which is usually not provided, or other sense-making technologies. When automated mechanisms fail, this work currently needs to be done by human experts. The Large Language Model GPT has been shown to have remarkable mastery of physics and mathematics [15], and it might be up to the task. In this exploratory study, GPT-4 [19] is used to translate the LaTeX source to HTML while narrating the formulas and graphics in plain language.

2 Presentation versus semantics

Any kind of document we see on paper or on screen is a presentation of content, and we are using certain visual clues and symbols to convey meaning ("semantics"). For example, something that is larger and boldface is interpreted as a heading for a new section or subsection of a document, and when we glance over the document, we might jump from heading to heading to see what we might like to read. Screen-reader software cannot glance over a document; it needs to rely on semantic clues in the document. To the eye, `\section*{Findings}` and `\textbf{\Large FINDINGS}` will look the same, but the former has embedded meaning, namely, this is where an unnumbered section starts, while the latter just specifies presentation that the eye (actually, our whole image-processing pipeline) needs to interpret and make sense of.

One of the many reasons for using LaTeX is its ability to beautifully typeset mathematical equations. A large challenge in narrating these formulas is that mathematical typesetting is not necessarily semantic, nor even designed to be, and in fact there are ambiguities [11, 14]. LaTeX, and for that matter also handwriting, provides a presentation view of an equation [7]. An expert usually sees what is meant,

Gerd Kortemeyer
doi.org/10.47397/tb/44-3/tb138kortemeyer-accessibility

even if the semantics are sloppily or not explicitly presented. For example, when we see $1/kT$, we will automatically assume $1/(kT)$ instead of T/k, since we are used to seeing those two symbols as essentially conjoined. Or when we see

$$\int_{R_i}^{R_f} \int_0^{2\pi} \int_0^{\pi} \varrho(r,\varphi,\theta)\ r^2 \sin\theta\ dr\ d\varphi\ d\theta \qquad (1)$$

we assume that the first d goes with the limits of the first $\int$, etc., and that whatever appears between the $\int$s and ds is what should be integrated (essentially abusing the span between those symbols as parentheses), even though there is nothing that explicitly says so.

When asked to read the above formula, we might say something like "The integral of the density ϱ as a function of the radius r and the angles φ and θ between the inner boundary R_i ...", instead of "integral sign lower index uppercase r lower index i upper index uppercase r lower index f integral sign lower index ...". By the way, very likely we automatically read "$\varrho(r,\varphi,\theta)$" as "$\varrho$ as a function of r, φ and θ" without even for a moment wondering if that is ϱ multiplied with some three-dimensional vector (r,φ,θ), which would have the same notation.

Arguably, both LaTeX and the conventions of mathematical notation would have allowed a construct like

```
\int_{R_i}^{R_f}{dr}{
   \int_{0}^{2\pi}{d\varphi}{
      \int_{0}^{\pi}{d\theta}{
            \ \varrho{(r,\varphi,\theta)}\
             r^2\sin{(\theta)}
      }
   }
}
```

which would have conveyed functional associations using curly brackets (and by extension semantics) in the source code and be presented as

$$\int_{R_i}^{R_f} dr \int_0^{2\pi} d\varphi \int_0^{\pi} d\theta\ \varrho(r,\varphi,\theta)\ r^2 \sin(\theta) \qquad (2)$$

Realistically, though, probably nobody would go to that kind of trouble; instead, making sense of a conglomerate of symbols is left to the experience of the reader, and yes: his or her intelligence. Can artificial intelligence do the job of making sense of the two-dimensional symbol arrangements and correctly narrate them as formulas?

3 Methodology

To explore the feasibility of using Large Language Models for the translation of LaTeX documents into WCAG-compliant HTML, a test document with some mathematical physics content was generated using REVTeX [18], TikZ [25], BibTeX [8], `\newcommand`, tables, equation references, and the `wasysym` package. Fig. 1 shows the PDF rendering of this document.

The document was translated using the May 24, 2023 release of GPT-4 [19] through the ChatGPT interface, with the following primary prompt:

> *Translate the following LaTeX and BibTeX into screen-reader-accessible HTML, translating the formulas into English (including the indices), and translating labels and citations into anchors and links. For figures, just give a description in English, no code for embedding an image:*

The LaTeX source and the BibTeX source were then pasted below this primary prompt, all three submitted as the prompt, and the generated HTML was copied using the built-in "Copy code" function. This process is illustrated in Fig. 2.

As GPT-4 uses a probabilistic algorithm, every query will result in different HTML rendering; thus, more than one output was generated. The results were analyzed for WCAG-compliance, as well as for the subjective quality of the transcriptions provided by GPT-4.

4 Findings

Figures 3 and 4 show the web browser output rendering the HTML from one query, while Figures 5 and 6 show the result from a second, immediately subsequent query using ChatGPT's "Regenerate response" function. Already at first glance, the outputs are different, which will be explored in more detail when considering the WCAG and physics properties of the responses.

It should be emphasized at this point that the output does not, and in fact should not, look like the PDF in Fig. 1. The "looks" literally do not matter, instead, the HTML needs to be speakable and navigable by screen-reader software, so blind users can listen to the content and find their way around the document structure. Arguably, in terms of Universal Design [12, 20], this is bad practice: the same document should be used by seeing and non-seeing users. Future work may bring satisfactory alignment with the principles of Universal Design; alternatively, a change in publishing practices, suggested in the discussion section, might be called for.

4.1 Document structure

An important component of a screen-reader compatible document is its structure. The screen-reader software builds an index of document sections, so the user can quickly navigate from one segment to another. In HTML, the essential tags are the section headers,

Testing LaTeX Accessibility

Zaphod Beeblebrox
Sirius Cybernetics Corporation
(Dated: June 3, 2023)

This is a document to explore the ability of GPT-4 to translate scientific content written in LaTeX into WCAG-compliant HTML. The document intentionally uses several cumbersome LaTeX constructs while at the same time not claiming to make much physics sense.

I. INTRODUCTION

When GPT-4 first came out, it immediately gained the attention of scientists [1]. Here we test if it can be used to create WCAG-compliant HTML [2] based on LaTeX-source code.

II. SOME USEFUL FORMULAS

When formulating what is called Special Relativity today, Einstein started with the Maxwell equations, like this one [3]:

$$\oiint \vec{E}\cdot d\vec{S} = \frac{1}{\epsilon_0}\iiint \varrho dV \tag{1}$$

Table I lists some of the most commonly used relativistic equations [3].

And then there is one of the most famous formulas of physics [4]:

$$E = \frac{mc^2}{\sqrt{1-\frac{v^2}{c^2}}} \overset{v=0}{=} mc^2 \tag{2}$$

Of course, all of these formulas are merely consequences of the Lorentz transformation between two moving frames of reference. Consider a rotation matrix about the z-axis in three-dimensional space,

$$\vec{x}' = D\vec{x} \tag{3}$$

$$D = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

This is illustrated in Fig. 1.

TABLE I. Some relativistic formulas

Common Name	Meaning	Formula
Time Dilation	Transformation of time-difference between moving frames	$\Delta t' = \frac{\Delta t}{\sqrt{1-\frac{v^2}{c^2}}}$
Length Contraction	Transformation of a length of a moving object	$L' = L\sqrt{1-\frac{v^2}{c^2}}$

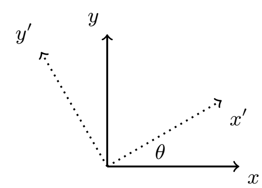


FIG. 1. Rotating a coordinate system.

.

The Lorentz Transformation is a four-dimensional rotation with a matrix ${\lambda^\nu}_\mu$:

$$x'^\nu = {\Lambda^\nu}_\mu x^\mu \tag{5}$$

where we imply the summation convention,

$$x'^\nu = \sum_{\mu=0}^{3} {\Lambda^\nu}_\mu x^\mu \tag{6}$$

III. OTHER REMARKS

One could say a lot more about Special Relativity, but *don't panic* [5]! At the very least, remember equation 2!

[1] K. Sanderson, GPT-4 is here: what scientists think, Nature **615**, 773 (2023).
[2] B. Caldwell, M. Cooper, L. G. Reid, G. Vanderheiden, W. Chisholm, J. Slatin, and J. White, Web content accessibility guidelines (WCAG) 2.0, WWW Consortium (W3C) **290**, 1 (2008).
[3] A. Einstein, Zur Elektrodynamik bewegter Körper, Annalen der Physik **17**, 891 (1905).
[4] A. Einstein, Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig?, Annalen der Physik **18**, 639 (1905).
[5] D. Adams, *The Ultimate Hitchhiker's Guide to the Galaxy: Five Novels in One Outrageous Volume* (Del Rey, 2010).

Figure 1: LaTeX rendering of the test document.

(Prompt text:)

Model: GPT-4

KO Translate the following LaTeX and BibTeX into screenreader-accessible HTML, translating the formulas into English (including the indices), and translating labels and citations into anchors and links. For figures, just give a description in English, no code for embedding an image:

```
\documentclass[aps,reprint,amsmath,amssymb,prper,floatfix]
{revtex4-2}
\usepackage{wasysym}
\usepackage{tikz}

\newcommand{\thesquareroot}{\sqrt{1-\frac{v^2}{c^2}}}

\begin{document}
```

...

```
\section{Other Remarks}
One could say a lot more about Special Relativity, but \textit{don't
panic}~\cite{adams2010}! At the very least, remember
equation~\ref{eq:famous}!
\bibliography{testdoc}

\end{document}

@article{einstein1905a,
 title={{Zur Elektrodynamik bewegter K{\"o}rper}},
 author={Einstein, Albert},
 journal={Annalen der Physik},
 volume={17},
 pages={891--921},
 year={1905}
}
```

...

Response:

Here's a translation of your LaTeX and BibTeX document into screenreader-accessible HTML:

html Copy code

```html
<!DOCTYPE html>
<html>
<head>
    <title>Testing LaTeX Accessibility</title>
</head>
<body>
    <header>
        <h1>Testing LaTeX Accessibility</h1>
        <p>Author: Zaphod Beeblebrox, Sirius Cybernet
        <p>Date: Today's Date</p>
    </header>
```

Figure 2: ChatGPT dialogue.

as marked up by `<h1>...</h1>`, `<h2>...</h2>`, etc. GPT correctly interpreted

```
\begin{document}

\title{Testing \LaTeX{} Accessibility}
\author{Zaphod Beeblebrox}
 \affiliation{Sirius Cybernetics Corporation}
\date{\today}

\begin{abstract}
This is a document to explore ...
```

and inserted structures such as

```
<body>
  <header>
    <h1>Testing LaTeX Accessibility</h1>
    <p>Author: Zaphod Beeblebrox, Sirius ...</p>
    <p>Date: Today’s Date</p>
  </header>
  <section>
    <h2>Abstract</h2>
    <p>This is a document to explore ...</p>
  </section>
```

which not only use the correct section headers, but also, in one of the responses, provide additional semantic markup like the HTML5 tags `<header>` and `<section>`; those potentially helpful tags were missing in the second rendering. However, both versions were perfectly accessible to and navigable by screenreader software.

4.2 New commands

Several of the regular HTML rendering tools fail to interpret `\newcommand` correctly. The test document included

```
\newcommand{\thesquareroot}{\sqrt{1-\frac{v^2}{c^2}}}
```

which was used in nearly all subsequent formulas and interpreted correctly in every case. For example,

```
$L'=L\thesquareroot$
```

was narrated by GPT as

> *L prime equals L times the square root of 1 minus v squared over c squared.*

4.3 Equations

GPT-4 overall did very well in narrating equations. As an example,

```
\begin{equation}
\oiint\vec{E}\cdot d\vec{S} =
   \frac{1}{\epsilon_0}\iiint\varrho dV
\end{equation}
```

was translated in the first version (Fig. 3) into

> *The closed integral of E dot dS equals one over epsilon zero times the triple integral of rho dV.*

and, in the second version (Fig. 5):

Testing LaTeX Accessibility

Author: Zaphod Beeblebrox, Sirius Cybernetics Corporation

Date: Today's Date

Abstract

This is a document to explore the ability of GPT-4 to translate scientific content written in LaTeX into WCAG-compliant HTML. The document intentionally uses several cumbersome LaTeX constructs while at the same time not claiming to make much physics sense.

Introduction

When GPT-4 first came out, it immediately gained the attention of scientists[1]. Here we test if it can be used to create WCAG-compliant HTML[2] based on LaTeX source code.

Some Useful Formulas

When formulating what is called Special Relativity today, Einstein started with the Maxwell equations, like this one[3]:

The closed integral of E dot dS equals one over epsilon zero times the triple integral of rho dV.

Table 1 lists some of the most commonly used relativistic equations[3].

Table 1: Some relativistic formulas

Common Name	Meaning	Formula
Time Dilation	Transformation of time-difference between moving frames	Delta t prime equals Delta t divided by the square root of 1 minus v squared over c squared
Length Contraction	Transformation of a length of a moving object	L prime equals L times the square root of 1 minus v squared over c squared

And then there is one of the most famous formulas of physics[4]:

E equals mc squared divided by the square root of 1 minus v squared over c squared, which simplifies to mc squared when v equals 0.

Figure 3: First HTML rendering of the test document, top part.

Of course, all of these formulas are merely consequences of the Lorentz transformation between two moving frames of reference. Consider a rotation matrix about the z-axis in three-dimensional space, x prime equals D x and D equals cos theta, sin theta, 0 in the first row, minus sin theta, cos theta, 0 in the second row, and 0, 0, 1 in the third row. This is illustrated in Figure 1.

Figure 1: Rotating a coordinate system.

A coordinate system is represented with x and y axes. Two additional axes, x prime and y prime, are represented, rotated from the original x and y axes by an angle theta.

The Lorentz Transformation is a four-dimensional rotation with a matrix lambda nu mu:

x prime nu equals Lambda nu mu x mu

where we imply the summation convention,

x prime nu equals the sum from mu equals 0 to 3 of Lambda nu mu x mu.

Other Remarks

One could say a lot more about Special Relativity, but *don't panic*[5]! At the very least, remember equation 2!

References

- Einstein, Albert. "Zur Elektrodynamik bewegter KÃ¶rper." Annalen der Physik 17.891-921 (1905).
- Einstein, Albert. "Ist die TrÃ¤gheit eines KÃ¶rpers von seinem Energieinhalt abhÃ¤ngig?." Annalen der Physik 18.639-641 (1905).
- Adams, Douglas. The Ultimate Hitchhiker's Guide to the Galaxy: Five Novels in One Outrageous Volume. Del Rey, 2010.
- Caldwell, Ben, et al. "Web content accessibility guidelines (WCAG) 2.0." WWW Consortium (W3C) 290.1-34 (2008).
- Sanderson, Katharine. "GPT-4 is here: what scientists think." Nature 615.7954: 773 (2023).

Figure 4: First HTML rendering of the test document, bottom part.

Testing LaTeX Accessibility

Author: Zaphod Beeblebrox

Affiliation: Sirius Cybernetics Corporation

Date: 3rd June, 2023

Abstract

This is a document to explore the ability of GPT-4 to translate scientific content written in LaTeX into WCAG-compliant HTML. The document intentionally uses several cumbersome LaTeX constructs while at the same time not claiming to make much physics sense.

Introduction

When GPT-4 first came out, it immediately gained the attention of scientists. Sanderson 2023. Here we test if it can be used to create WCAG-compliant HTML based on LaTeX-source code. Caldwell 2008.

Some Useful Formulas

When formulating what is called Special Relativity today, Einstein started with the Maxwell equations, like this one Einstein 1905:

"The surface integral over the vector E dotted with the infinitesimal area vector dS equals 1 over the permittivity of free space times the volume integral over the volume charge density times the infinitesimal volume dV."

Table 1 lists some of the most commonly used relativistic equations Einstein 1905.

Some relativistic formulas

Common Name	Meaning	Formula
Time Dilation	Transformation of time-difference between moving frames	Delta t prime equals Delta t divided by the square root of (1 minus (v squared divided by c squared))
Length		L prime equals L times the square root of (1 minus (v squared divided by c
Contraction	Transformation of a length of a moving object	squared))

And then there is one of the most famous formulas of physics Einstein 1905:

"E equals mc squared divided by the square root of (1 minus (v squared divided by c squared)) when v is not equal to 0. When v equals to 0, E equals mc squared."

Figure 5: Second HTML rendering of the test document, top part.

The surface integral over the vector E dotted with the infinitesimal area vector dS equals 1 over the permittivity of free space times the volume integral over the volume charge density times the infinitesimal volume dV.

Both narratives are adequate and could be spoken by screen-reader software. The first rendering is rather literal, though, as it fails to mention that the first integral is a surface integral, and that $\vec{E}$ and $d\vec{S}$ are vectors. A physicist will understand this concise narrative and probably be thankful that it is not too verbose, while a learner (actually, any learner, blind or seeing) would be thankful for the second, interpretive rendering. It is remarkable that GPT provided the physical meaning of the symbols alongside the narrative of their mathematical connections.

Another example is

```
\begin{equation}\label{eq:famous}
E=\frac{mc^2}{\thesquareroot}
  \stackrel{v=0}{=}mc^2
\end{equation}
```

which is rendered

E equals mc squared divided by the square root of 1 minus v squared over c squared, which simplifies to mc squared when v equals 0.

in one version, and

E equals mc squared divided by the square root of (1 minus (v squared divided by c squared)) when v is not equal to 0. When v equals to 0, E equals mc squared.

in the other. Both are essentially equivalent, and both do a remarkable job of interpreting `\stackrel` (although the full expression is correct regardless of v being zero or not).

Of course, all of these formulas are merely consequences of the Lorentz transformation between two moving frames of reference. Consider a rotation matrix about the z-axis in three-dimensional space,

"The vector x prime equals the matrix D times the vector x."

The rotation matrix D is a 3x3 matrix with elements cos theta, sin theta, 0 on the first row, -sin theta, cos theta, 0 on the second row, and 0, 0, 1 on the third row.

This is illustrated in Figure 1 which shows a rotation of a coordinate system with x, y axes and the rotated x prime, y prime axes.

The Lorentz Transformation is a four-dimensional rotation with a matrix lambda superscript nu subscript mu:

"x prime superscript nu equals lambda superscript nu subscript mu times x superscript mu"

where we imply the summation convention,

"x prime superscript nu equals the sum from mu equals 0 to 3 of lambda superscript nu subscript mu times x superscript mu."

Other Remarks

One could say a lot more about Special Relativity, but *don't panic* Adams 2010! At the very least, remember equation E equals mc squared divided by the square root of (1 minus (v squared divided by c squared))!

References

Einstein, Albert. (1905). Zur Elektrodynamik bewegter Körper. Annalen der Physik, 17, 891–921.

Einstein, Albert. (1905). Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig?. Annalen der Physik, 18, 639–641.

Adams, Douglas. (2010). The Ultimate Hitchhiker's Guide to the Galaxy: Five Novels in One Outrageous Volume. Del Rey.

Caldwell, Ben, Cooper, Michael, Reid, Loretta Guarino, Vanderheiden, Gregg, Chisholm, Wendy, Slatin, John, & White, Jason. (2008). Web content accessibility guidelines (WCAG) 2.0. WWW Consortium (W3C), 290, 1–34.

Sanderson, Katharine. (2023). GPT-4 is here: what scientists think. Nature, 615(7954), 773. Nature.

Figure 6: Second HTML rendering of the test document, bottom part.

A true challenge could have been posed by the rotation matrix:

```
\begin{eqnarray}
\vec{x}'&=&D\vec{x}\\
D&=&\left(\begin{array}{ccc}
\cos\theta&\sin\theta&0\\
-\sin\theta&\cos\theta&0\\
0&0&1
\end{array}\right)
\end{eqnarray}
```

Intriguingly, this was incorporated into the running narrative in one version (Fig. 4),

> *Consider a rotation matrix about the z-axis in three-dimensional space, x prime equals D x and D equals cos theta, sin theta, 0 in the first row, minus sin theta, cos theta, 0 in the second row, and 0, 0, 1 in the third row.*

and spelled out rather literally in the other (Fig. 6),

> *"The vector x prime equals the matrix D times the vector x."*
>
> *The rotation matrix D is a 3x3 matrix with elements cos theta, sin theta, 0 on the first row, -sin theta, cos theta, 0 on the second row, and 0, 0, 1 on the third row.*

Again, both versions correctly describe the situation, and a blind person would be able to envision the matrix in his or her head based on the description.

On the other hand, the equation

```
\begin{equation}
x'^\nu=\sum_{\mu=0}^3\Lambda^\nu_{\ \mu}x^\mu
\end{equation}
```

is the first example of a situation where one of the narratives falls short:

> *x prime nu equals the sum from mu equals 0 to 3 of Lambda nu mu x mu.*

is less helpful than

> *x prime superscript nu equals the sum from mu equals 0 to 3 of lambda superscript nu subscript mu times x superscript mu.*

since it fails to distinguish between covariant and contravariant indices. The nonchalant treatment of indices is also the reason for parenthetically introducing "including the indices" into the prompt; without that phrase, the GPT algorithm recognized the matrix multiplication for what it is and simply stated

> *The transformed x vector equals the sum from mu equals 0 to 3 of Lambda times the original x vector.*

In fact, even before that, it already assumed the summation convention and rendered $x'^{\nu} = \Lambda^{\nu}{}_{\mu} x^{\mu}$ as

> *The transformed x vector equals Lambda times the original x vector.*

This highly concise narration would likely be appreciated by an expert, but might generally be too far from the original.

In one of the renderings, uppercase Λ was replaced by lowercase λ. Again, one needs to remind oneself that it does not matter what the document looks like, but what it "sounds" like: screen-reader software would read both renderings simply as "lambda" without distinction between uppercase and lowercase symbols. In any case, independent of using GPT or more traditional HTML conversion methods, this could be a problem in a document that uses both uppercase and lowercase versions of the same symbol. As it turns out, GPT could recognize this; for example, `\[\Lambda=\sum_i\lambda_i\]` results in "The capital lambda (Λ) is equal to the sum of lambda (λ) over all i" (here, the Greek characters have been rendered as Unicode in the HTML).

4.4 Links and references

Links and references were handled differently in the two renderings. For example, toward the bottom of the LaTeX source, it says

```
One could say a lot more about Special Relativity,
but \textit{don't panic}~\cite{adams2010}!
At the very least, remember equation~\ref{eq:famous}!
```

In the first version, the GPT algorithm had included

```
<p id="eq:famous">
     E equals mc squared divided ... </p>
```

at the place where the formula appeared, as well as

```
<li id="adams2010">
     Adams, Douglas. The Ultimate ... </li>
```

into the reference list, and it rendered the LaTeX as

```
but <i>don't panic</i><a href="#adams2010">[5]</a>!
At the very least, remember equation
<a href="#eq:famous">2</a>!
```

This in fact only works if following the hyperlinks, since neither the equation nor the reference have numbers appearing in the output, as the reference list used an unordered list `<ul>` instead of an ordered list `<ol>`. A blind user, however, could follow those hyperlinks to get to the correct equation and references.

Within the references themselves, some encoding problems are visible with the German umlaut in Fig. 4. However, "visible" is irrelevant, since the real question, common to any non-ASCII characters, is whether or not the screen-reader software handles their encoding (rather than the Safari web browser used to generate Fig. 4 for this visually-oriented paper), which will depend on the product used. As an aside, for the German-language paper titles, WCAG would expect `<span lang="de">...</span>`, which was unfortunately not generated; missing that tag, the screen reader will attempt to read the German with English pronunciation, which makes the considerations about the encoding moot.

The second version uses a different approach:

```
... but <em>don't panic</em>
<a href="#adams2010">Adams 2010</a>!
At the very least, remember equation
E equals mc squared divided by the square root
of (1 minus (v squared divided by c squared))!
```

This approach works for both blind and seeing readers, but actually quoting the referenced equation could become awkward if the equation is a long one; in this case, GPT only quoted the first part, which may not have been what the author wanted to reference. This literature reference also works for seeing readers, since the author name is used. In this version, though, the table reference does not work for any readers, since it simply says

```
Table 1 lists some of the most commonly used ...
```

but the table has no number.

4.5 Tables

The table in the source document was coded as

```
\begin{table}
\caption{\label{tab:formulas}Some relativistic
   formulas}
\begin{tabular}{p{1.8cm}p{4cm}l}
Common Name&Meaning&Formula\\\hline
Time\newline Dilation&Transformation ...
\end{tabular}
\end{table}
```

which, as an example, in one of the HTML versions, GPT interpreted as

```
<table id="tab:formulas">
  <caption>Some relativistic formulas</caption>
  <tr>
      <th>Common Name</th>
      <th>Meaning</th>
      <th>Formula</th>
  </tr>
  <tr>
      <td>Time Dilation</td>
      <td>Transformation ...
</table>
```

Both versions generated by GPT were WCAG-compliant and can be navigated by screen readers. Particularly helpful is that the first row was interpreted as column headers (`<th>` instead of `<td>`), which is not even specified in the LaTeX markup.

4.6 Figures

While GPT-4 itself is multimodal, that is, it can support audio, image, and video input and output, the ChatGPT interface only allows for text-based conversations. The figure in the test document is generated using the popular Ti*k*Z package [25]:

```
\begin{figure}
\begin{tikzpicture}
\draw[thick,->] (2,0) -- (4,0)
           node[anchor=north west] {$x$};
\draw[thick,->] (2,0) -- (2,2)
           node[anchor=south east] {$y$};
\draw[dotted,thick,->] (2,0) -- (3.732,1)
           node[anchor=north west] {$x'$};
\draw[dotted,thick,->] (2,0) -- (1,1.732)
           node[anchor=south east] {$y'$};
\draw(3,0) node[anchor=south east] {$\theta$};
\end{tikzpicture}
\caption{Rotating a coordinate
  system.\label{fig:coord}}.
\end{figure}
```

GPT was prompted to describe this figure, but not draw it. The first version states

```
<figure><figcaption>
        Figure 1: Rotating a coordinate system.
    </figcaption>
    <p>A coordinate system is represented
    with x and y axes.
    Two additional axes, x prime and y prime,
    are represented,
    rotated from the original x and y axes
    by an angle theta.</p>
</figure>
```

which includes semantic tags such as `<figure>` and `<figcaption>`, while the second version has a simpler description that is woven into the narrative:

```
<p>This is illustrated in Figure 1 which shows
a rotation of a coordinate system with x, y
axes and the rotated x prime, y prime axes.</p>
```

This second version correctly recognizes the contents of the figure, but fails to mention the angle θ that is also in the figure. Overall, both descriptions are useful, which is fairly amazing: interpreting Ti*k*Z and subsequently narrating the contents of the resulting image could be considered an emergent ability of GPT (if such things exist).

5 Discussion

Ironically, if this paper gets published as a PDF, it will not be accessible to blind users. How can this be remedied, based on the findings of this exploratory study? In recent years, there has been a strong movement toward Open Access publication; based on this exploratory study, maybe publications should also become Open Source.

Over the coming years, it is to be expected that artificial-intelligence tools will increasingly become personalized agents; this is already suggested by the concepts that assistants like Siri and Alexa put forward, and by new products like Copilot. In this study, the document rendered differently every time, and the prompt was fine-tuned in an attempt to avoid particular interpretations that experienced physicists might in fact prefer. One could argue that AI-assisted sense-making should be under the control of the user, and this means any user, having a disability or not. Thus, instead of mangling the document into a pure presentation format, the source code should be made available, so AI tools can render it in ways preferred by the user.

This would mean free publication, not just in the sense of “free beer”, that is, free-of-charge, but in the sense of “free speech”, to reiterate an old free software adage [9]. Publishing the source code of manuscripts would likely have to go along with adequate licensing terms such as those defined by Creative Commons [5]. Publishers and preprint servers could make a corresponding link available to manuscripts published under those terms.

6 Limitations

The exact results of this study are not reproducible, since the GPT algorithm is not deterministic. Also, constructing a prompt for GPT is essentially trial-and-error, and better or more specific prompts than the one used for this study almost certainly exist. Finally, the study is purely exploratory: while the document was designed to be demanding in terms of LaTeX structures, there was no way to even remotely represent the creativity of LaTeX users. For large documents, GPT might encounter limitations due to its token limit.

7 Conclusion

Overall, GPT was found to be a useful tool to generate screen-reader compatible renderings of LaTeX documents, including document structure, equations, tables, references, and even figures that are based on textual source code. The output is compliant with accessibility standards.

As a probabilistic algorithm, the output is not reproducible, and it was found that each time a document is processed, some elements are translated better or worse than in other versions. Notably, in situations where GPT is able to interpret equations based on its training data in physics, it is capable of generating narratives that are concise and desirable for experts, as well as more elaborate narratives that are suitable for learners.

References

[1] D. Ahmetovic, T. Armano, et al. Axessibility: A LaTeX package for mathematical formulae accessibility in PDF documents. In *Proceedings of the 20th international ACM SIGACCESS conference on computers and accessibility*, pp. 352–354, 2018. See also `tug.org/TUGboat/tb39-3/tb123ahmetovic-axess.pdf`.

[2] Y. Borodin, J.P. Bigham, et al. More than meets the eye: a survey of screen-reader browsing strategies. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, pp. 1–10, 2010.

[3] D. Cervone. MathJax: a platform for mathematics on the web. *Notices of the AMS*, 59(2):312–316, 2012. `doi.org/10.1090/noti794`

[4] D. Cervone, P. Krautzberger, V. Sorge. New accessibility features in MathJax. *The Journal on Technology and Persons with Disabilities*, 4:167–175, 2016.

[5] Creative Commons. Share your work. `creativecommons.org/share-your-work/`.

[6] A. Darvishy. PDF accessibility: tools and challenges. In *Computers Helping People with Special Needs: 16th International Conference, ICCHP 2018, Linz, Austria, July 11–13, 2018, Proceedings, Part I*, pp. 113–116. Springer, 2018.

[7] R. Fateman. How can we speak math. *Journal of Symbolic Computation*, 25(2), 1998.

[8] J. Fenn. Managing citations and your bibliography with BibTeX. *The PracTeX Journal*, 4:2006–4, 2006. `tug.org/pracjourn/2006-4/fenn`

[9] Free Software Foundation. What is free software? `www.gnu.org/philosophy/free-sw.en.html`.

[10] J. Gilligan. Competencies for educators in delivering digital accessibility in higher education. In *Universal Access in Human-Computer Interaction. Applications and Practice: 14th International Conference, UAHCI 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22*, pp. 184–199. Springer, 2020.

[11] T. Groza, S. Handschuh, et al. SALT-semantically annotated LaTeX for scientific publications. In *The Semantic Web: Research and Applications: 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3–7, 2007. Proceedings 4*, pp. 518–532. Springer, 2007.

[12] S. Iwarsson, A. Ståhl. Accessibility, usability and universal design — positioning and definition of concepts describing person-environment relationships. *Disability and rehabilitation*, 25(2):57–66, 2003.

[13] A. Jembu Rajkumar, J. Lazar, et al. PDF accessibility of research papers: What tools are needed for assessment and remediation? In *Proceedings of the 53rd Hawaii International Conference on System Sciences (HICSS)*, 2020.

[14] M. Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[15] G. Kortemeyer. Could an artificial-intelligence agent pass an introductory physics course? *Phys. Rev. Phys. Educ. Res.*, 19:010132, May 2023. `doi.org/10.1103/PhysRevPhysEducRes.19.010132`

[16] A. Manzoor, S. Arooj, et al. ALAP: Accessible LaTeX-based mathematical document authoring and presentation. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–12, 2019.

[17] J.T. Nganji. The Portable Document Format (PDF) accessibility practice of four journal publishers. *Library & Information Science Research*, 37(3):254–262, 2015.

[18] A. Ogawa. RevTeX version 4.0, an authoring package by the American Physical Society. *TUGboat* 22(3):131–133, 2001. `tug.org/TUGboat/tb22-3/tb71ogawa.pdf`

[19] OpenAI. GPT Release Notes. `help.openai.com/en/articles/6825453-chatgpt-release-notes`.

[20] H. Persson, H. Åhman, et al. Universal design, inclusive design, accessible design, design for all: different concepts — one goal? On the concept of accessibility — historical, methodological and philosophical aspects. *Universal Access in the Information Society*, 14:505–526, 2015.

[21] A. Petz, K. Miesenberger. Supporting blind students in STEM education in Austria. In *AAATE Conf.*, pp. 27–31, 2015.

[22] Salt Lake Community College. Screen reader demo. `www.youtube.com/watch?v=q_ATY9gimOM`.

[23] V. Sorge, C. Chen, et al. Towards making mathematics a first class citizen in general screen readers. In *Proceedings of the 11th web for all conference*, pp. 1–10, 2014.

[24] University of California San Francisco. Screen reader demo for digital accessibility. `www.youtube.com/watch?v=dEbl5jvLKGQ`.

[25] Z. Walczak. Graphics in LaTeX using Ti*k*Z. *TUGboat* 20:176–179, 2008. `tug.org/TUGboat/tb29-1/tb91walczak.pdf`

[26] WWW Consortium (W3C). Web content accessibility guidelines (WCAG) 2.0. `www.w3.org/TR/WCAG20/`.

◇ Gerd Kortemeyer
ETH Zurich
Rämistrasse 101
8092 Zurich
Switzerland
`kgerd (at) ethz (dot) ch`
`https://gerdkortemeyer.com/`
ORCID 0000-0001-6643-9428

LaTeX News

Issue 38, November 2023 (LaTeX release 2023-11-01)

Contents

News from the "LaTeX Tagged PDF" project

The multi-year project to automatically tag LaTeX documents in order to make them accessible [3] is progressing steadily (at this point in time mainly as experimental `latex-lab` code).

Just recently we added support for automatic tagging of tabular structures including environments from tabularx and longtable. The code is still in its early stages and lacks configuration possibilities—these will be added in the future.

Approaching an important milestone

Nevertheless, with this new addition we are more or less able to automatically tag any document that confines itself to the commands and environments described in Leslie Lamport's *LaTeX Manual* [1] by simply adding a single configuration line at the top.

In addition, a number of extension packages that go beyond Lamport are already supported, most importantly perhaps amsmath (providing extended math capabilities) and hyperref (enhancing LaTeX with interactive hyperlinking features).

Also already supported are some of the major bibliography support packages such as natbib and biblatex.

For now activation is done through the line

```
\DocumentMetadata
  {testphase={phase-III,math,tabular}}
```

The math and the tabular support are not yet incorporated into `phase-III` but need their own activation, so that we can better experiment with additions and code adjustments.

The `latex-lab` bundle contains various (still untagged) documentation files about the new code that can be accessed with `texdoc -l latex-lab`.

A GitHub repository dedicated to the project

We have also started a new GitHub repository mainly intended for reporting issues, and offering a platform for discussions. For example, there is one discussion on ways to extend the LaTeX `tabular` syntax to allow describing the logical structure of tables (e.g., which cells are header cells, etc.).

Having all issues and discussions related to the project in a single place instead of being spread across multiple repositories such as latex2e, latex3, tagpdf, hyperref, pdfresources, etc., helps people to find information easily and report any issue related to the project without needing to know in which code repository the problematic code resides.

You find this repository at `https://github.com/latex3/tagging-project` and the mentioned discussion on `tabular` syntax at `https://github.com/latex3/tagging-project/discussions/1`.

Your feedback is important and reporting what doesn't yet work is beneficial to all users, so we hope to see you there and thank you for any contribution, whether it is an issue or a post on a discussion thread.

Hooks, sockets and plugs

In previous releases of LaTeX we introduced the general concept of hooks (both specific and generic ones). These are places in the code into which different packages (or the user in the document preamble) can safely add their own code to extend the functionality of existing commands and environments without the need to overwrite or patch them in incompatible ways. An important feature of such hooks is that the code chunks added by different packages can be ordered by rules, if necessary, thereby avoiding problems arising from

LaTeX News #38
doi.org/10.47397/tb/44-3/tb138ltnews38

differences in package loading order. See LaTeX News issues 32–34 [2] for more information.

However, sometimes you need a kind of "hook" into which only a single chunk of code is placed at any time.[1] For example, there is code that implements footnote placement in relation to bottom floats (above or below them). But at any time in the document only one such placement code can be in force. Or consider the extra code needed for making LaTeX documents accessible (e.g., adding tags to the PDF output). Such code is either there (perhaps in alternative versions) or not at all, but it cannot have code from other packages added at the same point interfering with the algorithm.

For these use cases we now introduce the concept of sockets and plugs. A socket is a place in the code into which one can put a plug (a chunk of code with a name) after which the socket is in use; to put in a different plug, the former one has to be taken out first.[2] A socket may or may not have inputs that can then be used by the plugs. While this is technically not much different to putting a command in the code and at some point alter its definition, the advantage is that this offers a consistent interface, allows for status information, supports tracing, etc.

You declare a new socket and possibly some plugs for it with

`\NewSocket{`⟨*socket name*⟩`}{`⟨*# of inputs*⟩`}`
`\NewSocketPlug{`⟨*socket name*⟩`}{`⟨*plug name*⟩`}{`⟨*code*⟩`}`

For example, after the declaration `\NewSocket{foo}{0}` you can immediately use this socket in your code with `\UseSocket{foo}`. The `\NewSocket` declaration automatically defines a simple plug with the name `noop` for the socket and assigns it to the socket (plugs it in), thus your `\UseSocket` sits idle doing nothing[3] until you assign it a different plug, which is done with `\AssignSocketPlug`. This takes the current plug out and puts the new one in. All the declarations and commands are also available in the LaTeX3 programming layer as `\socket_new:nn`, `\socket_new_plug:nnn`, etc.

With this concept we can, for example, add tagging support for the "LaTeX Tagged PDF" project to various packages without altering their behavior if the tagging code is inactive. Activating one or the other form of tagging then just means to assign named plugs to the different sockets.

This is just a brief introduction to the mechanism; for more detailed documentation see `texdoc ltsockets-doc`.

[1] While this is in theory possible to model with the existing hook mechanism, it is inefficient and cumbersome.

[2] Think of electric outlets and plugging something into them.

[3] Sockets with one input also define an `identity` plug and initially assign that to the socket—this means that their input is simply returned without processing.

Document properties and cross-referencing

Traditional LaTeX uses `\label{`⟨*key*⟩`}` to record the values of two "local" properties of the document: the textual representations of the *current page number* and the *current* `\ref` *value* set by `\refstepcounter` declarations [1, p. 209]. (These declarations are issued, for example, by sectioning commands, by numbered environments like `equation`, and by `\item` in an `enumerate` or similar environment.)

These two recorded values can then be accessed and typeset (from anywhere in the next run of the document) by use of the (non-expandable) commands `\ref` and `\pageref` using the *key* that was specified as the argument to `\label` when recording these values. This supported basic cross-referencing (within a document), using these recorded values to provide both page-related and counter-related information (such as the page xvii or the subsection number 4.5.2).[4]

Over the years LaTeX packages have appeared that extend this basic "label-ref system" in various ways. For example, the refcount package made a small but significant change to the functions used to access recorded values, by making them expandable. And the smart-ref package supports the storage of a larger collection of counter values so that, for example, a cross-reference can refer to the relevant chapter together with an equation tag. The cleveref package stores (by means of a second, internal "logical label") extra information such as the name of the counter. The hyperref package adds the `\autoref` command, which tries to retrieve the name of a counter from the *logical name* used for a link target. The tikzmarks library records information about *labelled positions* on the page when using tikz. Finally, the zref package implements many related ideas, including a general idea of properties and lists of properties, with methods to record, and subsequently access, the value of any declared property.

Starting with this release, the LaTeX kernel provides handling of general document properties as a core functionality with standard interfaces. This is based on concepts introduced by the zref package but with some differences in detail, particularly in the implementation. It supports the declaration of new properties, and the recording of the values of any list of properties. These values are retrieved expandably.

To set up a new property that is the current chapter number, for example, here is the declaration to use.

```
\NewProperty{chapter}{now}{?}{\thechapter}
```

[4] In the Spring 2023 release of LaTeX, the `\label` command was extended to record, in addition, both a title (such as the text used in a section head) and the *logical name* used for an associated link target provided these have been set by packages such as nameref or hyperref.

The second argument means that the property value will be recorded immediately ("now"), and not "during the next `\shipout`". The third argument sets a default to be used when, for example, an unknown label is supplied. The final argument contains the code that will, as part of the recording process, be expanded to obtain the value to record for this property.

Then, to record the value of this new property, together with others, use this command.

```
\RecordProperties{mylabel}
                  {chapter,page,label}
```

This records the current values for the properties `chapter`, `page`, and `label`, using `mylabel` as the label, or *key*, for the record.

To *reference* (i.e., retrieve) this recorded value for use in a cross-reference to this chapter, use the `\RefProperty` command with two arguments: the label, or *key*, and the property.

```
\RefProperty{mylabel}{chapter}
```

The LaTeX kernel itself contains declarations for some generally useful properties, including these:

`label` the textual representation of the *current* `\ref` *value*, see above;

`page` the textual representation of the page number for the page currently under construction;

`title` the title, if set by, e.g., nameref;

`target` the logical name of the associated link target, if set by, e.g., hyperref;

`pagetarget` the logical name of the target added by hyperref at the origin of each shipped out page;

`pagenum` the value of the LaTeX counter `page` in Arabic numerals;

`abspage` the absolute page number of the page under construction, i.e., one more than the number of pages shipped out so far (thus it starts at 1 and is increased by 1 whenever a page is shipped out);

`counter` the name of the counter that produced the *current* `\ref` *value*, i.e., the counter that was stepped in the most recent `\refstepcounter` within the current scope;

`xpos, ypos` the position on the shipped out page as set by the most recent `\pdfsavepos`: recording these properties should be done as soon as possible after saving the position.

Both LaTeX 2_ε commands (using camel-case names) and LaTeX3 programming layer commands are provided. For a more complete documentation, see `texdoc ltproperties-doc`.

New or improved commands

Testing for the LaTeX3 programming layer version: `\IfExplAtLeastTF`

The integration of expl3 (the LaTeX3 programming layer) into the kernel means that programmers can use all of the features available without needing to load it explicitly. However, as expl3 is upgraded separately from LaTeX 2_ε and is not a separate package, its version is different from that of LaTeX 2_ε and cannot be tested using `\IfPackageAtLeastTF`. To date, low-level methods have therefore been needed to check for the availability of features in expl3. We have now added `\IfExplAtLeastTF` as a test working in the same way as `\IfPackageAtLeastTF` but focused on the pre-loaded programming layer. Programmers can check the date of expl3 they are using in the `.log`, as it appears both at the start and end in the format

```
L3 programming layer <YYYY-MM-DD>
```

just after the line which identifies the format (`LaTeX2e`, etc.). *(github issue 1004)*

Code improvements

Support for tabs in `\verb*` and `verbatim*`

LaTeX converts a single tab to a single space, which is then treated like a "real" space in typesetting. The same has been true to date inside `\verb`, but was done in a way that meant that they remained as normal spaces even in `\verb*`, etc. We have now adjusted the code so that tabs are retained within the argument to `\verb` and `\verb*`, and the `verbatim` and `verbatim*` environments, independently from spaces, and are set up to print in the same way spaces do. This means that they now generate visible spaces inside `\verb*` and `verbatim*`, and their behavior can be adjusted if required to be different from that of spaces. *(github issue 1085)*

Improved argument checking for box commands

Previously if an alignment option had an unexpected value, such as `\makebox[4cm][x]{text}`, no warning was given but the box content was silently discarded. This will now produce a warning and act like the default `c` alignment. `\framebox` and `\parbox` have a similar change. *(github issue 1072)*

Aligning status of tilde with other active characters

Some time ago we revised the definition of active characters in pdfTeX to allow the full range of UTF-8 codepoints to be used in for example labels, file names, etc. However, `~` was not changed at that point as it is active independent of the engine in use. This has now been corrected: the definition of `~` is an engine-protected one which gives the string version of the character if used inside a csname.

In the programming layer

In the programming layer (expl3), we have revised the behavior of the titlecasing function to enable this to either titlecase only the first word of the input, or to titlecase every word. This should be transparent at the document level but will be useful for programmers.

We have also added the ability to define variables and functions inside `\fpeval` (at the expl3 level this is `\fp_eval:n`). This allows programmers to create non-standard functions that can then be used inside `\fpeval`. For example, this could be used to create a new function `dinner`:

```
\ExplSyntaxOn
\fp_new_variable:n{duck}
\fp_new_function:n{dinner}
\fp_set_function:nnn{dinner}{duck}
                    {duck - 0.25 * duck}
\fp_set_variable:nn{duck}{1}
$\fp_eval:n{duck}
 >\fp_eval:n{dinner(duck)}
  \fp_set_variable:nn{duck}{dinner(duck)}
 >\fp_eval:n{dinner(duck)}
  \fp_set_variable:nn{duck}{dinner(duck)}
 >\fp_eval:n{dinner(duck)}
  \fp_set_variable:nn{duck}{dinner(duck)}
 >\fp_eval:n{dinner(duck)}
$
\ExplSyntaxOff
```

The computation above would then generate

$$1 > 0.75 > 0.5625 > 0.421875 > 0.31640625$$

Users will be able to access added functions without needing to use the expl3 layer. It is possible that a future release of LaTeX will add the ability to create and set floating point variables at the document level: this will be examined based on feedback on the utility of the programming layer change.

Removed *kernel commands*

It is very rare that commands are removed from the LaTeX kernel. However, in this release we have elected to remove `\GetDocumentCommandArgSpec`, `\GetDocumentEnvironmentArgSpec`, `\ShowDocumentCommandArgSpec` and `\ShowDocumentEnvironmentArgSpec` from the kernel. These commands have been moved back to the "stub" xparse provided in l3packages. The reason for this change is that the removed commands exposed implementation details. They were essentially debugging tools which with hindsight should not have been made available directly in the kernel.

Changes to packages in the tools category

longtable: correct p-column definition

In general the longtable implementation follows the array usage but the package didn't take over a change made 1992 in array which adjusted the handling of the strut inserted at the begin of p-columns. As a consequence there are a number of inconsistencies in the output of p-columns between tabular and longtable. This has been corrected; longtable now uses for the strut the same definition as array. *(github issue 1128)*

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual.* Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–38.* https://latex-project.org/news/latex2e-news/ltnews.pdf

[3] Frank Mittelbach and Chris Rowley. *LaTeX Tagged PDF—A blueprint for a large project.* https://latex-project.org/publications/indexbyyear/2020/

Markdown 3 at TUG 2023: Reflections from the Q&A session

Vít Starý Novotný

Abstract

In my TUG 2023 talk, I unveiled the third iteration of the Markdown package for TEX. Following the talk, a number of intriguing questions were raised by attendees, which remained open-ended at that time. This article seeks to address the questions in depth.

1 Introduction

At TUG 2023, I presented the third major version of the Markdown package for TEX. A recording as well as presentation slides are available online [1].

Following the talk, three intriguing questions were raised by attendees:

1. What are the advantages of markdown compared to LATEX?
2. Does the Markdown package support markdown dialects such as GitHub Flavored Markdown?
3. Did I prepare my presentation slides using the Markdown package?

However, due to time constraints, the questions remained open-ended at the time of the conference. In this article, I address the questions more thoroughly and provide the comprehensive answers they deserve.

2 The dubious advantages of markdown compared to LATEX

In the first question of the Q&A session, an audience member expressed doubts about the need for a yet another markup language. Namely, they questioned the readability benefits of using punctuation instead of LATEX commands to denote document structure.

Punctuation was historically used to indicate structure in manuscripts. Markdown's adoption of punctuation adheres to this time-tested convention:

```
# Exotic Fruits Galore!
Discover these *unique fruits*:
- Rambutan: Hairy, red, sweet.
- Jackfruit: Large, sweet pods.
- Salak: Scaly, tangy-sweet.
```

By contrast, LATEX employs commands articulated in natural language, predominantly English:

```
\section{Exotic Fruits Galore!}
Discover these \emph{unique fruits}:
\begin{itemize}
\item Rambutan: Hairy, red, sweet.
\item Jackfruit: Large, sweet pods.
\item Salak: Scaly, tangy-sweet.
\end{itemize}
```

Although LATEX commands are more conspicuous than punctuation, they infuse English elements into non-English documents, which hinders readability, especially to non-English speakers.

During my talk, I presented new syntax extensions supported by the Markdown package such as task lists, superscripts, and subscripts. The audience member pointed out that each new syntax extension added more special characters, which would make the markdown language less predictable to authors.

In the Markdown package, each syntax extension must be deliberately activated. Therefore, when a new syntax extension is introduced to the package, it impacts authors only if they choose to activate it. Furthermore, if authors have portions of their document written without the new extension and are concerned that activating the syntax extension may alter their document, they can conveniently activate the extension for selected content only:

```
% Activate extension only in chapter 2.
\markdownInput              {chapter-1.md}
\markdownInput[newExtension]{chapter-2.md}
\markdownInput              {chapter-3.md}
```

3 Support for markdown flavors other than CommonMark

In the second question of the Q&A session, an audience member remarked that activating syntax extensions for tables makes the language recognized by the Markdown package close to GitHub Flavored Markdown (GFM). They asked whether we planned to further support GFM in the Markdown package.

The Markdown package supports the CommonMark dialect of markdown. Other dialects of markdown such as MultiMarkdown and PHP Markdown Extra have subtle differences in parsing various elements and are not supported by the Markdown package. If you have documents written in other dialects of markdown, you can use a conversion tool such as Pandoc to convert your documents to CommonMark.

For example, here is how you would convert a MultiMarkdown document with tables to CommonMark using Pandoc from the command line:

```
pandoc -i original.md -f markdown_mmd \
 -t commonmark+pipe_tables -o converted.md
```

You would then typeset the converted document using the Markdown package as follows:

```
\markdownInput[pipe_tables]{converted.md}
```

Although GFM is also a dialect, it is a strict superset of CommonMark. Therefore, documents written for the Markdown package with no syntax extensions activated are always valid GFM documents.

Vít Starý Novotný

doi.org/10.47397/tb/44-3/tb138starynovotny-markdown-qa

Furthermore, most extra GFM elements compared to CommonMark such as tables, task lists, and strikethrough are supported by the Markdown package as syntax extensions. Therefore, GFM documents with no other extra elements are valid in the Markdown package with the `pipe_tables`, `task_lists`, and `strike_through` syntax extensions activated.

In GitHub Docs, GFM also supports YAML metadata, which are valid in the Markdown package with the `jekyll_data` syntax extension activated.

If support for the remaining extra elements of GFM [2] is added to the Markdown package in the future, we may add a convenience option `gfm` that would enable all syntax extensions of GFM.

4 Writing presentation slides with the Markdown package for TeX

In the third question of the Q&A session, an audience member asked whether I used the Markdown package to prepare my presentation slides [1].

Markdown can be used to write many different kinds of structured documents, such as books, articles, and technical documentation. Although presentation slides are also structured, they lean towards the visual and somewhat informal style evident in posters and leaflets.

To prepare my presentation slides, I used the WYSIWYG editor of Google Spreadsheets. This allowed me to not only structure the textual content but also to craft the layout of each individual slide.

If your presentation slides are more structured and less visual, you can write them in markdown. Examples of using the Markdown package for presentation slides are given in my previous *TUGboat* article [3, Section 1.2] and elsewhere [4, 5, 6].

5 Conclusion

In my TUG 2023 talk, I unveiled the third iteration of the Markdown package for TeX. My talk generated a series of thoughtful questions from attendees during the ensuing Q&A session. In this article, I revisited the questions and answered them in depth.

At the time of the conference, Markdown 3 was still in development. A stable version of Markdown 3 was released in August [7, 8, 9] and will be included in TeX Live 2024.

References

[1] V. Starý Novotný. Markdown 3: What's new, what's next, July 2023. `youtu.be/U8XjTOhJkg0` and `tug.org/tug2023/program.html`

[2] Github flavored markdown spec, Apr. 2019. Version 0.29-gfm. `github.github.com/gfm`

[3] V. Novotný. Markdown 2.10.0: LaTeX themes & snippets, two flavors of comments, and LuaMetaTeX. *TUGboat* 42(2):186–193, Sept. 2021. `https://tug.org/TUGboat/tb42-2/tb131novotny-markdown.pdf`

[4] V. Novotný. One size does not fit all, Sept. 2021. `overleaf.com/read/mfpwbywkmgnv`

[5] V. Novotný. When tesseract brings friends, Dec. 2021. `overleaf.com/read/gsmdcmrmxrmr`

[6] V. Novotný. Interpretable representations for fast and accurate retrieval of mathematical information, 2022. `github.com/witiko/doctoral-thesis/blob/main/presentation.tex`

[7] V. Starý Novotný, P. Olšák, A. Genčur. Witiko/markdown: Release 3.0.0-alpha, 2023. `github.com/witiko/markdown/releases/tag/3.0.0-alpha`

[8] V. Starý Novotný. Witiko/markdown: Release 3.0.0-alpha.2, 2023. `github.com/witiko/markdown/releases/tag/3.0.0-alpha.2`

[9] V. Starý Novotný, L. Prentice, M. Peischl, D. Willis, T. Vrabcová. Witiko/markdown: Release 3.0.0, 2023. `github.com/witiko/markdown/releases/tag/3.0.0`

◇ Vít Starý Novotný
Studená 453/15
Brno, 638 00
Czech Republic
`witiko (at) mail dot muni dot cz`
`github.com/witiko`

Figure 1: Markdown wolf writes a letter to audience members. Illustration by `fiverr.com/quickcartoon`.

Simplifying LaTeX with ORG-mode in Emacs

Emmanuele F. Somma

Abstract

In writing academic or technical articles, you can reduce complexity by using markup or configurations, dropping the LaTeX *markup language*, and adopting ORG-mode in Emacs while retaining LaTeX's high typographical quality. This article shows how `ORG` markup exported to LaTeX works. Also, it explains the attributes of the most common elements of academic papers, such as links, tables, notes, figures, and bibliographic references. Finally, in the appendix, as a practical project, the author's choices for delivering the original article following the typographical rules of ArsTeXnica are documented.

1 Introduction

`ORG` and *Markdown* are *Lightweight Markup Languages* (or LMLs) for creating documents with a well-designed layout, using only a text editor. ORG-mode, or, as the full name says, "Carsten's[1] outline-mode for keeping track of everything", is a plain text environment for recording whatever is useful in a researcher's ordinary work process (agendas, to-do lists, scattered notes, mind maps, etc.), but its most interesting feature is that it can *export* the content of text files to many typesetting systems through translation *backends*. LaTeX is one of them.

The *Markdown* language was created by John Gruber and Aaron Swartz in 2004 [7], as an evolution of the conventions already used in writing simple texts, such as e-mail messages and newsletters, online forums, wikis, and short pieces of plain text documentation. Commonly considered the progenitor, or at least the archetype, of LMLs, it is certainly the one that has been able to get the highest number of online citations; however, it was in fact developed *after* some previous formulations of light markups and *together* with many others, almost similar. It had the merit of streamlining and extending the disparate features usually present in an LML and making documents more accessible, but also the defect of being implemented in many slightly incompatible dialects [12, 14, 22, 23].

Other LMLs, *AsciiDoc*, *Textile*, *reStructuredText*, *txt2tags*, and *WikiText*, the language of Wikipedia pages, i.e., the most common LML dialects, all have an almost contemporary genesis since the early 2000s. ORG-mode is also part of this group: it was first released in 2003. All were preceded by languages with fewer capabilities, such as *BBcode*, *setext*, and *POD*, which date back to the mid-1990s.

In the wake of the success of *Markdown*, interest in LMLs has grown, and there are now many tools based on this approach. The main characteristic of these languages is simplicity of expression, to the detriment of the variety and completeness of functionality. Their use has not gone beyond the limits of editing short and poorly structured texts for online pages. Wikipedia pages are the best-known examples; they are written with the specialized LML *WikiText* mentioned above, similar to Markdown.

ORG-mode is an exception: it allows broader and more structured elaborations and therefore deserves a presentation even to LaTeX users, who are used to producing high-quality texts.

The experience of using an LML, in the workflow of an editorial staff and then in the complete typesetting of a magazine based on LaTeX, has already been told in 2009 in the pages of ArsTeXnica [19]. In that case, a 2002 LML called `BHL` (*Brute to HTML and LaTeX*) by Bastien Guerry was adopted, a direct ancestor of the modern ORG-mode. At first, `BHL` was adopted for editing articles exported in OpenOffice format, in communication between the editorial and the typographical staffs of LinuxMagazine. Later, it was extended and renamed to `TCHL`, to include functions useful for drafting technical articles in a new editorial initiative, where it was dropped into a new workflow completely based on LaTeX.

However, the needs of a technology business magazine are less than those of an academic article, which is broader and more structured, with footnotes, bibliographic references, inline mathematics, and numbered equations, as well as management of multiple languages and possibly different alphabets, and greater variability and complexity of floating elements, such as figures, boxes, and tables.

The `BHL` and Markdown languages in general are not suitable for these uses. It is essential to turn to ad-hoc typesetting programs and, in the case of markup languages, to more sophisticated languages, such as precisely LaTeX or different versions of XML and SGML, at the cost of greater verbosity of the language and difficulty in managing the whole process, or with the need to adopt user interfaces that are not always easy to use and possibly expensive.

First published in *ArsTeXnica* #34, pp. 21–48 (`guitex.org/home/en/numero-34-aprile-2023`); translation by Carla Maggi. Published by permission.

[1] Carsten Dominik is Professor of Astronomy and director of the Anton Pannekoek Institute for Astronomy of the Faculty of Natural Sciences at the University of Amsterdam. He is also the author of the popular LaTeX reference management package called `RefTeX`.

doi.org/10.47397/tb/44-3/tb138somma-orgmode

Leveraging the benefits of both approaches, the ORG-mode language can be integrated with LaTeX to produce high-quality documents.

2 Brief comparison among markups

To give just three meaningful examples of the relative simplicity of ORG-mode, let us consider how to implement: a) an italicized word, b) an environment for a direct quote, and c) a table.

In the case of italics, in LaTeX we use a markup such as

```
\emph{in italics}
```

using a keyword ('`emph`') and 6 different characters. With ORG-mode we follow an ordinary text syntax and it is sufficient to use the same '`/`' character twice:

```
/in italics/
```

For quotations in display mode, in LaTeX it is possible to use the `quote` or `quotation` environment:

```
\begin{quote}
« Lorem ipsum dolor sit amet, consectetur [...]
\end{quote}
```

In ORG-mode, the *block* `quote` is used.

```
#+BEGIN_QUOTE
« Lorem ipsum dolor sit amet, consectetur [...]
#+END_QUOTE
```

To write a quote in a language with a non-Latin alphabet (e.g. Greek), it is sufficient to use a couple of *attributes* to specify the language, namely '`:environment foreigndisplayquote`' and '`:options {greek}`', and the right environment in the preamble:

```
#+LATEX_HEADER:\usepackage[autostyle=true]
#+LATEX_HEADER:           {csquotes}
```

to get something like:

> ἀγεωμέτρητος μηδεὶς εἰσίτω.

Lastly, for the typesetting of tables in LaTeX, between keywords and specification characters, seven different elements are used:

```
\begin{center}
\begin{tabular}{lll}
Header 1 & Header 2 & Header 3\\
\hline
Cell 11 & Cell 12 & Cell 13\\
Cell 21 & Cell 22 & Cell 23\\
Cell 31 & Cell 32 & Cell 33\\
\end{tabular}
\end{center}
```

whereas the same table in ORG-mode notation is much more parsimonious:[2]

```
| Header 1 | Header 2 | Header 3 |
|----------+----------+----------|
| Cell 11  | Cell 12  | Cell 13  |
| Cell 21  | Cell 22  | Cell 23  |
| Cell 31  | Cell 32  | Cell 33  |
```

in both cases, the result is the same:

Header 1	Header 2	Header 3
Cell 11	Cell 12	Cell 13
Cell 21	Cell 22	Cell 23
Cell 31	Cell 32	Cell 33

`ORG` markup is thus much more concise than LaTeX markup, and many aspects can be modified in the export of individual elements, making it possible to cover many, if not all, of the most common use cases in the writing of an academic article.

3 Integration with Emacs

The comparison between ORG-mode and LaTeX can proceed along two lines: a) comparing the markup language, as just done and as we'll see more in detail from the next section onwards, and b) comparing the integration with the Emacs editor [20].

LaTeX does not enforce the use of any specific editor but this is not the case for ORG-mode, which is closely related to Emacs. There are, it is true, alternative implementations,[3] but from the point of view of the production of most academic papers ORG-mode and Emacs are still one and the same.

On the other hand, it is also true that the *Emacs modes* for editing LaTeX files (and groups of files) and `ORG` are both equally powerful. Any comparison on this point would only involve design choices and implementation details of one or the other. As Emacs users well know, this editor is particularly effective at drafting texts for the various markup languages, as well as for programming ones. We cannot boast of any substantial difference in the use of ORG-mode compared to Emacs's LaTeX-mode [1, 3].

[2] It should also be noted that ORG-mode has a table editing environment that allows you to interact with these text elements as if they were in a spreadsheet, including the calculation of fields using formulas.

[3] See the extensive list of systems given in [16]. To give a single example, visualization of `ORG` markup as web pages has been integrated into Github, which has thus become a convenient tool for online publications using `ORG` markup without any mediation by Emacs.

Nevertheless, they are related: to use the ORG-mode pagination subsystem for the production of PDF documents, LaTeX still needs to be correctly installed and configured. Emacs, and consequently ORG-mode, can adapt, practically without intervention, to a well-made LaTeX installation, such as those now obtained from the common distributions.[4] This usually results in having the `ORG` system for LaTeX immediately available after the successful installation of LaTeX and Emacs.

Even without an installed LaTeX system, it is always possible to produce a typeset document by exporting the text of the `ORG` file in other formats (such as ODT, HTML, or plain text), and from these, discounting the lower quality in typesetting, a PDF.

Text files with the `ORG` markup are identified by the extension '`.org`'. As usual, when Emacs loads a file with a known extension, it starts interacting with the appropriate *major mode* (that's where the name ORG-mode comes from: it's the *major mode* by which Emacs interacts with `.org` files).

A major mode in Emacs activates one or more ad hoc menus and a series of specific *key combinations*, which can be used only when editing a file with that extension. For example, when editing a `.org` file there will be an '`Org`' menu, and when the cursor is inside a table, a further menu for editing tables will also appear, for adding columns, rows, and other operations, as mentioned earlier.[2]

In ORG-mode some interactive functions and related key combinations are fundamental in the work process. For example, `C-c C-e` (the two keystrokes CTRL-C and CTRL-E; the abbreviation to `C-` is customary in Emacs), which corresponds to the command `M-x org-export-dispatch`, opens the panel where the text can be exported in various output formats. From here you choose the export backend and launch the translation operation; for example, the combination '`l p`' creates a PDF file.

Furthermore, `C-u C-c C-e` repeats the last used typesetting command. After the first use of the previous panel, where the desired export is selected, `C-u C-c C-e` becomes the most used key combination in the typesetting phase.

The complete combination `C-c C-e l p` (`M-x org-latex-export-to-pdf`) transforms `.org` source into LaTeX and then typesets it to PDF using pdfLaTeX (or the user's chosen program). If, instead, you use `C-c C-e l o`, at the end of the compilation the system viewer will also open to show the typeset result in PDF.

[4] See also [5].

Thus, if properly configured, ORG-mode performs all the necessary, and possibly multiple, LaTeX compilations to correctly obtain indexes, bibliographies, and references, without the user having to do anything else.

You may find it useful to customize the PDF compilation process with the `latexmk` program (or similar) by specifically configuring the Emacs variable `org-latex-pdf-process` in `~/.emacs`, as in:

```
(setq org-latex-pdf-process (list
  "latexmk -shell-escape -bibtex -f -pdf %f"))
```

Another useful key combination is `C-c C-`; this opens a panel where you can choose a block type to insert (for example, you can use `e` for an *example*, `s` for a program listing in *source* code, `q` for a *quote*, `v` for poetic *verse*, etc.). Many other key combinations are defined to perform specific tasks, such as inserting links (`C-c C-l`), footnotes (`C-c C-x f`), quotes (`C-c C-x [`), and so on.

4 ORG for LaTeX

To obtain a LaTeX file starting from a `.org`, there is no need for any particular operations other than exporting to the `.tex` source file with `C-c C-e l l`.

With ORG-mode, the exported LaTeX document will be an article (class `article`) by default. You can configure this export in a more refined way.

There are two possibilities for configuring ORG-mode operations: a) through the Emacs customization (or configuration) mechanism,[5] which applies to all `ORG` documents, or b) with the definition of *directives*[6] in the `.org` file being processed, which obviously only apply to the file being processed.

[5] In Emacs, we make a distinction between *configuration* and *customization*, although the goal and the final result may be indistinguishable. By *configuration* we mean the introduction in the Emacs initialization file (usually `~/.emacs`) of commands in the Lisp programming language, usually very simple, for activating the packages and defining the *customization variables*. By *customization*, instead, we mean the use of the hierarchical structure of the customization panels of the `Options->Customize` menu to define the same *customization variables* and, in turn, generate a structure within the `.emacs` file based on the `custom-set-variable` Lisp function. The two options have equivalent results; adopting one or the other form, or a mix between the two, is a matter of user preference.

[6] In this article, to make the difference more understandable, we adopt the (uncommon) convention of naming as *directives* the configuration lines present inside the `.org` file (which the documentation usually calls *variables* or `ORG` *commands*), reserving the wording *customization variables* (or simply *Emacs variables*) for the variables at the editor level. Directives will always be represented between the characters `#+` and a `:`, which are necessary for their definition and will always be capitalized, although this is not strictly necessary. To make clear the distinction between *customization variables*, which can be configured by the user, and Emacs *interactive functions*, i.e. commands that can be executed by the user,

The first strategy is suitable for changes to the whole work environment, and is based on the customization mechanism of the `Options->Customize->Group->Specific Group` menu, indicating `org` as a group, or with the command `M-x customize-group RET org`. Alternatively, special Lisp command lines can be introduced in the `~/.emacs` file.

In contrast, in the second case, the configurations can be indicated directly in the `.org` file using `ORG` markup *directives*, which have the form:

```
#+<KEY>: <VALUE>
```

The key is case-insensitive.

Thus, the same effect as changing the variable `org-latex-default-class`, which defines the LaTeX class to be used in the `\documentclass` command at the beginning of the `.tex` file, is obtained using the `#+LATEX_CLASS:` directive in the `.org` file:

```
#+LATEX_CLASS: report
```

If you want to add options to the class, the directive `#+LATEX_CLASS_OPTIONS:` is used:

```
#+LATEX_CLASS_OPTIONS: [12pt,oneside]
```

The `#+LATEX_COMPILER:` directive specifies the compiler to employ.

```
#+LATEX_COMPILER: xetex
```

Lastly, one or more `#+LATEX_HEADER:` directives can be used to include extra lines in the preamble of your LaTeX document. They provide a convenient way to add more packages without having to configure `org-latex-default-packages-alist`, and also to possibly define new commands or particular configurations for LaTeX.

The conversion from `ORG` to LaTeX depends on the `org-latex-classes` variable, which is defined as follows:

```
(("article" "\\documentclass[11pt]{article}"
  ("\\section{%s}"       . "\\section*{%s}")
  ("\\subsection{%s}"    . "\\subsection*{%s}")
  ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
  ("\\paragraph{%s}"     . "\\paragraph*{%s}")
  ("\\subparagraph{%s}"  . "\\subparagraph*{%s}"))
 ("report" "\\documentclass[11pt]{report}"
  ("\\part{%s}"          . "\\part*{%s}")
  ("\\chapter{%s}"       . "\\chapter*{%s}")
  ("\\section{%s}"       . "\\section*{%s}")
  ("\\subsection{%s}"    . "\\subsection*{%s}")
  ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))
 ("book" "\\documentclass[11pt]{book}"
  ("\\part{%s}"          . "\\part*{%s}")
  ("\\chapter{%s}"       . "\\chapter*{%s}")
  ("\\section{%s}"       . "\\section*{%s}")
  ("\\subsection{%s}"    . "\\subsection*{%s}")
  ("\\subsubsection{%s}" . "\\subsubsection*{%s}")))
```

the name of the latter will always be prefixed with the key combination `M-x` necessary to activate them.

Each element of this list, defined as:

```
(class-name preamble-start
            segmentation-elements)
```

represents one of the usual classes of LaTeX documents (`article`, `book`, and `report`).

The *segmentation elements* of the definition are pairs of titling commands, the first to be used in case of numbered titles, and the second for unnumbered ones. As you can see, for `article` the segmentation starts from the `section`, while for `book` and `report` it starts from `part`, but this can clearly be changed.

When exporting to LaTeX, the backend will apply, before anything else, the class defined globally in `org-latex-default-class` or in the document with the directive `#+LATEX_CLASS:`, whose options are in `org-latex-default-class-options` or with the directive `#+LATEX_CLASS_OPTIONS:`. The default packages, which are listed in the variable `org-latex-default-packages-alist` are inserted later; they can be omitted using the special string `[NO-DEFAULT-PACKAGES]` in `preamble-start` of the class defined in `org-latex-classes`.

Other packages, given in `org-latex-packages-alist`, are also appended later, and can be omitted with `[NO-PACKAGES]`. Finally, the `EXTRA` text will be added, i.e. the lines defined in the `.org` file with the `#+LATEX_HEADER:` directives. This set can also be excluded using `[NO-EXTRA]`. The preamble of an article for ArsTeXnica is shown in the appendix.

The `org-latex-default-packages-alist` variable includes `inputenc`, `fontenc`, `hyperref`, and other packages that are needed for various ORG-mode functionalities. It should not be overridden.

To work with a different LaTeX class, for example, `arstexnica`, the class used for this article (in the Italian original), you will need to properly configure `org-latex-classes`, as shown in the appendix.

Whenever possible, it is preferable to act at the level of the `.org` file, to keep the compilation needs documented in the file itself and not depend on a general configuration that could change over time or between different users.

Only those directives dedicated to LaTeX are indicated, but many other generic directives affect the export. For example, document descriptive directives such as:

```
#+TITLE: Emacs ORG-Mode, LaTeX (and ArsTeXnica)
#+AUTHOR: Emmanuele F. Somma
#+CREATOR: Emacs 28.1 (Org mode 9.5.2)
#+DESCRIPTION: An article for ArsTeXnica
#+KEYWORDS: Emacs ORG LaTeX typesetting
```

are appropriately used by the export backend:

```
\author{Emmanuele F. Somma}
\date{\today}
\title{Emacs ORG-Mode, \LaTeX{} (and \ArsTeXnica{})}
\hypersetup{
 pdfauthor={Emmanuele F. Somma},
 pdftitle={Emacs ORG-Mode, LaTeX (and ArsTeXnica)},
 pdfkeywords={Emacs ORG LaTeX typesetting },
 pdfsubject={An article for ArsTeXnica },
 pdfcreator={Emacs 28.2 (Org mode 9.5.5)},
 pdflang={Italian}}
```

Note also the definition of the language in which the document is written:

```
#+LANGUAGE: it
```

The supported languages are in the variable org-latex-babel-language-alist. However, the language definition does not automatically include the related LaTeX package, so that the user can choose whether to use babel or polyglossia. This can be defined in the general list of default packages with the Lisp command:

```
(add-to-list 'org-latex-packages-alist
  '("AUTO" "babel"
    t ("pdflatex" "xelatex" "lualatex")))
```

or

```
(add-to-list 'org-latex-packages-alist
  '("AUTO" "polyglossia"
    t ("xelatex" "lualatex")))
```

where AUTO indicates the language chosen in the #+LANGUAGE: directive. As you can see, they can be differentiated by typesetting engine.

Alternatively, the language package can be inserted explicitly in the .org file with a line like:

```
#+LATEX_HEADER: \usepackage[italian,greek]{babel}
```

5 Document options

Some ORG document options affect LaTeX export. Options are defined in the format key:value and inserted in the directive named #+OPTIONS:.[7] Here is a lengthy example (we won't describe them all here; see the documentation):

```
#+OPTIONS: ':nil *:t -:t ::t <:t H:3 \n:nil ^:t
#+OPTIONS: author:t broken-links:nil creator:nil
#+OPTIONS: d:(not "LOGBOOK") date:t e:t
#+OPTIONS: email:nil f:t inline:t num:t
#+OPTIONS: prop:nil stat:t tags:t tasks:t tex:t
#+OPTIONS: timestamp:t title:t toc:t todo:t |:t
```

Boolean variables are *true* if set to t, and *false* if set to nil. Here are some useful options:

num: turns on section numbering

toc: typesets the table of contents[8]

|: turns on table typesetting

^: uses TeX syntax for superscripts and subscripts[9]

-: allows conversion of special strings

f: activates conversion of footnotes (indicated as [fn:...])

: activates text emphasis (for bold, / for italics, _ for underlined)

TeX: allows TeX macros in the text

author: includes author's name in the layout

email: includes author's email in the layout

creator: includes creation information in the layout

6 Element attributes

The directives and options seen so far apply to the entire export process and determine how the layout is formed. Usually, ORG-mode performs a standard layout of the elements of the LaTeX document, but this is not always sufficient for the author's purposes. It is possible to provide particular specifications for the various elements to be typeset, by defining attributes for the individual elements to be typeset (such as images, tables, etc.). Attributes are defined in ORG with a #+ATTR_<backend>: directive. For LaTeX it is #+ATTR_LATEX:.

For the same element, several attribute directives can be specified, each relating to different backends, to be used in the corresponding exports. And in each #+ATTR_... directive, many attributes can be included, in the format :<attribute> <value>. Element attributes should not be confused with document options, which are defined in the #+OPTIONS: directive, in the format <option>:<value>.

To give an example, if you want to introduce an image into the text flow, you can write:

```
[[./img/arstexnica.png]]
```

the following image will be obtained in the typeset text, extended to the page size:

[7] With the key combination C-c C-e # or the command M-x org-export-insert-default-template you can insert all options into your document. By choosing default, you can insert the options common to all backends; with latex, html, etc. inserting those related to LaTeX, HTML, etc.

[8] The table of contents is located after the title or at the point where a line with only the string [TABLE-OF-CONTENTS] is inserted.

[9] Warning: it is not a boolean flag but must be set as ^:{}; in this case, a_{b} is interpreted, whereas a_b is not.

We can define attributes for the image, such as the size or a possible rotation angle, as follows:

```
#+ATTR_LATEX: :width 4cm :options angle=45
[[./img/arstexnica.png]]
```

and you will get:

You can put the image in a floating block and give it a caption with the general command `#+CAPTION:`, which is valid for all export formats:

```
#+CAPTION: The ArsTeXnica logo
#+ATTR_LATEX: :width 4cm
[[./img/arstexnica.png]]
```

Figure 1: The ArsTeXnica logo

Many elements of the ORG markup have specific attributes, like the ones we have just seen for images. One of the most important is `:environment` which selects the environment to apply to the next element. For example:

```
#+ATTR_LATEX: :environment myverbatim
#+BEGIN_EXAMPLE
«  Lorem ipsum...
#+END_EXAMPLE
```

It will apply the `myverbatim` environment to the `EXAMPLE` block:

```
\begin{myverbatim}
« Lorem ipsum...
\end{myverbatim}
```

although, in this case, it would be even more appropriate to take advantage of the special syntax available to specify the block with the name of the environment:

```
#+BEGIN_myverbatim
« Lorem ipsum...
#+END_myverbatim
```

Using the attributes it is possible to specify the details of the typesetting without directly resorting to the LaTeX commands.

7 External and internal links and element identification

A URI, written verbatim in the document, or enclosed in angle brackets (`<LINK>`) or in double square brackets (`[[LINK]]`), is typeset as an active link in LaTeX. The general format of a hyperlink is:

```
[[LINK][DESCRIPTION]]
```

In this case, the description will be the typeset text, and the link will allow you to access the resource. There are precise rules to correctly represent within links the elements that need to be escaped using the character (\), such as square brackets, for example, and the escape character itself. The best way to insert and modify a URI in the text is to use the key combination provided by Emacs (`C-c C-l`) which automatically takes care of this aspect.

In addition to the external links, using all the usual schemes (`http`, `https`, `ftp`, `email`, etc.), ORG-mode also handles links *internal to the document*, such as references to figures and tables, to sections, and even to single items of a list.

An article element, such as a table or a figure, can be identified with the `#+NAME:` directive placed before the element. This name can then be used for internal references; for example:

```
#+NAME: Tab1
| Col1 | Col2 |
|------+------|
| A    | B    |

As indicated in *Table [[Tab1]]*.
```

which will be typeset like this:

Col1	Col2
A	B

As indicated in **Table 7**.

It is also possible to refer to the single items of a numbered list:

```
1. first item
2. <<p2>>second item

As indicated in item [[p2]].
```

which produces:

1. first item
2. second item

As indicated in item 2.

It is also possible to link to a section title with the notation *Section [[*Mathematics]]* (obtaining **Section 10**). The numbering option (num:t) must be active.

8 Some important block types

Various specialized blocks are delimited by the directives #+BEGIN_<type> and #+END_<type>. The #+BEGIN_ line can also specify options for the block. Common examples:

ABSTRACT the ABSTRACT block plays a special role because it is typeset as a summary of the document (*abstract*), according to the rules of the class used in the LaTeX document. For example, the ABSTRACT block of this article is:

```
#+BEGIN_ABSTRACT
In writing academic ...
#+END_ABSTRACT
```

CENTER center the block content on the page.

EXPORT a particularly important block, the specification key indicates the backend system (html, latex, etc.). The content will only be included in exported files for that backend. So to copy a piece of LaTeX code from the .org file to the .tex, we write:

```
#+BEGIN_EXPORT latex
<LaTeX code> ...
#+END_EXPORT
```

EXAMPLE block for showing examples in verbatim:

```
#+BEGIN_EXAMPLE
| Header 1 | Header 2 | Header 3 |
|----------+----------+----------|
| Cell 11  | Cell 12  | Cell 13  |
| Cell 21  | Cell 22  | Cell 23  |
| Cell 31  | Cell 32  | Cell 33  |
#+END_EXAMPLE
```

QUOTE block for typesetting display quotations.

SRC block for including source code in most common programming languages, as indicated in the first option of the begin directive.

A relevant aspect of ORG documents is that these code blocks are used not only to represent an example (for which there is also the block type EXAMPLE) but can be made *active*. Then ORG-mode, with the support of external or internal language interpreters, can in fact execute the code of the block and insert the result of the processing in the output, using the directive #+RESULT: placed directly after the SRC block.

Alternatively, the code can be exported to external source files, and subsequently interpreted or compiled. This allows a .org document to become an active *literate programming notebook* in the same way as the classic tangle and weave tools, with which TeX was written by Donald Knuth [9, 10], or the more recent *notebook interface* projects like the Jupyter project [4].

For the LaTeX representation of the SRC block, the verbatim environment is used, but it is also possible to choose listings or minted by configuring the Emacs variable org-latex-listings.

Here's an example:

```
#+BEGIN_SRC python
def fibonacci(n):
  if n <= 1: return n
  else: return fibonacci(n-1)+fibonacci(n-2)
#+END_SRC
```

VERSE block for typesetting poems and verses:

```
#+BEGIN_VERSE
There once was a student of science,
Whose essays cast off in decadence
She then used Org to write,
And LaTeX to paginate,
So her colleagues admired in silence.
#+END_VERSE
```

9 LaTeX inside ORG markup

ORG is a format that tends to represent text content unaltered, except for markup elements placed in the expected positions or combinations. Anything not recognized as markup is transferred, as is, to the exported file.

It is possible to include text intended for the backend. For example, you could include any LaTeX command, say \raggedright, to get the desired LaTeX behavior (the TeX:t option must be defined).

Working with a single backend, and therefore not needing to export the result in any other format, there is no need to use the EXPORT block. If, on the other hand, you want to take advantage of ORG-mode's ability to export content in multiple formats, you need to limit the export to just the desired backend, using EXPORT blocks when necessary.

If you need to insert shorter fragments of LaTeX you can use an inline expression enclosed within a double pair of symbols @ followed by latex:, as in @@latex: arbitrary LaTeX code@@. This code will only be inserted in the export for the selected backend. The same can be done for other backends:

```
...this document is typeset with
@@latex: \LaTeX{}@@@@html: HTML@@
and not with @@latex: HTML@@@@html: \LaTeX{}@@
```

Given this sentence in this article, when readers view the typeset PDF, they will see that this document is typeset with LaTeX (and not in HTML); if they see it on a web page, exported directly from ORG-mode in HTML, they would read the opposite.

If you don't need a whole block but just a single line, you can use the `#+LATEX:` directive, as in:

```
#+LATEX: \newpage
```

10 Mathematics

One of the strengths that make LaTeX supporters rightly proud is the advanced ability to typeset documents with mathematical expressions, both inline and displayed. From this point of view, ORG uses the most straightforward approach: the LaTeX notation for mathematics. Therefore, inside the `.org` source an expression like the following:

```
\[ \hat{y} = \hat{\beta}_{0}+\hat{\beta}_{1}
    x_{1}+\hat{\beta}_{2}x_{2}+\dots
    +\hat{\beta}_{p}x_{p} \]
```

produces:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p$$

For a numbered equation, it's enough to use an `equation` block or any other such environment usually used in LaTeX, e.g., `displaymath`, `align`, etc.

For inline math, you can also simply use the LaTeX notation, such as `\( x^{2+y} \)` to obtain x^{2+y}. From this point of view, therefore, ORG-mode does not add or subtract anything to/from LaTeX.[10]

Where the backend provides for it, the mathematics will be typeset correctly even when exported in formats other than LaTeX, for example, HTML, as you can see using the key combination `C-c C-e h O`.

11 Tables

Tables were already introduced above; we need only mention a few attributes:

`:environment` environment name for the LaTeX backend (`tabularx`, `longtable`, `array`, etc.);

`:float` typesets the table as a float, with options `sideways` or `multicolumn` to obtain it respectively rotated or on several columns;

`:mode` modifies the way the table is exported; takes values `table`, `math`, `inline-math`, `verbatim`, or `tabbing`;

`:placement` specifies the positioning of the table; it can take the values `h t b p ! H`, as in LaTeX.

Other options concern:

- export in a math environment: `:math-prefix`, `:math-suffix`, `:math-arguments`;
- the width: `:spread`, `:width`;
- switches to improve handling of the layout: `:booktabs`; `:center` for centered positioning; and `:rmlines` to remove all but the first line for tables typeset in the format expected by `table.el`.

The table format called `table.el` is an alternative Emacs format. It is a little less parsimonious than the standard ORG format, because you need to indicate all the intersections of the cells, but it allows you to typeset more complex tables than in the simplified ORG-mode syntax. For example, a table with merged cells, like this one:

<table>
<tr><td colspan="2">A</td><td rowspan="2">B</td></tr>
<tr><td>10</td><td>11</td></tr>
<tr><td>20</td><td>21</td><td>22</td></tr>
</table>

is written in the text file as:

```
+-----+-----+-----+
| A         | B   |
+-----+-----+     |
| 10  | 11  |     |
+-----+-----+-----+
| 20  | 21  | 22  |
+-----+-----+-----+
```

12 Footnotes

ORG-mode supports two ways to write footnotes.

1. In the first case, an inline label is used in the format `[fn:NAME]` and the footnote itself will be expressed as an autonomous paragraph, usually at the end of the file or of the section considered, in the format '`[fn:NAME] Footnote text.`'.[11] The definition must begin with the first square bracket in the first column of the line, with no preceding spaces, and ends at the start of a new note or section or with two blank lines.
2. The second case allows insertion of an inline footnote with the format `[fn::Inline footnote text...]` — pay attention to the double `:`. You can also use the notation `[fn:NAME: Inline text]` if you need to refer to the footnote elsewhere as well, as was the case in this article for

[10] Remember that the use of the TeX notation, `$...$` or `$$...$$`, for equations, while possible, is not recommended in LaTeX and, even more so, in ORG-mode.

[11] Footnote text.

the footnote at the end of this sentence, used already twice elsewhere.[2]

Footnotes are automatically numbered.

If you do not want to place your footnotes at the end of the page but at the end of the document, you can introduce the following LaTeX directives where you want the footnotes to appear, at the end of the document:

```
#+LATEX_HEADER: \usepackage{endnotes}
#+LATEX_HEADER: \renewcommand\footnote{\endnote}
#+LATEX: \theendnotes
```

13 Bibliographies and references

Citing, linking, and listing references is one of the fundamental tasks in writing scientific articles and, for a long time, ORG-mode did not have its own standard system for dealing with them. In recent years, this situation has changed, although in a way that may not entirely satisfy an author accustomed to LaTeX.

The most *modern* way in `ORG` of indicating bibliographic citations is entrusted to the `org-cite` subsystem (present in the `oc.el` library), already included in up-to-date ORG-mode distributions.

`org-cite` is based on the definition of *citation processors*, whose job is to offer several functionalities including:

- colored links and tooltips on hover;
- actions on user click in the editor;
- insertion and modification of citations;
- export to different formats.

The ORG-mode quoting mechanism, which was designed to be *back-end agnostic*, introduces a triad consisting of

```
(processor bibliographic-style citation-style)
```

The *citation style* consists of a string based on a uniform pattern, set up as follows:

```
[cite/s/v:@key1;@key2;...]
```

where the two specifiers `/s` and `/v` are optional and represent the *style* and *variant* indicators of the citation. They can be used to select a citation without author (`na` for `noauthor`), i.e. with date only, or in a variant with initial capitalization (`cf` for `caps-full`). They are equivalent to many `biblatex` citation commands, such as `\cite`, `\citetitle`, `\citeyear`, etc.

Following that, `@key1`, `@key2` specify the *citation keys*; there can be only one (in which case the final semicolon is not needed) or many, separated by semicolons. Each citation key is composed of the character `@` followed by the label of the reference used in the `.bib` file, preceded by a possible prefix, which will be typeset before the reference and followed first by a *locator*, i.e. a page, chapter or volume marker or other identifiable elements in the reference and, after a space, a suffix. These elements are optional. The general form of a citation key is therefore as follows:

```
prefix @key locator suffix
```

At the beginning of the entire citation string, there may be a prefix without a citation key, which acts as a common prefix for all citations. Likewise, there may also be a suffix common to all citations. Ultimately, a citation expressed as:

```
[citep/text/full:See;
 @hegel1807phenomenology pp. 184--6]
or
[citep:@hegel1807phenomenology]
```

could be typeset as:

> (See Hegel, The Phenomenology of Spirit (1807) pp. 184–6)
> or
> (Hegel, 1807)

Several processors have been defined, but the main variants are:

`csl` for exporting citations to all backends;

`bibtex` only for LaTeX; there is also `natbib` and `biblatex`, which differ from `bibtex` primarily in the LaTeX package loaded.

The `bibtex` processor uses LaTeX, so `[cite:@hegel1807phenomenology pp. 184--6]` becomes `\cite[184--6]{hegel1807phenomenology}`. In this case, LaTeX will also use `bibtex` (or `biber`, by user choice) to generate the citations. The downside of using this processor is that it *cannot* be used to typeset in formats other than LaTeX.

To have a universal `.org` file you need to use the `csl` processor (Citation Style Language). `CSL` is an XML-based citation and bibliography specification language. It is used by some well-known bibliographic reference management programs such as Zotero, Mendeley, and Papers, and was initially developed within the OpenOffice Bibliographic Project.[12]

The choice of one or the other approach is based on whether or not to rely on `bibtex` for the handling of LaTeX citations. If `csl` is used, both citations and bibliographic references will be composed in a plain text way, according to the chosen bibliographic standard: there will be no `\cite` citation commands

[12] `openoffice.org/bibliographic/`

in the file, nor a \bibliography to read the file created by bibtex.

Those who have mastered bibtex sufficiently, especially in selecting the citation variants or if they have to manage lesser-known fields of the .bib file (such as the crossref and related fields, used to indicate related volumes or language translations), know how powerful it is to rely on the bib(la)tex citation system. The agnostic approach to the csl backend may therefore not be adequate for sophisticated LaTeX enthusiasts.

It is also worth mentioning an alternative package, created by John Kitchin, which was originally used as a template for org-cite, called org-ref. Although it is a package that needs to be installed via M-x package-list-package, it is, still nowadays, the most used system for typesetting citations and bibliographies in LaTeX.

org-ref is comparable to the bibtex (and related) processors of org-cite but contains many features which have not yet found a place in org-cite, and maybe will never be integrated. It is generally more refined in relation to LaTeX. Unfortunately, Kitchin's attempt to integrate org-ref into org-cite, with a project called org-ref-cite, appears to have stalled. Instead, a new, even better functioning version 3.0 of org-ref has been released.

The org-ref syntax for citing is the same as using org-cite, except it uses the & character in the citation key instead of @.

Thus, if you are prepared to lose compatibility with any backends other than LaTeX, org-ref is the most effective choice for typesetting ORG-mode references and bibliographies. On the other hand, org-cite will improve over time, despite having structural limitations that Kitchin considers insurmountable.[13] Looking forward, adopting org-cite could create more maintainable documents over time.

However, org-cite refers to a specific directive #+BIBLIOGRAPHY: which indicates the bibliographic file to consider, whereas org-ref solves the problem by simply adding the suitable lines at the beginning of the LaTeX file, as in

```
#+LATEX_HEADER: \usepackage[
#+LATEX_HEADER:     citestyle=authoryear-icomp,
#+LATEX_HEADER:     bibstyle=authoryear,
#+LATEX_HEADER:     hyperref=true,backref=true,
#+LATEX_HEADER:     maxcitenames=3,url=true,
#+LATEX_HEADER:     backend=biber,natbib=true]
#+LATEX_HEADER:   {biblatex}
#+LATEX_HEADER: \addbibresource{test.bib}
```

and a \printbibliography at the end.

[13] lists.gnu.org/r/emacs-orgmode/2022-03/msg00250

14 Conclusions: why to use or not use ORG-mode

At the end of this introductory presentation to ORG-mode, we can balance the reasons for using or not using ORG markup instead of LaTeX. Using ORG-mode is beneficial for the following reasons:

1. The simpler and cleaner text format of ORG helps the readability of the source, integration in a less-demanding publishing workflow, and greater ease of archiving [17];
2. The ability to create documents that can be exported in different formats allows simultaneous publication on different media as well as simpler collaboration with co-authors who use different formats;
3. The presence of active source blocks, which can programmatically produce results, enables a research approach oriented towards *Open Science*, and the production of reproducible scientific results [11, 18, 21]. The ORG-mode solution is considerably simpler than those working directly with LaTeX [2].

On the other hand, there are also some clear shortcomings:

1. No journal or publisher accepts native .org format, so delivery of an article will necessarily have to be done in a different format (LaTeX, OpenOffice, etc.).
2. An author already competent with LaTeX, possibly already with his own chain of development tools tested, could consider the effort to obtain some modest advantage superfluous, or even counterproductive, from his point of view. (However, an author who is new to LaTeX may find it advantageous to directly learn ORG markup and how to use Emacs at the same time, thus obtaining in one fell swoop a research and publishing environment, moreover of excellent quality.)
3. ORG-mode requires that you use Emacs as your editor, which you may not necessarily like.

One goal of this article was to demonstrate in practice that *if you like* to use Emacs and ORG-mode to deliver an article in LaTeX, it is *always* possible to follow the guidelines of a magazine or a publisher. This is true for LaTeX, which is the subject of this article, but also true if it is to be delivered in a word-processor format (docx, odt, etc.).

ORG-mode represents an additional, supplementary approach, that you may add to your researcher's toolbox, and not a replacement for LaTeX.

Of one thing we can be sure: we can bet on the longevity of ORG-mode, based as it is on LaTeX and Emacs.

A Using ORG-mode with ArsTEXnica

To submit this article written in ORG-mode to ArsTEXnica,[14] it was necessary to comply with the editorial requirements; in particular, to export using the `arstexnica` class and respect the guidelines given in the kit [8].

It's one of those things that turned out to be easier to do than to document.

A.1 `arstexnica` class

The first thing to do was define `arstexnica` as the class used in the export, by indicating the directive `#+LATEX_CLASS:` in the `.org` file:

```
#+LATEX_CLASS: arstexnica
```

However, this is not enough. We need to configure `org-latex-classes` to add the `arstexnica` class; recall the triad:

```
(class-name start-of-preamble segmentation-elements)
```

The Lisp command for `~/.emacs` is therefore:

```
(with-eval-after-load 'ox-latex
 (add-to-list 'org-latex-classes
  '("arstexnica" "\\documentclass{arstexnica}"
    ("\\section{%s}" . "\\section{%s}")
    ("\\subsection{%s}" . "\\subsection{%s}")
    ("\\subsubsection{%s}" . "\\subsubsection{%s}")
    ("\\paragraph{%s}" . "\\paragraph*{%s}")
    ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))))
```

Already at this point, an export from `.org` allows a good view of the article. However, the produced file does not exactly reflect the example `name.tex` present in the kit. The preamble of the article generated by ORG-mode is different from that of ArsTEXnica. To make it the same, we need to configure some Emacs variables.

The main model of an ArsTEXnica article, contained in the `name.tex` file of the kit, has a fundamental difference from the ORG-mode model: it supports compilation with different typesetting engines for LATEX, namely pdfLATEX, XƎLATEX, or LuaLATEX.

The code that specifies this behavior is this:

```
1   \ifbool{PDFTeX}{%
2       \usepackage[T1]{fontenc}
3       \usepackage[utf8]{inputenc}
4       \usepackage[english, italian]{babel}
5     }{\ifbool{XeTeX}{%
6       \usepackage{polyglossia}
7       \setmainlanguage[babelshorthands]{italian}
8       \PolyglossiaSetup{italian}{indentfirst=false}
9       \setotherlanguage{english}
10    }{\ifbool{LuaTeX}{%
11      \usepackage{polyglossia}
12      \setmainlanguage[babelshorthands]{italian}
13      \PolyglossiaSetup{italian}{indentfirst=false}
14      \setotherlanguage{english}
15      }{%
16      \ArsTeXnicaError\endinput
17      }
18    }
19  }
20
21  \usepackage{cochineal}
22  \ifbool{PDFTeX}{%
23      \usepackage[varqu,varl,var0]{inconsolata}
24    }{%
25      \fontspec[StylisticSet={1,2,3},
26                Scale=MatchLowercase]{inconsolata}
27  }
28 \usepackage[scale=.9,type1]{cabin}
29 \usepackage[cochineal,vvarbb]{newtxmath}
30 \usepackage[cal=boondoxo]{mathalfa}
31
32 \usepackage{microtype}
33 \usepackage{natbib}
34 \usepackage{graphicx}
35 \usepackage{hyperref}
```

Listing 1: The preamble of the template file of an article for ArsTEXnica, present in the kit for authors

In the two blocks on lines 1–19 and 21–27 three cases are given:

1. If the engine is pdfLATEX, the packages used are `fontenc`, `inputenc`, and `babel` for language management, as well as, specified in the second part (line 25), the fixed-pitch font `inconsolata`.
2. If the engine is XƎTEX or LuaTEX, `polyglossia` is used for language management. (The `fontenc` and `inputenc` packages are not needed.) Furthermore, the font `inconsolata` is loaded and its characteristics specified with the `\fontspec` command (lines 25–26).
3. If the engine is none of these three, an error is given with `\ArsTeXnicaError`, and the typesetting stops.

In the next part of the preamble (lines 28–35), other packages are loaded, of which only `graphicx` and `hyperref` are already present in the preamble generated by ORG-mode.

The code produced by ORG-mode does not handle this switch between typesetting engines. Two choices can be made:

1. Assuming that the journal only uses pdfLATEX for typesetting, you may leave the ORG-mode configurations as they are and insert the missing packages (`cochineal`, `inconsolata`, `cabin`, `newtxmath`, `mathalfa`, `microtype`, and `natbib`)

[14] *Editor's note:* The original Italian article was prepared in ORG, while this translated version for *TUGboat* was not. We felt this real-world adaptation to a journal's requirement was still valuable to present.

with `#+LATEX_HEADER:` directives at the beginning of the `.org` file. It's the simpler choice.

2. We may try to completely replicate the file `name.tex`.

We will choose the second option. To achieve the desired result, we will add the entire preamble of the main ArsTEXnica file to the global configuration of the translation of the `arstexnica` class; disabling the default and additional packages in ORG-mode, but not the `[EXTRA]` ones.

In addition to this, both the matter of the double abstract, Italian/English, and the definition of the bibliographic style relating to `natbib` must be managed directly within the `.org` file.

Finally, some details will need to be resolved, such as the correct typesetting of the ArsTEXnica name and the employment of some useful macros.

So, the redefinition of the `org-latex-classes` variable becomes a bit more complex, adding a long string constant with the ArsTEXnica preamble (reformatted slightly for *TUGboat*):

```
1      (with-eval-after-load 'ox-latex
2         (add-to-list 'org-latex-classes
3                      '("arstexnica"
4                        "\\documentclass{arstexnica}
5 [NO-DEFAULT-PACKAGES]
6 [NO-PACKAGES]
7 % The following code allows the use of any type-
8 % setting engine among pdfLaTeX, XeLaTeX, or LuaLaTeX.
9 % Please, don't change the preamble unless it is
10 % strictly necessary. You can add other languages or
11 % fonts if it is required by the subject of the paper.
12 % Other customizations should be added to the files
13 % '\\jobname-package.tex' and '\\jobname-command.tex'.
14 %
15 \\ifbool{PDFTeX}{%
16     \\usepackage[T1]{fontenc}
17     \\usepackage[utf8]{inputenc}
18     \\usepackage[english, italian]{babel}
19   }{\\ifbool{XeTeX}{%
20     \\usepackage{polyglossia}
21     \\setmainlanguage[babelshorthands]{italian}
22     \\PolyglossiaSetup{italian}{indentfirst=false}
23     \\setotherlanguage{english}
24     }{\\ifbool{LuaTeX}{%
25       \\usepackage{polyglossia}
26       \\setmainlanguage[babelshorthands]{italian}
27       \\PolyglossiaSetup{italian}{indentfirst=false}
28       \\setotherlanguage{english}
29       }{%
30       \\ArsTeXnicaError\\endinput
31       }
32     }
33 }
34
35 \\usepackage{cochineal}
36 \\ifbool{PDFTeX}{%
37     \\usepackage[varqu,varl,var0]{inconsolata}
38   }{%
39     \\fontspec[StylisticSet={1,2,3},
40                Scale=MatchLowercase]{inconsolata}
41 }
42 \\usepackage[scale=.9,type1]{cabin}
43 \\usepackage[cochineal,vvarbb]{newtxmath}
44 \\usepackage[cal=boondoxo]{mathalfa}
45 \\usepackage{microtype}
46 \\usepackage{natbib}
47 \\usepackage{graphicx}
48 \\usepackage{hyperref}
49 [EXTRA]
50   "
51     ("\\section{%s}" . "\\section{%s}")
52     ("\\subsection{%s}" . "\\subsection{%s}")
53     ("\\subsubsection{%s}" . "\\subsubsection{%s}")
54     ("\\paragraph{%s}" . "\\paragraph*{%s}")
55     ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))))
```

In practice, the first part of the `name.tex` file has been moved here, while disabling the addition of the default packages via lines 5 and 6. Now, the generation of the `.tex` file by ORG-mode, for the `arstexnica` class, will exactly replicate the preamble of the `name.tex` file, except for what will be explained later.

A.2 Double abstract

ArsTEXnica has a double abstract, presenting the abstract in two languages, Italian and English. Once the first abstract has been defined, two strategies can be used directly inside the `.org` file. The first involves inserting everything necessary for the second abstract in English in a specific `EXPORT` block for LATEX.

```
#+BEGIN_EXPORT latex
  \begin{otherlanguage}{} % Second language
  \begin{abstract}
  In writing academic ...
  \end{abstract}
  \end{otherlanguage}
#+END_EXPORT
```

The main drawback of this strategy is that the text inserted in the abstract will not be processed by ORG-mode but copied as is. Therefore, any markup will not be transformed, for example, emphasis or representation of the LATEX logo, and must be expressed directly in LATEX markup.

The alternative is not to use an `EXPORT` block, but to include the second block of abstracts between two LATEX command lines that open and close the `otherlanguage` environment, like this:

```
#+LATEX: \begin{otherlanguage}{english}
#+BEGIN_ABSTRACT
In writing academic ...
#+END_ABSTRACT
#+LATEX: \end{otherlanguage}
```

In this case, the content of the second `ABSTRACT` block will also benefit from the `ORG` markup transformation.

A.3 Macros and ArsTeXnica, the logo

The correct typesetting of the ArsTeXnica logo is a detail that could not be overlooked. It was also the hardest thing to do.

An easier solution would have involved using the LaTeX command directly in the `.org` source, or using the `#+MACRO:` directive, but this would have meant having to write in the text `'\ArsTeXnica{}` in the first case or `{{{ArsTeXnica()}}}` in the second. Horrible!

For those macros that "are often used when talking about the TeX system" [8, p. 4] like `\pkgname`, `\clsname`, `\optname`, and so on, it can be useful to define ORG-mode macros using `#+MACRO:`, as in:

```
#+MACRO: envname @@latex:\envname{$1}@@
```

and then use them with `{{{envname(verbatim)}}}`.

This approach would make sense if the aim was to obtain a document that could also be typeset in other backends besides LaTeX, in which case the `ORG` macro would have to be changed by inserting alternatives for each backend. For example, to be able to handle HTML as well, we could do something like this (except all on one line):

```
#+MACRO: envname @@latex:\envname{$1}
#@@@@html:<b class="envname">$1</b>@@
```

and then define suitable CSS for the `envname` class, possibly with a common representation mode for all environment names. In this case, the increased complexity in the use of the macro would be justified.

However, this article is only intended for delivery in LaTeX format to ArsTeXnica, so the best choice is to directly use LaTeX macros in the `.org` source. To get an environment name like `verbatim`, we'd like to write `\envname{verbatim}`, just like in LaTeX.

But for the ArsTeXnica logo, a different approach will be taken, even only to demonstrate why the open source approach and Lisp programming make Emacs a superior editor, adaptable to the user's needs down to the smallest detail [6, 13, 15].

It should be noted that the text strings `LaTeX` and `TeX` present in the `.org` source are automatically transformed by the LaTeX backend into their equivalent LaTeX command, i.e. `\LaTeX{}` and `\TeX{}`. We want to get the same result with `ArsTeXnica` as well.

Unfortunately, this operation is not immediate or configurable in any way, because it is performed by a non-customizable Lisp function. We have to modify the programming of the `ox-latex.el` library.

Here the flexibility of the Lisp language comes to our aid. It is an interpreted environment where a function can be replaced simply by redefining it.

With a little research in the source code of the ORG-mode libraries, it turns out that the responsible function is `org-latex-plain-text` as defined in `ox-latex.el`. Its definition is this:

```
(defun org-latex-plain-text (text info)
 "Transcode a TEXT string from Org to LaTeX.
TEXT is the string to transcode.  INFO is a plist
holding contextual information."
 (let* ((specialp (plist-get info
                             :with-special-strings))
        (output
         ;; Turn LaTeX into \LaTeX{}
         ;; and TeX into \TeX{}.
         (let ((case-fold-search nil))
           (replace-regexp-in-string
            "\\<\\(?:La\\)?TeX\\>" "\\\\\\&{}"
            ;; Protect ^, ~, %, #, &, $, _, { and }.
            ;; Also protect \. However, if
            ;; special strings are used, be careful
            ;; not to protect "\" in "\-" constructs.
            (replace-regexp-in-string
             (concat "[%$#&{}_~^]\\|\\\\"
                     (and specialp "\\([^-]\\|$\\)"))
             (lambda (m)
               (cl-case (string-to-char m)
                 (?\\ "$\\\\backslash$\\1")
                 (?~ "\\\\textasciitilde{}")
                 (?^ "\\\\^{}")
                 (t "\\\\\\&")))
             text)))))
   ;; Activate smart quotes.  Be sure to provide
   ;; original TEXT string since OUTPUT may have
   ;; been modified.
   (when (plist-get info :with-smart-quotes)
     (setq output (org-export-activate-smart-quotes
                   output :latex info text)))
   ;; Convert special strings.
   (when specialp
     (setq output (replace-regexp-in-string
                   "\\.\\.\\." "\\\\ldots{}"
                   output)))
   ;; Handle break preservation if required.
   (when (plist-get info :preserve-breaks)
     (setq output (replace-regexp-in-string
                  "\\(?:[ \t]*\\\\\\\\\\)?[ \t]*\n"
                  (concat org-latex-line-break-safe
                          "\n")
                  output nil t)))
   ;; Return value.
   output))
```

This is a translation function that transforms the basic text from `ORG` markup to LaTeX, limited to some elements that cannot be expressed as they are; among other things, it translates `TeX` and `LaTeX`. So this is the right place to transform `ArsTeXnica` as well.

We need to change the regular expression on line 10 of the previous listing where the matching regular expression (`"\\<\\(?:La\\)?TeX\\>"`) captures the string `TeX`, possibly preceded by `La`, and replaces it with the replacement pattern (`"\\\&{}"`), i.e. it prefixes the recognized string with a \, followed by the string captured in the parser, denoted as `\&`, finally followed by the opening and closing braces (i.e., changing `LaTeX` to `\LaTeX{}`). The backslashes of the replacement pattern must, in turn, be protected with a backslash so the definitive pattern, present at the end of line 10 in the listing, will have as many as six backslashes in sequence: `"\\\\\\&{}"`. Besides the many parentheses, these *trains* of backslashes are one of the least appreciated features of the Emacs Lisp language.

To also match `ArsTeXnica`, we need to modify the matching regular expression in this way:

```
"\\<\\(?:La\\|Ars\\)?TeX\\(nica\\)?\\>"
```

At this point, we may as well also add the other acronyms defined in the `arsacro.sty` file included in the ArsTeXnica kit by completing the table:

String ORG-mode	Formatted with string in ORG-mode (e.g. `TeX`)	Formatted with LaTeX macro (e.g. `\TeX`)
`TeX`	TeX	TeX
`LaTeX`	LaTeX	LaTeX
`ArsTeXnica`	ArsTeXnica	ArsTeXnica
`PCTeX`	PC TeX	PC TeX
`pcTeX`	pcTeX	pcTeX
`pdfTeX`	pdfTeX	pdfTeX
`pdfLaTeX`	pdfLaTeX	pdfLaTeX
`PiC`	PiC	PiC
`PiCTeX`	PiCTeX	PiCTeX
`plain`	`plain`	`plain`
`SliTeX`	SliTeX	SliTeX

The final regular expression will then be (except written on one line):

```
"\\<\\(?:La\\|Ars\\|pc\\|PC\\|pdf\\|pdfLa\\|PiC
\\|Sli\\)?TeX\\(nica\\)?\\|plain\\|PiC\\>"
```

The problem is that there is no way to insert this regular expression change into the `org-latex-plain-text` function; we must *completely replace* this function, by redefining it with the new regular expression, and including it for example in the initial Emacs configuration file (`~/.emacs`). Even better will be to define an Emacs *customization variable*, with a meaningful name (let's say `org-latex-plain-subs-regexp`) and insert this variable in the new function, so that it can be subsequently configurable by the user.

```
; the concat is due to TUGboat's column width.
(defcustom org-latex-plain-subs-regexp (concat
  "\\<\\(\\(?:La\\|Ars\\|pc\\|PC\\|pdf\\|pdfLa\\|PiC"
      "\\|Sli\\)?TeX\\(nica\\)?\\|plain\\|PiC\\)\\>")
  "Regular expression that recognizes strings to be
   transformed into LaTeX commands."
  :group 'org-export-latex
  :type '(string :tag "Regular Expression")
  :safe #'stringp)
```

The new function to insert in the `~/.emacs` file is therefore just the same as presented on the previous page, except line 12 now uses our new variable:

```
...
            (replace-regexp-in-string
12           org-latex-plain-subs-regexp
             "\\\\\\&{}"
...
```

Now, in the `.org` source, it will be possible to use `ArsTeXnica` to obtain ArsTeXnica.

The last problem is that there is no LaTeX macro called `\ArsTeXnica` since the kit only defines the command `\Ars`. At this point, as recommended by the guidelines [8, p. 5], you can add a definition (either in the preamble or directly in the `.org` file with a `#+LATEX_HEADER:` directive):

```
#+LATEX_HEADER: \let\ArsTeXnica\Ars
```

A.4 Last details: `hyperref` and the `article` environment

One of the small problems with ORG-mode's standard export algorithm is the automatic creation of the PDF file metadata definition in the `\hypersetup` command, which must not be defined for delivery to ArsTeXnica:

```
\hypersetup{
 pdfauthor={Emmanuele F. Somma},
 pdftitle={Emacs ORG-Mode, LaTeX (and ArsTeXnica)},
 pdfkeywords={Emacs ORG LaTeX typesetting },
 pdfsubject={An article for ArsTeXnica },
 pdfcreator={Emacs 28.2 (Org mode 9.5.5)},
 pdflang={Italian}}
```

The `org-latex-hyperref-template` variable must therefore be set to `nil`. It can be done also directly in the `.org` file using the `#+BIND:` directive:

```
#+BIND: org-latex-hyperref-template nil
```

Finally, all the article text must be enclosed by an `article` environment, which is specific to ArsTeXnica. To handle this, we can use two `#+LATEX:` directives, one at the beginning and one at the end of the text.

```
#+LATEX: \begin{article}
...
#+LATEX: \end{article}
```

A.5 And finally... the delivery

After all the revisions and checks, when the moment of delivery arrives, one wonders: "Now, what to deliver?"

Currently, ArsTeXnica does not accept `.org` sources: we therefore need to generate the LaTeX source with `C-c C-e l l` and include it together with all necessary additional elements, such as the images, in a zip package and finally deliver it to the editorial and peer-review process.

Done! Happy peer review... and happy reading to all.[15]

References

[1] A. Babenhauserheide. Tutorial: Writing papers for ACPD using Emacs Org-mode. `www.draketo.de/files/howto-write-for-acpd-with-emacs.pdf`, 2014.

[2] H. Bar, H. Wang. Reproducible Science with LaTeX. *Journal of Data Science*, 19(1):111–125, 2021. `arxiv.org/abs/2010.01482`, `doi.org/10.6339/21-JDS998`

[3] M. Borkowski. TeXing in Emacs. *TUGboat* 39(1), 2018. `tug.org/TUGboat/tb39-1/tb121borkowski-emacs.pdf`

[4] M. Fruchart, B. Guinhouya, et al. Jupyter Notebooks for introducing data science to novice users. *Studies in Health Technology and Informatics*, 294:823–824, 2022. `doi.org/10.3233/shti220598`

[5] M. Giordano, O. Iovino, M. Leccardi. *Guida pratica all'uso di GNU Emacs e AUCTEX*. GuIT, 2013. `github.com/GuITeX/guidaemacsauctex`

[6] B. Glickstein. *Writing GNU Emacs Extensions: Editor Customizations and Creations with Lisp*. O'Reilly Media, Inc., 1997.

[7] J. Gruber. Markdown: Syntax, 2004. `daringfireball.net/projects/markdown/syntax`

[8] GuIT. Istruzioni per gli autori, 2022. `guitex.org/home/en/for-authors`

[9] D.E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984. `doi.org/10.1093/comjnl/27.2.97`

[10] D.E. Knuth, S. Levy. *The CWEB System of Structured Documentation*. Addison-Wesley Professional, 1993. `www-cs-faculty.stanford.edu/~knuth/cweb.html`

[11] A. Leha, T. Beißbarth. The Emacs Org-mode: Reproducible research and beyond. In *The R User Conference, useR! 2011 August 16-18 2011 University of Warwick, Coventry, UK*, p. 28, 2011. `www.r-project.org/conferences/useR-2011/abstracts/010411-lehaandreas.pdf`

[12] S. Leonard. Guidance on Markdown: Design philosophies, stability strategies, and select registrations. RFC 7764, IETF, Mar 2016. `www.rfc-editor.org/rfc/rfc7764.txt`

[13] B. Lewis, D. LaLiberte, et al. *GNU Emacs Lisp Reference Manual*. Free Software Foundation, 1997. `gnu.org/s/emacs/manual/html_node/elisp`

[14] T. Mailund. *Introducing Markdown and Pandoc: Using Markup Language and Document Converter*. Apress, 2014. `doi.org/10.1007/978-1-4842-5149-2`

[15] S. Monnier, M. Sperber. Evolution of Emacs Lisp. *Proceedings of the ACM on Programming Languages*, 4(HOPL):1–55, 2020. `doi.org/10.1145/3386324`

[16] Org-mode contributors. Org mode tools, 2022. `orgmode.org/worg/org-tools/`

[17] C. Schöch. The right tool for the job: Five collaborative writing tools for academics, 4 April 2014. Impact of Social Sciences Blog. `blogs.lse.ac.uk/impactofsocialsciences/2014/04/04/five-collaborative-writing-tools-for-academics/`

[18] E. Schulte, D. Davison. Active documents with Org-mode. *Computing in Science & Engineering*, 13(3):66–73, 2011. `doi.org/10.1109/MCSE.2011.41`

[19] E. Somma. Il respawn di Infomedia (LaTeX-based). *ArsTeXnica*, (8):92–101, Ottobre 2009. `guitex.org/home/numero-8`

[20] R.M. Stallman. Emacs: The extensible, customizable self-documenting display editor. In *SIGPLAN SIGOA Symposium on Text Manipulation*, 1981. `doi.org/10.1145/872730.806466`

[21] L. Stanisic, A. Legrand, V. Danjean. An effective Git and Org-mode based workflow for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):61–70, 2015.

[22] J. Voegler, J. Bornschein, G. Weber. Markdown – A simple syntax for transcription of accessible study materials. In *Computers Helping People with Special Needs: 14th International Conference, ICCHP 2014, Paris, France, July 9-11, 2014, Proceedings, Part I*, pp. 545–548. Springer, 2014. `doi.org/10.1007/978-3-319-08596-8_85`

[23] J.J. White. Using markup languages for accessible scientific, technical, and scholarly document creation. *Journal of Science Education for Students with Disabilities*, 25(1):22 pp., 2022. Article 5. `scholarworks.rit.edu/jsesd/vol25/iss1/5/`

⋄ Emmanuele F. Somma
Bank of Italy
Economics, Statistics and Research
Structural Economic Analysis
`ef dot somma (at) exedre dot org`

[15] The `.org` code of this article is available on the repository `gitlab.com/exedre/arstexnica-orgmode`.

Plain TeX's `\ooalign`

Udo Wermuth

Abstract

This article looks at a single macro defined in the file `plain.tex`: `\ooalign`. Its definition is analyzed and several examples for its use are presented.

1 Introduction

On page 9 of *The TeXbook* [1] the author, Donald E. Knuth, makes a promise: "TeX understands about 900 control sequences as part of its built-in vocabulary, and all of them are explained in this manual somewhere." And on the next page the number 900 is split into two parts: "In the following chapters we shall frequently discuss 'plain TeX' format, which is a set of about 600 basic control sequences that are defined in Appendix B. These control sequences, together with the 300 or so primitives, are usually present when TeX begins to process a manuscript; that is why TeX claims to know roughly 900 control sequences when it starts." So the `plain` format is covered in the book at least by Appendix B.

It's necessary to write "at least" in the previous sentence since we read in Appendix B on page 343, "When we come to macros whose usage has not yet been explained — for example, somehow `\vglue` and `\beginsection` never made it into Chapters 1 through 27 — we shall consider them from a user's viewpoint; but most of the time we shall be addressing the issues from the standpoint of a macro designer." (Note however, in Chapter 23, page 259f., a different, more complex `\beginsection` from a real-world book design is discussed.)

Another macro of the `plain` format that never made it into the numbered chapters of the book is `\ooalign`. It is defined in Appendix B on page 356. Well, there is an important difference between the two macros `\ooalign` and `\beginsection`. The first is definitely something for macro writers, the second serves users who want to enter text. But the latter is only used in a test file of Appendix B, not in *The TeXbook* itself. (As mentioned above, a more complex macro is shown in Chapter 23 and later in Appendix E, page 418f., Knuth explains the non-`plain` macro `\beginchapter` that is used in the creation of *The TeXbook*.) On the other hand the macro `\ooalign` is used several times in the `plain` format.

Nevertheless it seems that the macro `\ooalign` is a little bit hidden through this treatment. Like the simple `\beginsection`, other authors of books that concentrate on TeX's primitives and the `plain` format hardly mention the macro `\ooalign`. I surmise `\ooalign` is seen by most people as a support macro that Knuth uses to define an accent (`\c`: ¸) and the copyright symbol (`\copyright`: ©) although it appears in macros for math modes too.

But I assume that most "support macros" in the `plain` format are written with the '`@`' symbol in their names to make them private to the format. As `\ooalign` doesn't contain the '`@`' in its name, it might indicate that Knuth wants people other than TeX experts to consider `\ooalign` a useful tool.

Contents. Section 2 shows and explains the definition of the macro `\ooalign`. Then we look at simple examples of its use in section 3. The constructed symbols are meant to be entered in normal text. Symbols that should be used in a math mode need more complex code as section 4 shows. Section 5 increases the complexity and handles a weird use case.

2 `\ooalign`'s definition

As the macro name implies the macro has something to do with alignment. Here is the definition as it appears in `plain.tex`:

```
\def\ooalign{\lineskiplimit-\maxdimen \oalign}
```

This definition is followed by a comment: "chars over each other". Okay, one of the two 'o's vanishes in the replacement text to leave us with another unknown macro of the `plain` format: `\oalign`. As we will see in a moment, `\oalign` calls another unknown macro, `\ialign`, so I assume it's best to study the whole chain of macro calls in one code block.

Note that the macro `\ooalign` has no parameters. Of course, we expect that "characters" follow so that the comment makes sense. The truth is that it's `\oalign` that has a parameter but for the users of `\ooalign` it looks as if this macro takes an argument. (Some people call this a *pseudo-parameter* for `\ooalign`.) This technique is used if certain parameter settings must be done before the argument is digested by TeX. Here the `\lineskiplimit` is set.

This assignment is a bit of a surprise in the above definition. The dimension `\lineskiplimit` occurs usually in a triumvirate with the two glue parameters `\baselineskip` and `\lineskip`. Together they control the *interline glue* design, i.e., the vertical line spacing; see [1], p. 78f.

Interline glue. When TeX typesets text in paragraphs it observes that the distances between adjacent lines have a certain value stored in the glue parameter `\baselineskip`. To get this distance TeX looks for every pair of lines at the depth of the first

doi.org/10.47397/tb/44-3/tb138wermuth-ooalign

line and the height of the second. Both values are subtracted from the \baselineskip to get the value for the interline glue. TEX adds this glue value between the two lines to keep the baselines exactly \baselineskip apart. (Usually this glue parameter has only a natural width and no stretch or shrink component. Nevertheless, TEX allows this kind of flexibility here.)

But there is an exception. If the natural width of the computed interline glue is smaller than the dimension \lineskiplimit, TEX ignores the computed value and inserts glue with the value of the parameter \lineskip between the lines.

The format plain sets \baselineskip to 12 pt, \lineskip to 1 pt, and \lineskiplimit to 0 pt. So, in plain TEX lines are kept 1 pt apart if the natural width of the computed interline glue becomes negative, i.e., if the lines might overprint each other. ("Might" because the places with large depth in the first line and large height in the second might not be at the same position inside the lines.)

The macro chain. Let's return to the definition of \ooalign and describe it in detail with comments.

```
\def\ooalign{% ‘‘chars over each other’’
 \lineskiplimit=-\maxdimen %-\maxdimen: smallest
 \oalign}%                          dimension in TeX
```

From the discussion about interline glue we expect that the macro \oalign sets the two missing parameters. The following code from plain.tex — extended by my comments — meets our expectation.

```
\def\oalign#1{% #1: cell entries for \ialign
 \leavevmode   % in horizontal mode: do nothing;
           % otherwise switch to horizontal mode
 \vtop{%   put the alignment into an unbreakable
                % box with height of the 1st row
    \baselineskip=0pt % switch off \baselineskip
         % and set a default value to \lineskip:
    \lineskip=.25ex %  a quarter of the x-height
    \ialign{##\crcr   % preamble of a one-column
        #1\crcr}}}%        initialized alignment
```

Before we discuss the settings of the three parameters for the interline glue, let's look at the macro that is called at the end of \oalign.

```
\def\ialign{% initialized horizontal alignment
 \everycr={}\tabskip=0pt \halign}
```

The primitive \halign reacts to two TEX parameters with their current settings: First, the token list \everycr is applied directly after the preamble and at every \cr (or non-redundant \crcr); see [1], p. 275. Second, TEX applies the current value of the \tabskip glue from outside of an \halign to the left of the alignment; see [1], p. 238. So it makes sense to first reset both values to control the appearance of the horizontal alignment.

Now we can also describe the pseudo-parameter of \ooalign: It consists of a sequence of rows for a horizontal alignment. Each row entry except the last must end with \cr (or \crcr).

Why \lineskip? If you paid attention to the discussion about interline glue then you might wonder why \oalign sets \lineskip to a nonzero value. The macro \ooalign assigns the smallest acceptable dimension of TEX to \lineskiplimit, so no computation can give a smaller value: TEX never discards its computed value for the interline glue and thus it never applies the value of \lineskip.

Of course, there is a reason that \lineskip is set by \oalign. The macro \ooalign has a sibling \o@lign that's private to the plain format.

```
\def\o@lign{\lineskiplimit=0pt \oalign}
```

Now we can see why \lineskiplimit is set outside of \oalign. In \o@lign, \lineskip is always applied if the interline glue is negative; and this is always the case, as \baselineskip is 0 pt. Thus, every nonzero depth and height of the pair of lines makes TEX insert glue with a natural height of 0.25 ex.

The macro \o@lign is private to plain as it has fewer use cases than \ooalign. The macro only occurs in the definition of two accents. As with the above-mentioned cedilla accent, where \ooalign is used, these accents are set below the baseline.

So, what does \ooalign do? It takes material provided as cell entries for an alignment and TEX typesets these entries as one column with baselines that are 0 pt apart. Or in other words: TEX prints the second row on top of the first, then the third on top of the combination of the first and second, etc. As the entries are characters, we get "chars over each other". The material in the first row determines the height of the result, that in the last row its depth.

But recall that TEX determines in a horizontal alignment the widest cell entry in all the rows of a column and uses this width for the other rows in this column too. Users of \ooalign must make sure that TEX finds enough glue in the cell entries to accomplish this.

Next, \ooalign sets a lot of parameters, sometimes to unusual values. For example, we certainly want to return to the current interline glue parameters as soon as \ooalign finishes its job. Thus, users of the macro should call it only inside a group.

3 Simple examples

The Introduction mentions that \ooalign is used to create the copyright symbol. So, for our first example, let's see how to generate this symbol. It's based on two characters of Computer Modern Roman. First, a circle is needed, called `\Orb`: ◯. Second, we need the letter ‘c’ from the default font `cmr10`. (I present the code in my style.)

```
\def\copyright{% typeset a ‘c’ inside a circle
 {\ooalign{%      call \ooalign inside a group
   \hfil  % start 1st row; center its contents
     \raise0.07ex\hbox{c}% output ‘c’ (raised)
     \hfil\crcr               % end the first row
   \Orb\crcr}}}%   the 2nd row with the circle
```

The `\crcr` after `\Orb` is not required as the code of `\oalign` shows. But it doesn’t hurt to add it and to make the code for the symbol easier to understand.

The value 0.07 ex that occurs in the code of the macro might have been found by trial and error or by insight into the definitions of the involved characters. So either you have to experiment to find the correct value or look up the definitions of the involved characters in [2] and do some computations.

We could have coded the symbol a little bit different but with the same output:

```
\def\Copyright{% typeset a ‘c’ inside a circle
 {\ooalign{%      call \ooalign inside a group
   \Orb\crcr     % the 1st row with the circle
   \hfil  % start 2nd row; center its contents
     \raise0.07ex\hbox{c}% output ‘c’ (raised)
     \hfil\crcr}}}%       % end the second row
```

Here is a direct comparison: ⓒ ⓒ entered as `\copyright` and `\Copyright`. But remember what was stated in section 2: The symbols aren’t identical. Let’s add a superscript and a subscript to both symbols: a) ⓒ* ⓒ* b) ⓒ$_+$ ⓒ$_+$. Thus the sequence of the rows is important if the created symbol is later used in certain situations.

Many symbols are based on a circle and another character. Let’s create some textile care labels as defined by the organization GINETEX. We want to write a macro that produces the symbol Ⓟ (dry clean with tetrachloroethylene (PCE) only): A sans-serif font is combined with the circle called `\Orb`.

```
\font\DCfont=cmss8 % sans-serif font for the ‘P’
\def\PCE{{%             open a group for \ooalign
 \ooalign{%  horizontal alignment with two rows
  \hidewidth\hbox{\DCfont\kern0.06em P}%
     \hidewidth\cr % 1st row, centered contents
  \raise0.1ex\hbox{\Orb}\cr % 2nd row: a circle
}}}%     end \ooalign, the group, and the macro
```

As mentioned above, the symbol is created using row entries of an alignment. Hence the usual techniques known from horizontal alignments can be used. Here we suppress the width of the letter with `\hidewidth` and apply `\cr` instead of `\crcr`.

And we can apply other aspects from horizontal alignments too. The above symbol has a variant if the dry cleaning must be done in a more “gentle” way. This is indicated by a horizontal rule below the circle: $\underline{\text{Ⓟ}}$. We can realize this rule with a `\noalign`.

```
\def\gentlePCE{{% open a group for \ooalign
 \ooalign{%      the rows are known from \PCE
    \hidewidth\hbox{\DCfont\kern0.06em P}%
       \hidewidth\cr
    \raise0.1ex\hbox{\Orb}\cr
    \noalign{\vskip1pt \hrule}}}}% add a rule
```

Using symbols from math fonts. Of course the cell entries can also contain inline math. But there is a new aspect to consider if the defined macros should work in all situations.

In [6], p. 444, the following macro is shown to print the symbol for acid-free paper.

```
\def\goodpaper{{%
  \ooalign{\hfil
    \raise.25ex\hbox{%
      $\scriptstyle\mathchar"231$}%
    \hfil\crcr
    \mathhexbox20D}}}
```

As `\mathchar"231` stands for the symbol ‘∞’ and `\mathhexbox20D` for the `\Orb`, i.e., ‘◯’, the result is: ∞⃝. As stated in the article, the macro works well at the site of the author.

But the macro isn’t as universal as it could be; it makes an assumption that a certain parameter has more or less its default value. TeX adds space around inline math as specified by the value of the dimension `\mathsurround`; [1], p. 162. The default value in `plain` is 0 pt. But at a site that changes this default value, say, setting `\mathsurround=2pt`, the above definition gives a distorted symbol: ∞⃝ .

The macros that define `\ooalign` assign special values to a lot of parameters and we use therefore a group when `\ooalign` is called. Thus, it’s easy to reset inside this group one more parameter to avoid the described problem. Let’s set `\mathsurround` to 0 pt if inline math is used in the `\ooalign`. (Note, `\scriptspace` should be set to 0 pt too if you use super- or subscripts in the math.)

All the above examples are made from two symbols but that isn’t a fixed limit. For example, to express that dry cleaning with chemicals must be avoided, the symbol ‘⊗’ was designed. This time three characters are involved as the cross symbol consists of the less-than and greater-than symbols.

```
\def\NOTchemical{{\mathsurround=0pt % init math
 \ooalign{\hidewidth % 1st row with two symbols
   \raise0.1ex\hbox{$>\mkern-3mu<$}%
   \hidewidth\cr          % 1st row is centered
  \raise0.1ex\hbox{%     same \raise as in \PCE
   \kern2pt\Orb\kern2pt}\cr}}}% 2nd row: circle
```

Note that inside the math mode I work with `\mkern` instead of `\kern`.

Up to now all examples were constructed with two row entries. But of course, this isn't a limit either. Any number of rows can be used. Let's look at an example with inline math and three rows.

```
\def\frowny{{% a group is required for \ooalign
 \mathsurround=0pt %  required with inline math
 \ooalign{\hidewidth \raise.3ex\hbox{% the eyes
   $\cdot\mkern 2mu\cdot$}\hidewidth\cr
  \hidewidth \lower.2ex\hbox{%       the mouth
   $\scriptscriptstyle \frown$}\hidewidth\cr
  \Orb\cr}}}%                        the head
```

The above code typesets this symbol: ☹. It shouldn't be difficult to create the symbol '☺' if you prefer this one.

4 Complex examples

The symbols that we defined in the previous section are designed to be entered in horizontal mode, i.e., inside a paragraph. But `\ooalign` can also be used to create symbols that occur in math mode. We need to be more careful, as such a symbol must be allowed to appear in a displayed equation, in inline math inside a paragraph, or as a super- or subscript to another symbol. In other words we must code the symbol in a way that TeX scales its size for the different use cases.

The Computer Modern math fonts provide the symbol '$\in$' through the control word `\in`. It represents a relation between two entities like the equal sign. Mathematicians write this relation to express the phrase "is a member of". For example, it is used to express that a number is a member of a set of numbers: $8 \in \{0,2,4,6,8\}$. To state that a number is not in this set the symbol '$\in$' is crossed out: $7 \notin \{0,2,4,6,8\}$.

No Computer Modern math font contains this crossed-out symbol. Plain TeX names it "`\notin`"; a macro builds it from a slash, i.e., '/', and the symbol '$\in$'. We look at this macro in a moment but first let's see how we would define this macro. I name it `\NOTin` as I don't want to replace the macro of `plain.tex`.

```
\def\crossedoutin{% print ‘/’ over \in in math
 {\mathsurround=0pt %  a group, no \mathsurround
  \ooalign{$\hfil\mkern 1mu/\hfil$\crcr% the ‘/’
     $\in$\crcr}}}%    close group and end macro
\let\NOTin=\crossedoutin
```

The indirect approach to define `\NOTin` will become clear in a moment.

First, let's perform a test of the new symbol. In the following I entered `\NOTin` every time TeX typesets $\notin$: "$8 \in \{0,2,4,6,8\}$ and $7{\notin}\{0,2,4,6,8\}$ and (a little bit silly) as a subscript

$$X_{8\in\{0,2,4,6,8\}} \quad \text{and} \quad X_{7\notin\{0,2,4,6,8\}}"$$

It is obvious that (a) the spacing for the new symbol and (b) the size in the subscript aren't good. Look closely and you see that the axis of the new symbol in the subscript isn't perfect.

TeX doesn't know that the new symbol is a relation. We must inform TeX about this fact. The primitive `\mathrel` assigns the new symbol to the class that represents relations ([1], p. 155). This fixes problem (a):

```
\def\NOTin{\mathrel\crossedoutin}
```

"$8 \in \{0,2,4,6,8\}$ and $7 \notin \{0,2,4,6,8\}$ and (a little bit silly) as a subscript

$$X_{8\in\{0,2,4,6,8\}} \quad \text{and} \quad X_{7\notin\{0,2,4,6,8\}}"$$

Mathematics: classes and styles. TeX assigns to every math character one of eight *classes*. "Relation" is one of these classes. The others are: ordinary, large operator, binary operation, opening, closing, punctuation, and variable family; see [1], p. 154. Any symbol can be assigned to one of these classes (except the last): Precede the symbol by one of these primitives `\mathrel`, `\mathord`, `\mathop`, `\mathbin`, `\mathopen`, `\mathclose`, or `\mathpunct`, respectively. The class of an object determines how TeX treats it in certain situations. For example, TeX looks at the class to determine the space around the object.

In math mode TeX knows different styles. We all know that there is a difference between inline math in a paragraph — started and ended with a single `$` — and the display math mode that's started and ended with two dollar signs. TeX knows four styles when it typesets mathematics ([1], p. 141): (a) *display style* for material in display math mode, (b) *text style* for inline math, (c) *script style* for super- and subscripts in display or text style, and (d) *scriptscript style* for super- and subscripts of super- or subscripts. These four styles have also a *cramped* version, so there are eight styles in all.

TeX decides when to use the cramped versions so we don't need to be worried about them. But the original styles must be addressed if we want to create a symbol that TeX can use in all situations. TeX provides us with primitives that select one of the four styles: `\displaystyle`, `\textstyle`, `\scriptstyle`, and `\scriptscriptstyle`. The last two were already used in the code for `\goodpaper` and `\frowny`.

We don't have to use these primitives directly to code four different symbols. The `plain` format helps us as it provides the macro `\mathpalette` ([1], p. 151). It offers TeX the symbol in the four styles.

The macro receives two parameters. The first is the name of the symbol that should be processed. `\mathpalette` requires that the macro for the symbol accepts at least one argument. This argument is supplied by `\mathpalette` and is the primitive that switches to one of the four styles available in math modes. For example, the argument might be `\displaystyle`.

The second argument of `\mathpalette` adds some flexibility to the macro of the new symbol: The argument is passed on to the macro of the symbol, but has no special meaning for `\mathpalette`.

Combine `\mathpalette` and `\ooalign`. To apply `\mathpalette` in `\NOTin` we must change the macro `\crossedoutin` as it now gets an argument. In our example there is no need for an additional argument to `\crossedoutin`, so we will use `{}` in the call of `\NOTin`. But, please, don't be confused: This empty group isn't the first argument to `\crossedoutin`, it's an argument of `\mathpalette`. In a second step, `\mathpalette` takes this empty group and offers it to the macro found as its first argument if this macro asks for two arguments.

```
\def\crossedoutin#1{% #1: a math style
 {\mathsurround=0pt %  a group, no \mathsurround
  \ooalign{% build two rows and use the style #1
     $#1\hfil\mkern 1mu/\hfil$\crcr    % the ‘/’
     $#1\in$\crcr}}}%  close group and end macro
\def\NOTin{% applicable in all math styles
 \mathrel{\mathpalette\crossedoutin{}}}
```

This code fixes problem (b). Here is our test input: "$8 \in \{0,2,4,6,8\}$ and $7 \notin \{0,2,4,6,8\}$ and (a little bit silly) as a subscript

$$X_{8\in\{0,2,4,6,8\}} \quad \text{and} \quad X_{7\notin\{0,2,4,6,8\}}\text{"}$$

We no longer need a group for `\ooalign` (and `\mathsurround`) as the construction is later placed in a group for `\mathrel`. But it doesn't hurt to have this group.

It's time now to look at `plain.tex` again and see how Knuth implemented the macro `\notin`. The macro `\m@th` sets `\mathsurround` to 0 pt.

```
\def\m@th{\mathsurround\z@}
\def\c@ncel#1#2{\m@th
 \ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
\def\notin{\mathrel{\mathpalette\c@ncel\in}}
```

We find a more general macro, `\c@ncel`, that prints the slash over its second argument. Thus it can be used in more constructions than just for the symbol `\in`. Which symbol gets canceled is the second argument of `\mathpalette`. Otherwise the construction is very similar to our solution.

Create two new binary operators. As the last example in this section, let's consider creating two new binary operators: ⧆ and ⧆, named with control words `\bast` and `\bstar`, respectively. As we need more than one symbol, we apply the technique that was used in the macro `\notin` with a second argument for `\mathpalette`.

The symbols `\ast` ($\ast$) and `\star` ($\star$) can be used for the inner symbol. For the square, let's check the font tables in Appendix F of [1]. But none is there. Either we now check font tables of other fonts, for example, the AMS symbol font, or we can construct a square: Use a combination of `\sqcap` and `\sqcup`.

```
\def\bbox#1#2{% #1: a math style;
% #2: a symbol to be placed inside a square
 \mathsurround=0pt % initialize for inline math
 \ooalign{% with three rows; two for the square
     $#1\sqcap$\cr  % three sides of the square
     $#1\sqcup$\cr        % and its fourth side
   \noalign{\vskip-0.109ex}
     % center the symbol and cancel its width
     $#1\hidewidth#2\hidewidth$\cr}}
\def\bast{\mathbin{\mathpalette\bbox\ast}}
\def\bstar{\mathbin{\mathpalette\bbox\star}}
```

Now we can type $x \boxast y$ and $X \boxast Y$ as well as $Z_{x\boxast y}$. More symbols of this kind are possible; for example, `\bdot` and `\bcirc`: $x \boxdot y \neq x \circledcirc y$.

```
\def\bdot{\mathbin{\mathpalette\bbox\cdot}}
\def\bcirc{\mathbin{\mathpalette\bbox\circ}}
```

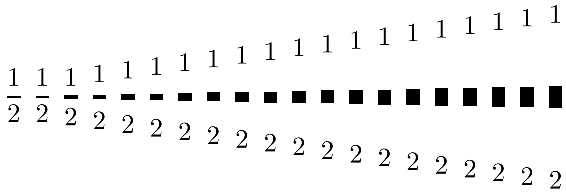
Note, in the definition of `\bbox` the extra group for `\mathsurround` is omitted as we have a group for `\mathbin`. Moreover, `\cr` is used instead of `\crcr`; it doesn't make a difference.

5 An experiment

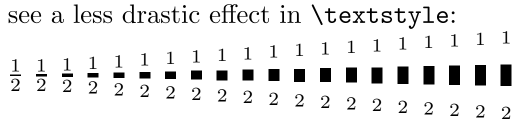
I presume that every reader will agree that TeX is a complex system. And in Knuth's words ([3], p. 661): "Any complex system can be improved; therefore the goal of absolute perfection and optimality is unattainable." Along with the errors that he has made and fixed during the lifetime of TeX, he also lists some aspects of TeX that can be improved; some of them he calls "design flaws" in [3], p. 660.

Here is the text of the first design flaw from [3]: "Additional parameters in symbol fonts could govern the minimum distance between ruled lines in fractions, `\sqrt`, `\overline`, and `\underline`; at present this minimum distance depends only on the thickness of the line." (The first statement under the heading "Design errors that are too late to fix" in [5] is very similar.)

What does that mean? Let's look at the quotient `$1\above` x`pt 2$` with x going from 0.4 to 8 in steps of 0.4. In `\displaystyle` we get:



We see a less drastic effect in `\textstyle`:



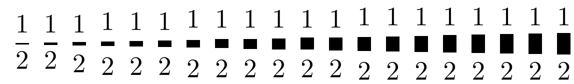
The analog sequences created using `\scriptstyle` and `\scriptscriptstyle` are quite similar to the sequence for `\textstyle`.

Quotients up to $x = 1.6$ seem to look acceptable, so a writer of a textbook on elementary mathematics can choose from clearly distinct line thicknesses ([1], p. 143). Nevertheless, the situation is disappointing. Can the macro `\ooalign` help?

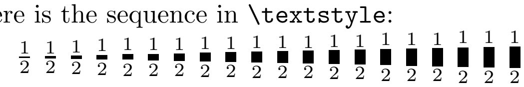
This is the idea: We can set the fraction with a small size for the line thickness and then we overprint this thin line with a line of the desired thickness. We assume that (a) the fraction is enclosed by `\lA` and `\rA` instead of braces and (b) `\mA` is put after the line thickness. Macro `\lA` splits the input and uses the macro `\mathpalette`, whose second argument collects the input. But this input must be split again to be usable for the main macro `\Aalign`.

```
\dimendef\Adim=0 % the new line thickness, local
\def\lA#1\above#2\mA#3\rA{% read fraction
   \mathord{\mathpalette\Aargs{{#1}{#2}{#3}}}}
\def\Aargs#1#2{\Aalign#1#2\end}% #1: math style
\def\Aalign#1#2#3#4\end{% #1: math style;
% #2: numerator; #3: dimen<=8pt; #4: denominator
 % guess an acceptable value for \above's dimen
 % based on the integer part and the 1st decimal
 % of the given dimen (compared to unit ``pt'')
{\Adim=\ifx#1\displaystyle
   \ifdim #3<1.5pt #3\relax
   \else\ifdim #3<2.5pt 1.5pt
   \else\ifdim #3<5pt 1.75pt
   \else 2pt \fi\fi\fi
 \else % the other styles need more cases
   \ifdim #3<1.5pt #3\relax
   \else\ifdim #3<2pt 1.5pt
   \else\ifdim #3<3pt 1.75pt
   \else\ifdim #3<4pt 2pt
   \else\ifdim #3<5pt 2.25pt
   \else\ifdim #3<6pt 2.5pt
   \else\ifdim #3<7pt 3pt
   \else 3.5pt \fi\fi\fi\fi\fi\fi\fi   \fi%\ifx
 \ooalign{% build fraction thrice: 1) output
   % fraction, 2) overprint line, 3) fix depth
   $#1{#2\above\Adim #4}$\cr  % use a thin line
   $#1{\phantom{#2}\above#3\phantom{#4}}$\cr
   $#1{\phantom{#2}\above\Adim\phantom{#4}}$}}}
```

Next, let's look at the two figures from above using `\Aalign`. This time I entered the quotients as `\lA 1 \above 0.4pt\mA 2\rA`, etc.



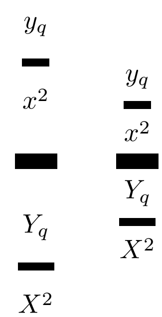
Here is the sequence in `\textstyle`:



And here is another comparison between TEX's default output and the one from `\Aalign`. The code

```
$${\displaystyle{y_q \above1mm x^2}\above2mm
   \displaystyle{Y_q \above1mm X^2}}    \qquad
\lA{\displaystyle\lA y_q \above1mm\mA x^2\rA}
   \above2mm\mA
   {\displaystyle\lA Y_q \above1mm\mA X^2\rA}
\rA$$
```

outputs



Clearly, there is an improvement. Nevertheless, the values for `\Adim` should be studied in more depth if you want to use this trick. The excessive use of `\mathpalette` — the `\phantom` macro ([1], p. 360) applies it too — makes it an expensive procedure.

References

[1] Donald E. Knuth, *The TEXbook*, Volume A of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1984.

[2] Donald E. Knuth, *Computer Modern Typefaces*, Volume E of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986.

[3] Donald E. Knuth, "The Final Errors of TEX", Chapter 34 of [4], 655–662, written August 1998.

[4] Donald E. Knuth, *Digital Typography*, Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes No. 78, 1999.

[5] Donald E. Knuth, "`tex82.bug`", file in the distributed errata for *Computers & Typesetting*. `ctan.org/tex-archive/systems/knuth/dist/errata/tex82.bug`

[6] Pierre A. MacKay, "Recycled METAFONT", *TUGboat* **15**:4 (1994), 444–446. `tug.org/TUGboat/tb15-4/tb45mack.pdf`

◇ Udo Wermuth
Dietzenbach, Germany
u dot wermuth (at) icloud dot com

Vertical alignments in plain TEX

Udo Wermuth

Abstract

Alignment structures are created in plain TEX using one of three methods: tabbing, horizontal or vertical alignment. Tabbing might be the easiest method; horizontal alignment with `\halign` has more complicated instructions. Both methods are described in great detail in Chapter 22 of *The TEXbook*. Vertical alignment with `\valign` is only briefly mentioned in *The TEXbook*; this might be the reason why it's rarely used. This article looks at `\valign` and presents a few tables that might benefit from it.

1 Introduction

The TEXbook [4] not only contains an in-depth description of plain TEX, its author Donald E. Knuth also writes in a style that is often entertaining. Or in his own words from page iv: "Computer system manuals usually make dull reading, but take heart: This one contains JOKES every once in a while, so you might actually enjoy reading it. (However, most of the jokes can only be appreciated properly if you understand a technical point that is being made— so read *carefully*.)" So you directly get a taste of his humor. But it might not always be obvious if he is joking. Neither the index nor exercise 27.5 helps us; we must find the jokes through our own reading.

One famous sentence that some readers might classify as a joke appears on page 249 in the *The TEXbook*: "People usually work with TEX at least a year before they find their first application for `\valign`; and then it's usually a one-row '`\valign {\vfil#\vfil\cr...}`'." In [3] T. Hoekwater comments on this statement: "Well, it took me a lot longer than that: nearly three decades, in fact." And M. Clark makes a similar judgment about Knuth's assessment in [2, p. 231]: "This might even be a conservative estimate. The only examples I have ever seen were rather forced (and could probably have been done another way)." (See also the appendix.)

This article looks at this rarely used command `\valign` [4, p. 249, p. 285]. It's implemented in the program TEX, i.e., it is one of TEX's *primitives*. Knuth programmed `\valign` and its sibling `\halign` to create vertical and horizontal alignments, respectively. No reader can identify the used primitive only from a typeset result: All alignments contain rows and columns and might have horizontal and/or vertical rules. With both commands one can create the image of joined cells across columns or rows. The only difference that I see between the two primitives is the amount of input needed to realize a certain output. Clark's statement seems to be correct and is valid for both. This text presents nine alignments with `\valign`. I find it not too difficult to realize them with `\halign` although the solutions are less pleasant to code and need sometimes more code.

Contents. The text expects that the reader is familiar with horizontal alignments created with the primitive `\halign`: Section 2 explains `\valign` for someone who knows `\halign`. Section 3 introduces a few macros that I find useful when I work with `\valign`; they are applied in sections 4–7. Next, in section 4 we look at three tables that seem to be more easily realized with `\valign` than `\halign`. Section 5 contains a table with spanned rows; a simple thing for `\valign`. The topic of section 6 is ruled tables. In section 7 `\halign` and `\valign` work together to build alignments. Finally, in section 8, I briefly mention unusual applications of `\valign`.

2 From horizontal to vertical alignment

TEX provides the primitive `\valign` to create vertical alignments. There is a close relationship between `\halign` and `\valign`. To describe what the latter primitive does, one can take the description of the former and exchange "horizontal" with "vertical" and "row" with "column" if the last two refer to the typeset output. For example, the sentence

> The primitive `\halign` is a vertical command that expects horizontal material in its entries: Each input row stands for the column entries in one row of the alignment.

becomes

> The primitive `\valign` is a horizontal command that expects vertical material in its entries: Each input row stands for the row entries in one column of the alignment.

These exchanges describe most aspects that one has to know to work with `\valign`. But the details are more complex and the consequences of this transposition should be explicitly stated at least once.

1. `\valign{...}` is a horizontal command: The command can be used inside a paragraph.

⇒ As a consequence the command starts a paragraph; thus TEX indents the alignment. One must start with '`\noindent\valign`' to get an alignment that's allowed to cover the full page width.

It also means that `\leftskip` & `\rightskip`, `\hangindent` & `\hangafter`, and `\parshape` influence the position of the whole alignment.

2. TEX rejects `\valign` in math modes. Thus, it can't be used for *alignment displays* ([4, p. 190]).

doi.org/10.47397/tb/44-3/tb138wermuth-valign

3. `\valign` has the same two-part structure as `\halign`: After a mandatory *preamble* with at least one *template*, any number of input rows with cell entries can follow. The number of templates must not be less than the number of entries in any input row.
⇒ The templates are now applied to the rows of the typeset columns.

4. '`#`', '`&`', `\cr`, and `\crcr` are used as in `\halign`. The last three don't structure the current alignment if they appear after a yet unmatched '`{`' inside `\valign`'s braces. (1) A `\cr`/`\crcr` ends the preamble and each input row, templates and cell entries are separated by the *alignment tab* '`&`', and the *parameter symbol* '`#`' defines the unique place in a template to insert a cell entry. (2) A *periodic preamble* has an additional '`&`' in front of one template: TeX repeatedly reuses the templates up to the preamble's end if it needs more for an input row. (3) Starting with the preamble's end, TeX expands the token after each (i) `\cr`/`\crcr` and (ii) '`&`' until it finds an unexpandable one. TeX checks if `\omit` or `\noalign` (only with case (i)) follows. (See nos. 8 and 10.)

5. The cell entries expect vertical material: Imagine that TeX builds a group through a `\vbox` with depth 0 pt. The vboxes are put together horizontally.
⇒ One consequence is that a letter (and other items; see [4, p. 283]) that starts a cell entry switches to horizontal mode, i.e., TeX starts a paragraph. As its width TeX uses the current width of the context, i.e., usually the full `\hsize`. Thus, with more than one column the `\hsize` must be changed locally.

A longer text should be placed in a `\vtop`, in which the `\hsize` might be reduced.

Single lines with horizontal material for the row entries should use an `\hbox`. This allows setting the column width using the keyword '`to`'; see no. 6.

In the case of entries consisting simply of hboxes that form the column we need `\strut` commands in each `\hbox`. Otherwise the distances between the hboxes might become unequal because TeX doesn't apply *interline glue* [4, p. 78] as in a paragraph.

6. In a `\valign` TeX computes and unifies the row heights, not the column widths.
⇒ The column widths must be given to TeX by other means if the text should be centered or set flush right. Using an `\hbox`, as suggested in no. 5, one could set the width of the column in the template. For example, use a construction like '`\hbox to \columnwidth`' in which the dimension register `\columnwidth` is set to an appropriate value.

7. `\tabskip` now specifies the distance between the rows. Its value outside of `\valign` creates white space above the table. Its last value adds white space after the alignment. As in an `\halign` the value must be changed inside the preamble.
⇒ However, it's a standard glue specification, not comparable to interline glue. (See also no. 5.)

8. `\omit` at the start of a cell entry works as before: TeX ignores the associated template from the preamble and uses '`#`' instead.

9. `\span` works as before: In the preamble it expands the following token; in input rows for cells it replaces '`&`' and combines the cells. Up to 255 `\span` commands can be used in sequence between cells.
⇒ Of course, in a `\valign` rows are spanned. The macro `\multispan` [4, p. 243] can be used if several row entries in a typeset column have to be combined.

10. `\noalign{...}` is used to insert horizontal material after a column of the alignment; it's allowed after a `\cr`/`\crcr` or another `\noalign`. A `\noalign` after the preamble's end puts its material to the left of the table. After the '`}`' TeX expands the next token to check for `\omit` or `\noalign`; cf. no. 4(3).
⇒ It can be used to specify the horizontal distance between the columns of the alignment comparable to `\tabskip` in an `\halign`. But there is no automation: TeX must get a value after each `\cr`.

11. `\everycr` works as before: Its tokens are inserted after every `\cr` and after a `\crcr` that's not immediately preceded by `\cr`, `\crcr`, or `\noalign`.
⇒ An '`\everycr={\noalign{\hskip ...}}`' sets the distance of columns without the need to repeat `\noalign` after every `\cr`. An `\unskip` after the closing brace of the `\valign` eliminates the white space to the right of the table. But see also no. 10.

12. Vertical rules are now easily drawn: '`\noalign {\vrule}`' after a `\cr` puts a rule between two columns. After the `\cr` of the preamble this construction draws a rule to the left of the alignment.
⇒ As vertical rules are nowadays mostly applied to avoid confusion [1, 13.53], the advantage to draw them easily might not be very important.

After the definition of `\vrulefill` — the plain format doesn't define this command — $n \leq 256$ rows receive a vertical rule with the command sequence '`\multispan{`n`}\vrulefill`', i.e., this is the vertical analog of one of the methods to get horizontal rules in `\halign` [4, p. 246]. (See also no. 9.)

Horizontal rules are created by an '`\hrule#`' in a template like vertical rules in `\halign` [4, p. 246]. But there is often a second method if hboxes are used (see no. 5): Use '`\omit\hrule`' in a cell entry.

13. Like `\halign`, `\valign` accepts the keywords '`to`' and '`spread`' followed by a dimension.
⇒ Now these keywords set or change the height of the overall alignment, not its width.

Table 1: TEX's error messages for bad input with commands for vertical alignment

#	The error is ...	Example of user input	TEX's (first) error message
1	a `\valign` in math mode	`$\valign{`... or `$$\valign{`...	`Missing $ inserted`
	... made in the construction of `\valign`:		
2	a forgotten keyword	`\valign 100pt{`...	`Missing { inserted`
3	a forgotten dimension	`\valign to {`...	`Missing number, treated as zero`
4	a forgotten `{`	`\valign #`...	`Missing { inserted`
5	a forgotten `}`	`\valign{#\cr X\cr\bye`	`You can't use '\end' in internal vertical mode`
	... made in the construction of the preamble:		
6	a missing preamble	`\valign{}\bye`	`Runaway preamble? with }` `Forbidden control sequence found while scanning preamble of \valign` inserted: `\cr }` read again: `\bye`
7	a forgotten `\cr`	`\valign{#}\bye`	`Runaway preamble? with }` (see #6)
8	an additional `}`	`\valign{#}\cr X\cr}\bye`	`Runaway preamble? with }\cr X\cr } \par` (#6)
9	an additional `{`	`\valign{#{\cr X\cr}\bye`	`Runaway preamble? with {\cr X\cr } \par` (#6)
10	a forgotten `#`	`\valign{\cr` ...	`Missing # inserted in alignment preamble`
11	a forgotten `&`	`\valign{##\cr` ...	`Only one # is allowed per tab`
12	an incomplete `\tabskip`	`\valign{#\tabskip\cr` ...	`Missing number, treated as zero`
13	a missing unit	`\valign{#\tabskip=0\cr` ...	`Illegal unit of measure (pt inserted)`
14	a second `&&` in preamble	`\valign{#&&#&&#\cr` ...	`Missing # inserted in alignment preamble`
15	an `\omit` in preamble	`\valign{\omit#\cr x\cr}`	`Misplaced \omit`
16	a `\noalign` in preamble	`\valign{#\noalign`...	`Misplaced \noalign`
	... made in the contents of a cell entry:		
17	an unmatched `{`	`\valign{#\cr {X\cr}`	`Missing } inserted`
18	an unmatched `}`	`\valign{#\cr X}\cr}`	`Missing \cr inserted`
19	a `#` in the entry	`\valign{#\cr #`...	`You can't use 'macro parameter #' in internal vertical mode`
20	an additional `&` or `\span`	`\valign{#\cr X&X\cr` ...	`Extra alignment tab has been changed to \cr`
21	an `\omit` that's not first	`\valign{#\cr X\omit`...	`Misplaced \omit`
22	the use of `\noalign`	`\valign{#\cr X\noalign{}`...	`Misplaced \noalign`
23	the use of `\raise` or `\lower`	`\valign{#\cr \lower...`	`You can't use '\lower' in internal vertical mode`
24	the use of `\spacefactor`	`\valign{#\cr \showthe\spacefactor`...	`Improper \spacefactor`
25	a badly used `\string\cr`	`\valign{#\cr \tt\string\cr\cr}`	`Misplaced \cr` & `Missing \cr inserted` (100 times, i.e., an endless loop)
26	the use of the template's end although it's like `\outer` [4, p. 206]	`\valign{#\cr \def\\#1{}\\\cr}`	`Runaway argument?` `Forbidden control sequence found while scanning use of \\` inserted: `\par` read again: `\endtemplate`
	... made in the construction of a `\noalign`:		
27	a missing `{`	`\valign{#\cr\noalign X}\cr}`	`Missing { inserted`
28	a missing `}`	`\valign{#\cr\noalign{X\cr}`	`Misplaced \cr`
29	a vertical command	`\valign{#\cr\noalign{\vss}}`	`Missing } inserted`
30	the use of `\moveleft` or `\moveright`	`\valign{#\cr \noalign{\moveleft`...	`You can't use '\moveleft' in restricted horizontal mode`
31	the use of `\prevdepth`	`\valign{#\cr \noalign{\showthe\prevdepth`...	`Improper \prevdepth`
	... made by an unsuitable `\everycr` for the `\valign`; here with the simple `\valign{#\cr}`:		
32	a missing `\noalign`	`\everycr={\hbox{x}}`	`Missing \cr inserted` (100 times)
33	a recursive `\everycr`	`\everycr={\noalign{x}\cr}`	`TeX capacity exceeded, sorry`
	... made by a devious user who deserves no sympathy [4, p. 299]:		
34	the improper nesting of alignments	`\valign{\halign#\cr\cr}`	`Emergency stop (interwoven alignment preambles are not allowed)`
35	too many `\span` in series	`\valign{&#\cr\multispan{257}`...	`This can't happen (256 spans)` (a joke?!)

As with `\halign` [4, p. 238] `\tabskip`'s glue specification (see no. 7) must be able to stretch or shrink to fulfill the request; otherwise TeX responds with an underfull vbox warning and writes a *prototype column* [4, p. 302f.] in the log file.

For example, with '`\tabskip=0pt`' a '`\valign spread 1sp{#\cr \vbox to 1.1\vsize{\vss}\cr}`' creates an underfull vbox warning as `\tabskip` cannot stretch. The overfull vbox is reported at the next page break.

14. With `\valign`, alignments with overfull typeset lines cause an overfull `\hbox` warning.
⇒ TeX builds a paragraph and the line-breaking procedure triggers the "`Overfull \hbox`" warning.

For example, set '`\hsize=10pt\tabskip=0pt`' and TeX accepts '`\halign{&#\cr u&w\cr}`'. But it complains about the identical looking output created with '`\noindent\valign{#\cr \hbox{u}\cr \hbox{w}\cr}`' and prints an overfull rule.

15. Alignments don't require programming in TeX but the rules for their construction must be strictly obeyed. Table 1 presents examples of what can be done wrong with a `\valign` and how TeX responds.

16. `\valign` is a horizontal command, so it interacts with the special integer `\spacefactor` [4, p. 76]; `\halign` interacts with `\prevdepth` [4, p. 79f.]. The `\spacefactor`'s value in front of the `\valign` is used for a `\noalign` directly after the preamble. After a column the `\spacefactor` is 1000. The `\noalign` that follows another `\noalign` gets the former's final value. At `\valign`'s end the current `\spacefactor` is used, i.e., either 1000 after a column, a `\noalign`'s final value, or the value from the start if the `\valign` ends after the preamble. (See [4, p. 286].)

17. To look into a vertical alignment, use the command `\showlists` [4, p. 88f.], either with the `\ddt` technique [4, p. 248] or insert, for example, the following macro, say as `\showalign1`, before the closing brace of the alignment.

```
\def\showalign#1{% #1: value of \showboxdepth
% call it after a \cr; shows unset boxes in log
 \noalign{\scrollmode \showboxbreadth=100
  \showboxdepth=#1 \showlists \errorstopmode}}
```

There is one surprise: Spanned rows (see no. 9) are reported as "columns". This word is hardcoded in §185 of *TeX: The Program* [5].

3 Macros for vertical alignments

From the list in the previous section we must conclude that vertical alignments require some parameters and macros to ease the coding. At least this is necessary if the typeset cell entries are text or numbers. Most of the following code snippets are of general use. Maybe only the last one is more specialized and might need a change if you want to apply its code in other texts.

First we need to control the widths of certain columns; see section 2, no. 6. I declare a dimension register for this task.

```
\newdimen\Vcolumnwidth % set the column width
```

The entries themselves must often use an `\hbox` of the current width and they should have a `\strut`; see section 2, no. 5. Therefore, I use the following macros to work with the width. Note, I use short names for the control words as they are often used. They all start with a capital 'V'.

```
\def\VR#1{% #1: text, right aligned
 \hbox to \Vcolumnwidth{\strut\hss#1}}
\def\VC#1{% #1: text, centered
 \hbox to \Vcolumnwidth{\strut\hss#1\hss}}
\def\VL#1{% #1: text, left aligned
 \hbox to \Vcolumnwidth{\strut#1\hss}}
```

Sometimes the width doesn't play a rôle. As the command `\omit` works as before (see section 2, no. 8) I define another macro that excludes the template. But an hbox and a `\strut` are still required.

```
\def\Vo#1{% #1: text, output without template
 \omit\hbox{\strut#1}}
```

Besides `\vrulefill` a macro without an hbox and a strut is needed for horizontal rules; see section 2, no. 12.

```
\def\vrulefill{\leaders\vrule\vfill}% no 'V'
\def\Vh{\omit\hrule}% a horizontal rule
```

To span several columns, for example, in the headers, I use the following macro that uses an hbox whose width is a multiple of `\Vcolumnwidth`.

```
\def\Vc(#1)#2{% #1: # columns to span; #2: text
 \omit\hbox to 0pt{\hbox to #1\Vcolumnwidth{%
   \hss\strut#2\hss}\hss}}% no effect on 1st col
```

To initialize the dimension `\Vcolumnwidth` I provide two methods. The first is to specify the value, in "pt" units, directly in a cell entry.

```
\def\VI#1pt{% #1pt: a valid dimension for the
 \global\Vcolumnwidth=#1pt }%    column width
```

The second method is useful for column widths of numeric data. As the digits have all the same width only the maximum number of digits in the column must be given. With a second argument of '1' it can be used for text strings too.

```
\newdimen\Vcolwdmodule % one unit times a factor
\def\Vi(#1){% #1: text; its width is the unit
 {\setbox0=\hbox{#1}%
  \global\Vcolwdmodule=\the\wd0}}
\def\VII(#1){% #1: number; the factor
 \global\Vcolumnwidth=#1\Vcolwdmodule}
```

Glue between the columns is set by `\noalign`; see section 2, no. 10. Note, `\VD` uses 'PT' as an argument delimiter in the glue's last component.

```
\def\VD#1PT{% #1: glue between columns
 \noalign{\hskip #1pt}}% valid after a \cr/\crcr
\def\Vd#1pt{\VD#1PT}% fixed width case
```

Longer text should be placed inside a `\vtop` (see section 2, no. 5) and the next macro allows for typesetting justified or ragged right. This macro must be called in the preamble with `\span`.

```
\def\Vv(#1)(#2){%#1: hsize frac; #2:rskip w/o em
 \vtop{\hsize=0.#1\hsize \parindent=0pt
  \leftskip=0pt \rightskip=#2em minus #2em
  \strut##\strut}}%usage: in preamble with \span
```

As said above, `\multispan` works with `\valign` and I created a variant tailored for this text that puts lines across rows. I assume that `\tabskip` and `\baselineskip` have only natural width.

```
\newtoks\Vlft \newtoks\Vrt % surround the rows
\def\Vmultispan#1/#2{% #1: # rows; #2: # lines
% center text lines of \Vmultirows with \vcenter
% (in an \hbox) of height (stored in \dimen255):
% #1\baselineskip + (#1-1)\tabskip
 \multispan{#1}\bgroup\dimen255=#1\baselineskip
 \advance\dimen255 by #1\tabskip
 \advance\dimen255 by -\tabskip
 \hbox\bgroup $\the\Vlft \vcenter to \dimen255
  \bgroup\vss \count255=#2\vbox\bgroup
   \Vmultirows}%  the macro leaves 4 groups open
\newdimen\Vrowskip % skip between text lines
\def\Vmultirows#1{%#1: a text line; calls itself
 \hbox{\strut#1}\advance\count255 by -1
 \ifnum\count255=0 %all lines done; close groups
  \egroup \vss\egroup \the\Vrt $\egroup \egroup
 \else\vskip\Vrowskip\expandafter\Vmultirows\fi}
```

4 Examples for `\valign`

Let's state one point directly at the beginning. Although every alignment can be implemented with `\valign`, one should only use this primitive if it provides an advantage over `\halign`.

When we look at the list that was presented in section 2 on how `\valign` works, it becomes clear how an alignment should look to be a good use case. The typeset data should be connected column-wise and each row should have a different format. Table 2 fulfills these criteria.

Table 2: Some numbers in four positional systems

Decimal	10	100	991
Binary	1010	1100100	1111011111
Octal	*12*	*144*	*1737*
Hexadecimal	0xA	0x64	0x3DF

Here is one implementation with `\halign`.

Example 1: Implementation with `\halign`

```
\centerline{\bf Table 2: \rm Some numbers in
   four positional systems}\medskip
\everycr={}\tabskip=0pt % initialize
\centerline{\vbox{\halign{#\hfil\hskip4pt&&
     \hskip4pt\hfil#\cr % periodic preamble
   Decimal&         10&        100&
                991\cr \noalign{\vskip3pt}
   Binary&    \tt 1010&\tt 1100100&
      \tt 1111011111\cr \noalign{\vskip3pt}
   Octal&     \it 12\/&  \it 144\/&
          \it 1737\/\cr \noalign{\vskip3pt}
   Hexadecimal&\tt 0xA&   \tt 0x64&
           \tt 0x3DF\cr}}}
```

Note, I keep '`\tabskip=0pt`' to be able to use a periodic preamble and perfect centering, i.e., the values of the first and last application of `\tabskip` must be identical. I cannot write all binary numbers in one physical row because of the journal's column width. But even with another width the line break is required if a few more numbers are added.

The table can be easily implemented with the primitive `\valign` if we use the macros from the above described toolset. Here numeric data is used so the column width gets determined by the number of digits in the binary representation. This number is specified as a table entry with the template `\VII(#)`. The other entries use the macro `\VR` to keep this column width. The exception is, of course, the first typeset column, in which we just output hboxes without the template.

Example 2: Implementation with `\valign`

```
\centerline{\bf Table 2: \rm Some numbers in
   four positional systems}\medskip
\everycr={}\tabskip=0pt \Vi(\tt0)%  initialize
\centerline{\valign{\VII(#)&\tabskip=3pt
   \VR{#}&\VR{\tt#}&\VR{\it#\/}&
   \VR{\tt0x#}\tabskip=0pt\cr% end of preamble
  0&\Vo{Decimal}&\Vo{Binary}&\Vo{Octal}&
    \Vo{Hexadecimal}\cr           \Vd8pt
  4& 10&         1010&   12&   A\cr\Vd4pt
  7& 100&     1100100&  144&  64\cr\Vd4pt
 10& 991& 1111011111& 1737& 3DF\cr}}
```

We moved the format of the entries into the templates so that the entries are easier to read. The entries in an input row belong together and are more easily understood than in the version with `\halign`.

Imagine adding another typeset column for the decimal number 128. In example 2 this is easy: Add the line "`8& 128& 10000000& 200& 80\cr\Vd4pt`" after the line for 100. In example 1 we have to work harder: All input rows must be changed, as each row receives one new entry.

In a new table we could turn the first column into headers; this supports `\halign`. But such a table cannot fill the double-column width if we add more numbers; `\valign` can do that better.

Of course, the general idea is to design an alignment in such a way that its contents communicates best with the readers. But the world isn't ideal and every alignment must also obey the constraints given by the available space, the document structure, or a journal's style guide. And there might be aesthetic reasons to format an alignment horizontally or vertically.

Let's look at an example that illustrates what I mean with the last sentence. The following table fits into the width of this journal's columns.

Table 3: No. of *TUGboat* articles by rating in 2022

	Issues of Volume 43		
Category	**43:1**	**43:2**	**43:3**
Reports and Notices	9	13	8
Introductory	4	4	5
Intermediate	3	12	8
Intermediate Plus	2	4	9
Advanced	3	3	6

Of course, this table can be implemented with `\halign` but less comfortably. With `\valign` I take one issue, count the articles on the page where the articles are rated by difficulty, and type one input row that becomes a typeset column. With `\halign` I have to add to each input row one number.

A solution with `\halign` should have the first column as headers and the issues as rows. This table is wide and short: The headers dominate the column widths. The numbers in the cells aren't equidistant except one works with `\hidewidth` for the headers and an extremely stretching `\tabskip`. Would such a table look aesthetically pleasing? Does it help to turn two of the categories into two-line headers?

Here is the implementation with `\valign`. It shows how `\Vc` builds the centered header. Note, this works because the columns have no additional skips between them. Also note, the table has an additional 3 pt space between caption and first header as well as an additional 1 pt fixed-width distance between all other rows and after the table because of the two `\tabskip`s.

Example 3: Implementation of Table 3

```
\medskip\centerline{\spaceskip=3.2pt\bf Table 3:
 \rm No.\ of \TUB\ articles by rating in 2022}
\everycr={}\tabskip=3pt \Vcolumnwidth=35pt
\valign{\tabskip=1pt&\VC{#}\cr % periodic
  &\Vo{\bf Category}&\Vo{Reports and Notices}&
   \Vo{Introductory}&\Vo{Intermediate}&
   \Vo{Intermediate Plus}&\Vo{Advanced}\cr
  \Vc(3){\bf Issues of Volume 43}& % centered
   \bf 43:1&        9& 4&        3& 2& 3\cr
  &\bf 43:2&\llap{1}3& 4&\llap{1}2& 4& 3\cr
  &\bf 43:3&        8& 5&        8& 9& 6\cr}
```

The next example uses a `\vtop` through the macro `\Vv` from the toolset as longer text is output; see Table 4. The table fills the full `\hsize`.

Table 1, #25, shows that we cannot use '`\tt \string\cr`' in a cell entry. Therefore, I define the macro `\(...)` to typeset reserved tokens like `\cr`.

Digression: Note, however, because of section 2, no. 4(3), '`\noindent\valign{\tt#\cr \string\cr \cr}`' is valid input. In this case `\string` is expanded by TeX early as it appears after a `\cr` and it digests the next token, the `\cr`.

Example 4: Table 4; some text is omitted

```
\def\(#1){% #1: text; output \#1 in \tt
  {\tt\string\ \unskip#1}}
\tabskip=8pt %    space between title and table
```

Table 4: Alignment methods in TeX

Method	*Tabbing*	*Horizontal/vertical alignment*
Description	Sets fixed width columns via tabulators to align text that fits into the column width line by line. A user can create arbitrarily long alignments with this method. Three methods are available to set the tabulators: (1) a fixed number of columns, (2) column widths via a sample line, and (3) remove and set tabulators on the fly. It is implemented in plain TeX, using macros.	TeX calculates the necessary width (for horizontal)/height (for vertical) of each column/row after it reads all input lines of the alignment; thus TeX needs a lot of memory to store a very long alignment. Each column/row gets an individual format for its material through templates collected in the preamble of the alignment. This is a built-in function of the program TeX.
Used tokens	`\settabs`, `\columns`, `\+`, `&`, `\cr`, `\cleartabs`	`\halign`/`\valign`, `&`, `#`, `\tabskip`, `\cr`, `\crcr`, `\span`, `\omit`, `\noalign`

Table 5: Elementary properties of the digits

Digit	**Parity**	**Factors**	**One unique property**
0	even	neither prime nor composite	additive identity: $x + 0 = x$
1	odd		multiplicative identity: $x \times 1 = x$
2	even	prime	only 2 obeys $x + x = x \times x = x^x$
3	odd		unique sequence: prime followed by a square
4	even	square: 2×2	
5	odd	prime	unique: member of two twin prime pairs: 3 and 5, 5 and 7
6	even	composite: 2×3	smallest perfect number: $1 + 2 + 3 = 6$
7	odd	prime	smallest: not sum of < 4 squares as $1^2 + 1^2 + 1^2 + 2^2 = 7$
8	even	cube: $2 \times 2 \times 2$	unique: two integer powers that differ by 1 (nonelementary)
9	odd	square: 3×3	

```
\centerline{{\bf Table 4:}
  Alignment methods in \TeX}
\noindent\valign{\tabskip=10pt %   row distance
  \hbox{\strut\it #}&\span\Vv(415)(0)\vfil&
    \span\Vv(415)(3)\vfil\cr         % preamble
 \Vo{Method}&\Vo{Description}&
   \Vo{Used tokens}\cr              % 1st column
\VD 10pt plus 5pt minus 5PT % glue between cols
 Tabbing&
 Sets fixed width columns via tabulators to ...
  (3)~remove and set tabulators on the fly.\par
  It is implemented ...&
 \(settabs), \(columns), \(+), {\tt\&}, \(cr),
  \(cleartabs)\cr
\VD 20pt plus 10pt minus 10PT %   glue doubled
 Horizontal/vertical alignment&
 \TeX\ calculates the necessary width (for ...
  in the preamble of the alignment.\par
  This is a built-in function of the program
  \TeX.&
 \(halign)/\(valign), {\tt\&}, {\tt\#},
  \(tabskip), \(cr), \(crcr), \(span), \(omit),
  \(noalign)\cr}
```

The nice thing about the code in example 4 is that all elements of the alignment methods are kept together. That is, after the description of *tabbing* the relevant tokens are listed.

Of course, Table 4 can be implemented with `\halign`. Although I repeat myself, the input is less pleasant as it requires data from tabbing and alignment in each input row.

In this case, when we turn the first column into headers to improve the input, the typeset table looks unbalanced: a short text, a wide and long text, and a wide and short text. The third column should be made smaller but I'm convinced that such a table doesn't look better than Table 4.

5 Spanning rows with `\valign`

Another case for `\valign` are alignments in which rows have to be joined. Table 5 does not have a vertical structure but spanned rows. Three columns have their text left aligned and we need not calculate the column widths. Only the first column has to use `\Vcolumnwidth` and we can set it outside of the alignment.

I should mention that the use of the large brace forces us to assign `\tabskip` a certain value. The large delimiters aren't available in all sizes and so we have to make sure that two `\baselineskip` plus the `\tabskip` equal the height of an available brace.

Example 5: Implementation with `\valign`

```
\Vi(\bf Digit)\VII(1)% init for 1st column
\centerline{\bf Table 5: \rm Elementary
                properties of the digits}
\tabskip=6pt % get right distance for the \}
\Vlft={\left\}\mkern3mu}\Vrt={\right.}
\Vrowskip=2pt % initialization for \Vmultispan
\noindent\valign{&\hbox{\strut#}\cr % periodic
 \bf Digit& \VC{0}&\VC{1}&\VC{2}&\VC{3}&
   \VC{4}&\VC{5}&\VC{6}&\VC{7}&\VC{8}&\VC{9}\cr
\Vd6pt \bf Parity& even& odd&
  even& odd& even& odd& even& odd& even& odd\cr
\Vd6pt \bf Factors&
  \Vmultispan2/2{neither prime}{nor composite}&
  \Vmultispan2/1{prime}&
  square: $2\times2$&      prime&
  composite: $2\times3$&   prime&
  cube: $2\times2\times2$& square: $3\times3$\cr
\Vd10pt \bf One unique property&
  additive identity: $x+0=x$&
  multiplicative identity: $x\times1=x$&
  only 2 obeys $x+x=x\times x=x^x$&
  \Vmultispan2/1{unique sequence: prime
```

Table 6: Holidays in at least one federal state and well-known (local) celebration days in Germany, 2023

Day	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	New Year				Labor Day						All Saint's	
2												
3										Unity Day		Advent Sun
4												
5												
6	Epiphany			Holy Thu								
7				Good Fri								
8			Women's Day			Corpus Christi		(Augsburg)				
9				Easter Sun	Europe Day							
10				Easter Mon								
11											11.11.	
12												
13												
14			Pi Day		Mother's Day							
15					Kalte Sofie			Assumption Day				
16												
17			St. Patrick									
18					Ascension Day							
19												
20		Rose Mon							Children's Day			
21												
22		Ash Wed									Rep&Pray	
23					Constitution Day							
24												Heiligabend
25												Christmas
26												Boxing Day
27						Siebenschläfer						
28					Whit Sun							
29					Whit Mon							
30					(Wäldchestag)							
31										Reformation		Silvester

```
   followed by a square}&
unique: member of two twin prime pairs:
   3 and 5, 5 and 7&
smallest perfect number: $1+2+3=6$&
smallest: not sum of $<4$ squares as
   $1^2+1^2+1^2+2^2=7$&
\Vmultispan2/1{unique: two integer powers
   that differ by 1 (nonelementary)}\cr}
```

Note: The unique property for the digits 8 and 9 is called *Catalan's Conjecture*. Eugène Catalan published it in 1844; Preda Mihăilescu proved it in 2002.

6 Rules in vertical alignments

Some readers might be proud to have reached the level "TEX Master" [4, p. 245] and they might want to see how to earn this title with \valign.

Let's start with a calendar that shows all holidays for 2023 in Germany; see Table 6. (I made the output fit into this journal's page width. We want to typeset such a table; so it shouldn't be important that it isn't presented in one or two landscape pages.) Not all listed days are a holiday in every federal state. Some aren't a holiday at all and the days in parentheses are only locally celebrated.

With a \valign we create the table columnwise, i.e., month by month. Thus we can define macros that list the holidays in sequence to insert them into the table in the correct sequence just by calling the macro. The floating holidays are separately listed. Of course, they have to be reassigned every year.

Example 6: Holidays in Germany

With the macro \(...) I shorten some names in the output as the column width for each month in Table 6 is rather small and I want to avoid overprinting.

TEX input

```
\begingroup \eightpoint % see TeXbook, p.415
  \countdef\Hfcnt=1 \countdef\Hvcnt=3
  \countdef\Cfcnt=5 \countdef\Cvcnt=7
  \edef\RESETcounters{% save current values
```

```
    \global\count1=\the\count1
    \global\count3=\the\count3
    \global\count5=\the\count5
    \global\count7=\the\count7 }
  \global\Hfcnt=0 \global\Hvcnt=0
  \global\Cfcnt=0 \global\Cvcnt=0
\def\(#1){\ignorespaces}% to shorten names
\def\Hf{% public holidays with fix date
 \ifcase\Hfcnt New Year\('s Day)\or Epiphany\or
 Women's Day\or Labor Day\or (Augsburg\( Peace
 Festival))\or Assumption Day\or \(World)
 Children's Day\or \(German) Unity Day\or
 Reformation\( Day)\or All Saint's\( Day)\or
 Christmas\( Day)\or Boxing Day\fi
 \global\advance\Hfcnt by 1 }
\def\Hv{% public holidays with variable date
 \ifcase\Hvcnt Good Fri\(day)\or Easter
 Sun\(day)\or Easter Mon\(day)\or Ascension
 Day\or Whit Sun\(day)\or Whit Mon\(day)\or
 Corpus Christi\or Rep\(entance)\&Pray\(er Day)
 \fi \global\advance\Hvcnt by 1 }
\def\Cf{% special days with fix date
 \ifcase\Cfcnt Pi Day\or St.\ Patrick\('s Day)
 \or Europe Day\or Kalte Sofie\or Constitution
 Day\or Siebenschl\"afer\or 11.11.\or
 Heiligabend\or Silvester\fi
 \global\advance\Cfcnt by 1 }
\def\Cv{% special days with variable date
 \ifcase\Cvcnt Rose Mon\(day)\or Ash
 Wed\(nesday)\or Holy Thu\(rsday)\or Mother's
 Day\or (W\"aldchestag)\or Advent Sun\(day)\fi
 \global\advance\Cvcnt by 1 }
\def\tblstrut{% an own strut for this table
 \vrule height 8pt depth 4pt width 0pt }
\Vi(Day\ )% initialize first column; no \VII
\def\\#1#2{% #1, #2: digits
 \omit\hbox to\Vcolwdmodule{% use the unit
    \tblstrut\hfil\number#1#2\ }}
```

Now we just have to enter the macro names into a 31-day scheme for each month.

I prefer to draw only a few rules. So I use just one vertical rule and for each three-day group a horizontal rule. This is enough to identify the day of a holiday quickly.

The vertical rule is simply drawn by a `\vrule` in a `\noalign`; see section 2, no. 12. But note that the spacing to the left and the right is controlled in the macro `\\` and in the preamble, respectively.

The horizontal rules are included in the columns. This is possible as an `\hrule` gets the width of its context, i.e., here it's the width of the column. The `\Vh`s in the 5th, 9th, …, 41st entry of the months also help to find the entry for each day.

Example 6 continued: The calendar with `\valign`

```
\everycr={}\tabskip=0pt
\Vcolumnwidth=37.4pt % ridiculously small
\centerline{\bf Table 6: \rm Holidays in at
 least one federal state and well-known (local)
 celebration days in Germany, 2023}\medskip
\noindent\valign{\VL{\hskip1pt#}&& % first row
  \VL{\hskip1pt\tblstrut\sixrm#}\cr % periodic
\Vo{Day }& \\01&\\02&\\03&\Vh&\\04&\\05&\\06&
\Vh&\\07&\\08&\\09&\Vh&\\10&\\11&\\12&\Vh&\\13&
\\14&\\15&\Vh&\\16&\\17&\\18&\Vh&\\19&\\20&\\21&
\Vh&\\22&\\23&\\24&\Vh&\\25&\\26&\\27&\Vh&\\28&
\\29&\\30&\Vh&\\31\cr        \noalign{\vrule}
 Jan&\Hf&   &   &\Vh&   &   &\Hf&\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        &   &   &\Vh&   &   &   &\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        \cr
 Feb&   &   &   &\Vh&   &   &   &\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        &   &   &\Vh&   &\Cv&   &\Vh&\Cv&   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        \cr
...
 Dec&   &   &\Cv&\Vh&   &   &   &\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        &   &   &\Vh&   &   &   &\Vh&   &   &
     \Cf&\Vh&\Hf&\Hf&   &\Vh&   &   &   &\Vh&
     \Cf\cr}
\RESETcounters \endgroup
```

Other than example 6 the next example doesn't seem to benefit from `\valign`. Well, the table does not prefer `\halign` either. That is, there is neither a horizontal nor a vertical structure in Table 7.

Table 7: Lo-Shu

4	9	2
3	5	7
8	1	6

It's a magic square, i.e., sums of columns, rows, and diagonals always give the same result. It's probably been known in China for > 2000 years but received the above-stated name only centuries after its discovery.

I think Table 7 is a very good exercise for TeX Masters if they want to start to think in vertical structures. Here is an implementation with `\valign`; its output is shown in Table 7. Note, this time I set `\leftskip` in the code; see section 2, no. 1.

Example 7: Rules with `\valign`

```
\noindent\bf Table 7: \rm Lo-Shu\medskip
\def\balign{\leftskip=12pt \noindent}%       use
\def\ealign{\par\leftskip=0pt }\relax% \leftskip
\def\digstrut{% a strut to center the digits
  \vrule height 9.2pt depth 2.8pt width 0pt }
\everycr={}\tabskip=0pt \Vcolumnwidth=20pt
\balign\valign{\hrule#&\vskip1pt\hrule#&
  \VC{\digstrut#}&\hrule#&\VC{\digstrut#}&
  \hrule#&\VC{\digstrut#}&\hrule#&
  \vskip1pt\hrule#\cr% preamble with 9 templates
    \multispan9\vrulefill\cr         % left rule
    &\multispan7\hbox to1pt{\hss}&\cr      % gap
    &&\multispan6\vrulefill&\cr% inner left rule
```

Table 8: SI prefixes

Name	Symbol	Multiplying factor	Name	Symbol	Multiplying factor
quetta	Q	10^{30}	quecto	q	10^{-30}
ronna	R	10^{27}	ronto	r	10^{-27}
yotta	Y	10^{24}	yocto	y	10^{-24}
zetta	Z	10^{21}	zepto	z	10^{-21}
exa	E	10^{18}	atto	a	10^{-18}
peta	P	10^{15}	femto	f	10^{-15}
tera	T	10^{12} (trillion)	pico	p	10^{-12}
giga	G	10^{9} (billion)	nano	n	10^{-9}
mega	M	10^{6} (million)	micro	µ	10^{-6}
kilo	k	10^{3} (thousand)	milli	m	10^{-3}
hecto	h	10^{2}	centi	c	10^{-2}
deca	da	10^{1}	deci	d	10^{-1}

Source: `https://www.bipm.org/en/measurement-units/si-prefixes`

```
&&  4&&  3&&  8&&\cr     % first data column
&&\multispan6\vrulefill&\cr            % rule
&&  9&&  5&&  1&&\cr    % second data column
&&\multispan6\vrulefill&\cr            % rule
&&  2&&  7&&  6&&\cr     % third data column
&&\multispan6\vrulefill&\cr   % right border
&\multispan7\hbox to1pt{\hss}&\cr
\multispan9\vrulefill\cr}\ealign
```

In this table the above-mentioned `\vrulefill` is used. The horizontal rules are realized by the method that associates them with templates, i.e., it is the dual method used for the vertical rules in an implementation with `\halign`.

7 Combining both methods

Above, I wrote all alignments can be realized with `\halign`. I don't find it complicated if one follows the description in Chapter 22 of *The TEXbook* [4]. People who have difficulties with the required discipline of this approach can benefit from a combination of `\halign` and `\valign`. Let's study two cases.

As the first example let's look at a *double-up* table, like Table 8. It is one of the rare cases in which vertical rules are allowed according to [1, 13.53]. I implement the two halves of the table as horizontal alignments and a `\valign` glues them together with a vertical rule (see section 2, no. 12).

Example 8: Combine `\halign` and `\valign`

The font `cmmu10.mf` is built from the font `cmmi10.mf` by setting `slant:=0` and `math_fitting:=false`. Note, in a macro the '`#`' of an alignment must be doubled [4, p. 203].

TEX input

```
\font\tenmu=cmmu10 \let\hw=\hidewidth
\def\lhalf{\vbox{\offinterlineskip \tabskip=0pt
 \halign to 0.4\hsize{\tabskip=20pt
      \strut##\hfil&\hfil##\hfil& % Name, Symbol
      $10^{##}$\hfil\tabskip=5pt& % Factor
      ##\hfil\tabskip 0pt plus 100pt\cr % (text)
  Name\hw& \hw Symbol\hw&
   \omit\span Multiplying factor\hw\cr
 \noalign{\vskip 3pt}
  quetta& Q&  30\cr   ronna&  R&  27\cr
  yotta&  Y&  24\cr   zetta&  Z&  21\cr
  exa&    E&  18\cr   peta&   P&  15\cr
  tera&   T&  12& (trillion)\cr
  giga&   G&   9& (billion)\cr
  mega&   M&   6& (million)\cr
  kilo&   k&   3& (thousand)\cr
  hecto&  h&   2\cr   deca& \hw da\hw& 1\cr}}}
```

The main aspects of the preamble are repeated for the right half. The difference is that it has one less column because there is no "(text)".

Example 8 continued: The second half

```
\def\rhalf{\vbox{\offinterlineskip \tabskip=0pt
 \halign to 0.4\hsize{\tabskip=20pt
      \strut##\hfil&\hfil##\hfil& % Name, Symbol
      $10^{##}$\hfil              % Factor
        \tabskip=5pt plus 100pt\cr% no (text)
  Name\hw& \hw Symbol\hw&
   \omit Multiplying factor\hw\cr% no \span
 \noalign{\vskip 3pt}
  quecto& q& -30\cr   ronto&  r& -27\cr
  yocto&  y& -24\cr   zepto&  z& -21\cr
  atto&   a& -18\cr   femto&  f& -15\cr
  pico&   p& -12\cr   nano&   n&  -9\cr
  micro&  \tenmu\char22& -6\cr % upright $\mu$
  milli&  m&  -3\cr   centi&  c&  -2\cr
  deci&   d&  -1\cr}}}
```

(A side note: The organization wants us to typeset all physical prefixes upright, including the Greek μ for "micro". I created `cmmu10.mf` as a simple realization of an upright math italic font. It's sufficient for Table 8 and doesn't look too bad in text: ng, µg, mg. AMS Euler Roman looks too different: µg.)

The last step is to take the halves and create a one-row vertical alignment with two columns.

Example 8 continued: Application of \valign

```
\leftline{\bf Table 8: \rm SI prefixes}
\tabskip=4pt % space around alignment
\noindent\valign{#\cr % preamble
    \lhalf\cr
\noalign{\hskip 10pt\vrule\hskip 40pt}
    \rhalf\cr}
\par\leftline{{\it Source:}
  {\ninett https://www.bipm.org/en/%
   measurement-units/si-prefixes}}
```

Sure, this example is too simple to show all advantages of the method but I hope the reader understands the concept.

The next example is like Table 7: Neither alignment method is preferred. But it's clear that nested alignments help. I use `\valign` for the larger alignment as its output can be directly applied in a paragraph. There is no need to put the alignment into a `\vbox` with `\offinterlineskip`, which is necessary if `\halign` is used; see the code of the inner blocks. (Compare also to the discussion in the appendix.)

We want to typeset sudoku puzzles. As we have to typeset a lot of them in order to explain how to solve one, we implement a simplified input method that doesn't require us to use `\halign` or `\valign`. The idea is to enter the code on the left to get the result on the right:

```
\sudokublocks
~~3~~~5~~
974~~~613
8~~~~~~~4
8~14~97~2
3~9~~~5~1
4~25~19~6
2~~~~~~~4
156~~~237
~~8~~~1~~
~~~end~~~
```

		3	9	7	4	8		
5			6	1	3			4
8		1	3		9	4		2
4		9				5		1
7		2	5		1	9		6
2			1	5	6			8
		4	2	3	7	1		

Each input line represents one 3×3 block of the sudoku. A tilde is used for an empty cell. The sentinel `~~~end~~~` finishes the input. (The input encoding might not be the best choice, but, again, for this article it's sufficient and seems to require the smallest amount of extra code compared to other methods.)

First, we code the command that the users enter; it calls the macro which reads the input.

Example 9: Input for sudoku

```
\def\SPsudokuend{~~~end~~~}% the sentinel
\newif\ifSPbadinput % true: <> 9 input rows
\countdef\SPblockcnt=8 % counts the blocks
\def\sudokublocks{% user macro to start input
 \SPbadinputfalse  % assume all cells get input
 \begingroup           % closed in \SPblocksend
  \SPblockcnt=1 \def~{\phantom0}%~ = empty cell
  \let\next=\SPrdblock \next}% initialize \next
```

Next we read one input line representing one sudoku block. I assume the sentinel is always present.

Example 9 continued: Reading a sudoku block

```
\def\SPrdblock#1
{% #1: 9 elements (digits and ~) EOL delimited
 \def\nxt{#1}\ifx\nxt\SPsudokuend % the sentinel
   \let\next=\SPblocksend   % true: end of input
  \else   % store '.#1' in \SPblock<\SPblockcnt>
   \expandafter     % build the block macro name
   \def\csname SPblock\romannumeral\SPblockcnt
       \endcsname{%  we need at least 9 elements
    .#1{}{}{}{}{}{}{}{}{}}%empty groups are okay
   \ifnum\SPblockcnt<10 \advance\SPblockcnt by 1
   \else \SPbadinputtrue % >9 blocks is an error
  \fi\fi \next}
\def\SPblocksend{% prints sudoku if input okay
 \ifnum\SPblockcnt=10 \else \SPbadinputtrue \fi
 \ifSPbadinput              % not nine input lines
  \errmessage{You need exactly nine data rows}%
 \else \SPprint     % no error: print the puzzle
 \fi\endgroup}
```

Next we turn our attention to the topic of this article. First, I implement a 3×3 alignment with thin rules and centered digits. The macro gets nine arguments for the nine cells and builds one cell of the larger alignment. Note the parameter text starts and ends with a period. Later I explain why the first period is important in this implementation.

Example 9 continued: A sudoku block

```
\def\Vs{\vrule height9.2pt depth2.8pt width0pt }
\def\SPmkcell.#1#2#3#4#5#6#7#8#9.{% #1--#9:block
 \vbox{\offinterlineskip \tabskip=3.5pt
  \everycr={}\halign{% one 3x3 block, thin rules
      \Vs ##&& \vrule ##& \Vs ##\cr   % periodic
    #1&& #2&& #3\cr \noalign{\hrule}
    #4&& #5&& #6\cr \noalign{\hrule}
    #7&& #8&& #9\cr}}}
```

As in example 8, the '`#`' is doubled in the preamble as this is required in a macro to get a single '`#`'. (The strut `\Vs` centers the digits vertically.)

Finally, here is the macro that typesets the sudoku puzzle. Note the macro `\SPmkcell` expects expanded `\SPblock`⟨`\SPblockcnt`⟩ macros and as TeX expands a macro after '`&`' in the entries for an alignment (see section 2, no. 4(3)) we can use `\SPmkcell` in the preamble.

Example 9 continued: The complete sudoku

```
\def\SPprint{% typeset 9 \SPblock<\SPblockcnt>s
 \everycr={\noalign{\vrule width 1pt }}%
 \tabskip=0pt
  \valign{\hrule height 1pt ##&&\SPmkcell##.&
     \hrule height 1pt ##\cr % periodic preamble
   &\SPblocki  &&\SPblockiv&&\SPblockvii &\cr
   &\SPblockii &&\SPblockv &&\SPblockviii&\cr
   &\SPblockiii&&\SPblockvi&&\SPblockix  &\cr}}
```

Because of the `\SPblock`⟨*n*⟩s the necessary transposition of the input rows of `\sudokublocks` is easily accomplished.

Let's look at the important point of why we need the period in the `\SPblock...` macros. TeX expands after the '`&`' the argument of `\SPmkcell` to check if it starts with `\omit`. Without the period and with '`~`' as the first character in an input row, TeX expands this active character, finds no `\omit`, but leaves us with the seven tokens "`\let\if@=\iftrue \h@true\ph@nt0`", i.e., with the tokens from the expanded "`\phantom0`". These tokens throw errors as they aren't expected by `\SPmkcell`. The period avoids this problem.

Sometimes TeX's alignment methods are used for structures that don't look like a table. Next, let's study one such structure: a tournament. See Table 9 on the next page.

Example 10: Shortcuts for the teams

```
\def\AR{Argentina}    \def\AU{Australia}
\def\BR{Brazil}       \def\CH{Switzerland}
\def\EN{England}      \def\ES{Spain}
\def\FR{France}       \def\HR{Croatia}
\def\JP{Japan}        \def\KR{South Korea}
\def\MA{Morocco}      \def\NL{Netherlands}
\def\PL{Poland}       \def\PT{Portugal}
\def\SN{Senegal}      \def\US{USA}
```

Clearly, there are four columns in Table 9. Its typeset row elements are somewhat complex: The teams of the match appear above a rule and the result below it. Sometimes a vertical rule appears at the right end of such an element above and sometimes below the horizontal rule. And the elements for two matches, the finals, have no vertical rules.

To get a perfect alignment of the described elements two other elements are needed. One is completely empty but has the height and width of the other elements. The second is empty except for a vertical rule at the right end. The structure of these basic elements is realized by two `\halign`s.

Example 10 continued: The elements

```
\def\tblstrut{% use an own strut
  \vrule height 10pt depth 4pt width 0pt}
\def\Match#1#2-#3-#4#5{% #1, #2: teams;
 % #3: result; #4, #5: \relax or \omit for rule/
 % no rule at right of lines 1 (#4) and 3 (#5)
 \halign{\VC{\tblstrut##}&\vrule##\cr % preamble
  {#1}--{#2}&#4\cr    % the match (plus \vrule?)
  \multispan2\hrulefill\cr     % horizontal rule
  #3&#5\cr}}%          the result (plus \vrule?)
\def\Empty#1{% #1: \relax/\omit for rule/no rule
 \halign{\VC{\tblstrut##}&\vrule##\cr % preamble
  \relax&#1\cr
  \omit\phantom{\vrule height0.4pt}&#1\cr
  \relax&#1\cr}}
```

With the help of macros the five different cases can be easily implemented. Here I use short names as these macros are later used in the big alignment: 'f', 't', 'b' stand for "final", "top", and "bottom", respectively. A pair of "top" and "bottom" elements is connected via vertical rules as the winners of these matches build an element in the next column.

Example 10 continued: The five building blocks

```
\def\Mf#1#2-#3-{% #1, #2: teams; #3: result
   \vbox{\offinterlineskip\tabskip=0pt
      \Match{#1}{#2}-#3-\omit\omit}}%   no rules
\def\Mt#1#2-#3-{% #1, #2: teams; #3: result
   \vbox{\offinterlineskip\tabskip=0pt
      \Match{#1}{#2}-#3-\omit\relax}}
\def\Mb#1#2-#3-{% #1, #2: teams; #3: result
   \vbox{\offinterlineskip\tabskip=0pt
      \Match{#1}{#2}-#3-\relax\omit}}
\def\E{\vbox{% an empty block
   \offinterlineskip\tabskip=0pt\Empty\omit}}
\def\R{\vbox{% an empty block with a rule
   \offinterlineskip\tabskip=0pt\Empty\relax}}
```

The last step is to create the tournament with a `\valign`. I use a `\valign` as it represents the four columns better than an `\halign`. Each input row in the vertical alignment stands for a round of the tournament; in an `\halign` the input rows cross the tournament rounds.

Nevertheless, we must add the empty elements and thus, we have to do some counting in the columns. To make the input easier and shorter I decided to create blocks of three `\E` and `\R` as there is some pattern.

Note, the small column width stretches the input; otherwise top and bottom matches could be placed in one input row. Note also that the `\hsize` is 39 pc = 468 pt and this width minus 2 pt (1.6 pt are needed for the vertical rules) is distributed to the four columns; a tight fit.

Example 10 continued: Apply a `\valign`

```
\def\EEE{\E&\E&\E}\def\RRR{\R&\R&\R}% short-cuts
\centerline{\bf Table 9: \rm Knockout matches of
  the FIFA World Cup 2022}
\medskip \hrule height 1pt \medskip % use \hsize
\begingroup\offinterlineskip \tabskip=0pt
\noindent\valign{\VI#&&#\cr% periodic preamble
124pt&\Vo{\bf\qquad Round of 16}\vskip6pt&
         \Mt{\bf\NL}\US-3:1-&            \R&
         \Mb{\bf\AR}\AU-2:1-&
      \E&\Mt\JP{\bf\HR}-1:1, 1:1, (1:3)-&\R&
         \Mb{\bf\BR}\KR-4:1-&
      \E&\Mt{\bf\EN}\SN-3:0-&            \R&
         \Mb{\bf\FR}\PL-3:1-&
      \E&\Mt{\bf\MA}\ES-0:0, 0:0, (3:0)-&\R&
         \Mb{\bf\PT}\CH-6:0-\cr
124pt&\Vo{\bf\qquad Quarter-finals}&
```

Table 9: Knockout matches of the FIFA World Cup 2022

Round of 16	**Quarter-finals**	**Semi-finals**	**Final**
Netherlands–USA 3:1			
	Netherlands–**Argentina** 2:2, 2:2, (3:4)		
Argentina–Australia 2:1			
		Argentina–Croatia 3:0	
Japan–**Croatia** 1:1, 1:1, (1:3)			
	Croatia–Brazil 0:0, 1:1, (4:2)		
Brazil–South Korea 4:1			
			Argentina–France 2:2, 3:3, (4:2)
England–Senegal 3:0			
	England–**France** 1:2		
France–Poland 3:1			
		France–Morocco 2:1	
Morocco–Spain 0:0, 0:0, (3:0)			**Third place play-off**
	Morocco–Portugal 1:0		**Croatia**–Morocco 2:1
Portugal–Switzerland 6:0			

Results: First entry for normal time, second for extra time; goals in penalty shootout with parentheses.
Source: `https://en.wikipedia.org/wiki/2022_FIFA_World_Cup_knockout_stage`

```
    \E&\Mt\NL{\bf\AR}-2:2, 2:2, (3:4)-&\RRR&
       \Mb{\bf\HR}\BR-0:0, 1:1, (4:2)-&
  \EEE&\Mt\EN{\bf\FR}-1:2-&            \RRR&
       \Mb{\bf\MA}\PT-1:0-\cr
109pt&\Vo{\bf\qquad Semi-finals}&
  \EEE&\Mt{\bf\AR}\HR-3:0-&\RRR&\RRR&\R&
       \Mb{\bf\FR}\MA-2:1-\cr
109pt&\Vo{\bf\qquad Final}&
  \EEE&\EEE&\E&
       \Mf{\bf\AR}\FR-2:2, 3:3, (4:2)-&
  \EEE&\E&\omit
     \vtop to 28.4pt{% 2 rows plus rule height
      \vskip 14.4pt % +12pt => 2pt white space
      \hbox to \Vcolumnwidth{\strut\bf\hss
        Third place play-off\hss}\vfil}&
       \Mf{\bf\HR}\MA-2:1-\cr}\endgroup
```

There is one more task to do: The source of the data should be stated in a footnote to the table [1, 13.44].

Example 10 continued: A footnote

```
\medskip \hrule height 1pt \medskip\noindent
{\it Results:} First entry for normal time,
second for extra time; goals in penalty shootout
with parentheses.\par\noindent {\it Source:}
{\tt\catcode`\_=12 https://en.wikipedia.org/%
 wiki/2022_FIFA_World_Cup_knockout_stage}
```

8 Other use cases for `\valign`

One macro of plain TEX based on `\halign` is very useful. It's the macro `\ooalign` [4, p. 356], for overprinting characters. A new symbol can be created by this macro if this symbol is just a combination of characters available in the Computer Modern fonts.

Example 11: Using `\ooalign`

```
\def\smiley{{% a group is required for \ooalign
 \mathsurround=0pt %  required with inline math
 \ooalign{\hidewidth \raise.3ex\hbox{% the eyes
    $\cdot\mkern 2mu\cdot$}\hidewidth\cr
  \hidewidth \lower.2ex\hbox{%        the mouth
    $\scriptscriptstyle \smile$}\hidewidth\cr
  \Orb\cr}}}%                          the head
```

Now we can write `\smiley` and get: ☺.

There is no need to implement a replacement of `\ooalign` with `\valign`. But we learned that one of `\halign`'s features can be utilized to overprint characters.

What makes `\valign` unique to turn it into an unusual application? The next example uses the fact that TEX adjusts the row heights (see section 2, no. 6) to move a character vertically; see also [6, p. 36]. In the second application we use the fact that it's easy to stack characters one above the other with the help of `\valign` (see section 2, nos. 5 and 7).

Case 1: Between digits the hyphen is too low as it is placed at a height that looks good for lowercase letters. But with telephone numbers a hyphen is used, thus we need a fix for these numbers.

Example 12: Moving a symbol with `\valign`

```
\def\D-{% center ‘-’ with respect to a digit
 {\everycr={}\tabskip=0pt \valign{##\cr% 1 row
   \vphantom0\cr   % invisible; defines height
   \vfil\hbox{-}\vfil\cr}}}%  centered wrt ‘0’
```

It is easy to raise the hyphen with the above defined `\D-`: The input "`555\D-6789 555-6789`" typesets "555-6789 555-6789". It also works with other font sizes.

Case 2: In former times lowercase German umlauts (ä, ö, ü) were written in texts that were typeset in "Fraktur" (blackletter) sometimes with an 'e' above the vowel instead of using two periods.

Example 13: Stacking symbols with `\valign`

```
\def\e"#1{% #1: a, o, u; put an ‘e’ above it
 {\everycr={}\tabskip=0pt \setbox0=\hbox{#1}%
  \valign{##\tabskip=0.18ex&% distance e--vowel
      ##\tabskip=0pt\cr      % two-row preamble
    \hbox to \wd0{\hss\fivebf e\hss}& % the ‘e’
    \box0\cr}}}%                  output the vowel
```

Now we can *simulate* the old writing with Computer Modern Roman. For example, TEX typesets the input "`Albrecht D\e"urer, Hans Sch\e"aufelin`" as "Albrecht Duͤrer, Hans Schaͤufelin". Note, however, kerning information is lost; for example, vaͤ va.

References

[1] *The Chicago Manual of Style*, 15th edition, Chicago, Illinois: University of Chicago Press, 2003.

[2] Malcolm Clark, *A plain TEX primer*, Oxford University Press, 1992.

[3] Taco Hoekwater, "A use case for `\valign`", *ConTEXt Group Journal* 2018, 7–18. `articles.contextgarden.net/journal/2018/7-18.pdf`

[4] Donald E. Knuth, *The TEXbook*, Volume A of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1984.

[5] Donald E. Knuth, *TEX: The Program*, Volume B of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986.

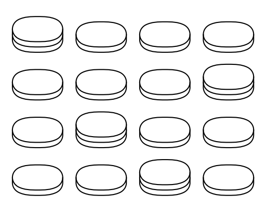
[6] Peter Wilson, "Glisterings: Glyphs, long labels", *TUGboat* **35**:1 (2014), 36–38. `tug.org/TUGboat/tb35-1/tb109glister.pdf`

Appendix

I don't know if Clark was aware of answer 22.∞ in `texbook.tex`, i.e., the source of [4], that applies `\valign`. Knuth's macros `\one` and `\two` are as simple as possible. Sure, `\halign` can be used but only with templates like '`\vbox{\offinterlineskip#}`'. Or we redefine the macros, for example, with a more complex `\ooalign` (see section 8). Such macros can also be used in Knuth's `\valign` answer [4, p. 335].

Example 14: An `\halign` solution of exercise 22.∞

```
\font\manual=manfnt % symbols for The TeXbook
\def\pennytop{% circular symbol like a coin
   \hbox to 24pt{\manual\char'130\hfil}}
\def\pennyedge{% like \pennytop but not closed
   \hbox{\manual\char'133}}
\def\OA#1{% #1: hboxed char on top of \pennyedge
   {\ooalign{\raise2pt#1\cr\pennyedge\cr}}}
\def\one{\OA\pennytop}\def\two{\OA{\hbox{\one}}}
\halign{&\tabskip=0pt#\cr% periodic
   \two&\one&\one&\one\cr \noalign{\vskip6pt}
   \one&\one&\one&\two\cr \noalign{\vskip6pt}
   \one&\two&\one&\one\cr \noalign{\vskip6pt}
   \one&\one&\two&\one\cr}
```



The alignment was created with more effort. As Clark indicated: It can be done the other way too. But as shown here, sometimes one needs more code.

◇ Udo Wermuth
Dietzenbach, Germany
u dot wermuth (at) icloud dot com

JabRef: BibTeX-based literature management software

Oliver Kopp, Carl Christian Snethlage, Christoph Schwentker

Abstract

JabRef is a comprehensive literature management tool built entirely around the BibTeX data format. JabRef's features include web search, import and export in various formats, file attachment management, and quality checks of entries. Moreover, JabRef supports conducting systematic literature reviews. As an open-source initiative, JabRef also provides a suitable platform for software engineering education, particularly for newcomers to the field.

1 Overview on JabRef

JabRef is a comprehensive, open-source, cross-platform reference management tool that offers a rich set of features. The tool has its roots dating back to a merge of Bibkeeper[1] and JBibtexManager[2]. Since its first release on November 30, 2003,[3] JabRef has come a long way in serving researchers and writers. JabRef not only boasts a user-friendly graphical interface,[4] but also provides a command line interface.[5] It empowers users to gather, organize, and share bibliographic references with ease. What sets JabRef apart from other reference management tools is its exclusive use of BibTeX as its internal file format [13]: All entries and metadata are stored inside a single BibTeX file, providing seamless integration with numerous typesetting systems.

Since JabRef 3.1, released on December 24, 2015, JabRef keeps the BibTeX entries unmodified by default. When a user changes an entry, JabRef applies its formatting rules to that entry and only modifies that single entry. All others remain untouched.[6] The ordering of entries in BibTeX files is also preserved. If a defined ordering is desired, a user can opt in to JabRef's feature to sort the file on save. All in all, JabRef can be used seamlessly with other third-party tools for editing or consuming BibTeX files.

[1] `bibkeeper.sourceforge.net/`

[2] `ctan.org/pkg/jbibtexmanager`

[3] `sourceforge.net/p/jabref/news/2003/11/jabref-10-released/`

[4] `docs.jabref.org/getting-started`

[5] `docs.jabref.org/advanced/commandline`

[6] Two exceptions: 1) If "save actions" is activated, JabRef rewrites entries updated by the save actions (Section 4.3). 2) Fields migrated during load. For instance, the update of the storage of JabRef's entry grouping, the upgrade of "markings" to JabRef groups, converting "special fields" from keywords to separate fields, and converting the field `review` to `comment`. The JabRef team is currently investigating the automatic update to provide improved transparency (JabRef#10370).

Additionally, JabRef provides explicit support for BibLaTeX. This is visible to the user at several places. For instance, when adding a new entry, JabRef first asks for the entry type. That dialog is filled with the entry types available in BibTeX and BibLaTeX, respectively, depending on the library's setting. It is always possible to change the library mode between BibTeX and BibLaTeX at a later point. JabRef's functionality to check for non-standard field content is aware of BibLaTeX's field content handling. For instance, when page numbers are connected by a single hyphen, the integrity check warns only when in BibTeX mode.

To help guide readers through JabRef in more detail, the overall user interface of JabRef is presented in Section 2. The internal data model is discussed in Section 3. Subsequently, selected features are outlined in Section 4. Important facts on JabRef's open-source development and especially the onboarding of students are presented in Section 5. Collaboration is a hot topic and the current state of JabRef is discussed in Section 6. Finally, contribution possibilities are outlined in Section 7.

2 The UI of JabRef

Figure 1 shows the main window of JabRef.

The *search* bar ❶ in the toolbar enables searching in the library. Besides free text search, the search supports search in selected fields as well as a regular-expression based search. It is also possible to search in the attached PDFs.[7]

The *web search* ❷ side pane allows for searching for bibliographic entries out of more than 20 catalogs. For a concrete search, the user inputs a search query as a combination of keywords using the syntax of Apache Lucene[8]. This query is transformed to the syntax of the respective search catalog and subsequently used to query the corresponding catalog API. The returned entries are displayed as a list of bibliographic references, for some catalogs even with abstract. The user can now select those references to be imported into the library. All in all, the use of a general query syntax enables switching online resources transparently.[9]

The *groups* ❸ side pane allows for grouping references into hierarchical collections based on keywords, search terms, or manual assignment.[10]

[7] `docs.jabref.org/finding-sorting-and-cleaning-entries/search`

[8] `lucene.apache.org/core/2_9_4/queryparsersyntax.html`

[9] `docs.jabref.org/collect/import-using-online-bibliographic-database`

[10] `docs.jabref.org/finding-sorting-and-cleaning-entries/groups`

doi.org/10.47397/tb/44-3/tb138kopp-jabref

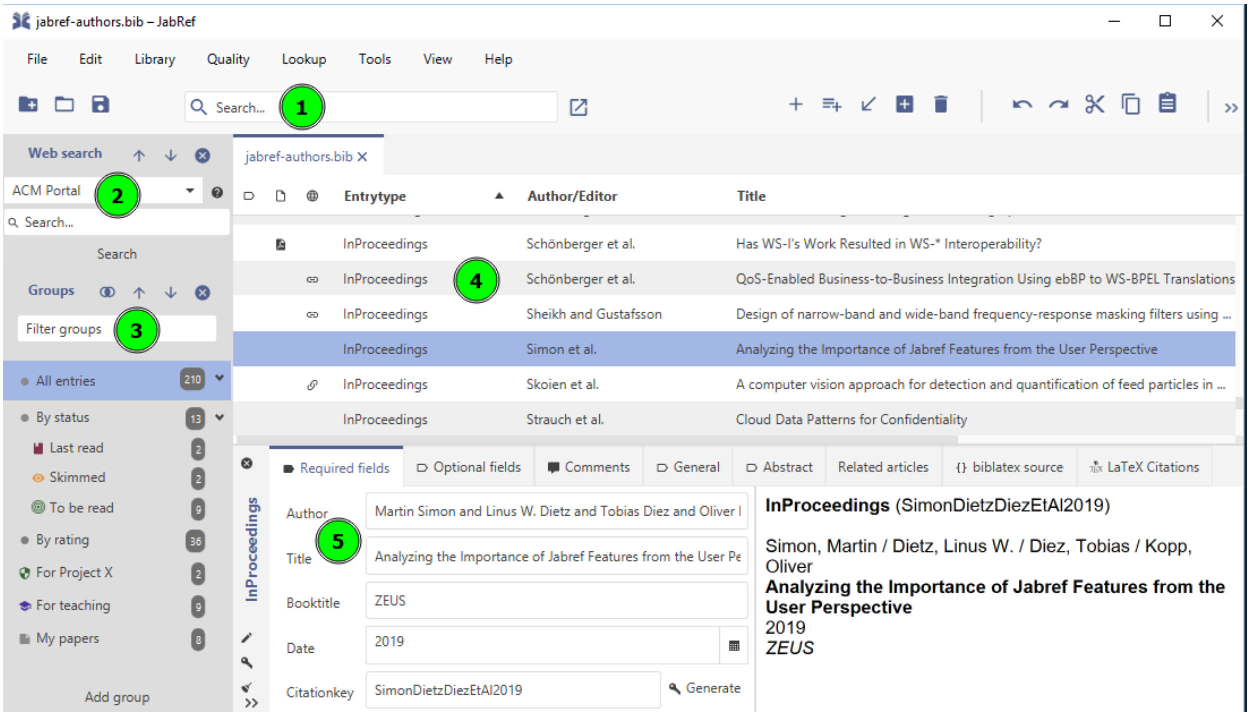


Figure 1: JabRef's main window

All entries are displayed in the *entry table* ④. The column headers represent field names. The contents of the table are an interpreted version of the field content: The displayed version is text that looks approximately as it would be output by BibTeX: Basic LaTeX formatting commands are interpreted and converted to Unicode to provide a better preview.

The *entry editor* ⑤ categorizes the BibTeX fields and provides editing capabilities for field values. The categorization is rendered as tabs in the entry editor. The distinction between required and optional fields is based on the field definition in the common BibTeX styles [11]. JabRef adds additional categories to support unique JabRef features.

The "Comments" tab lists comments to the entry. Starting with JabRef 5.10, user-specific comments are supported: If a BibTeX field name follows the format `comment-X`, it will be treated as a comment by user `X`.

The "General" tab groups general metadata of the entry. Examples are its DOI, keywords, groups of the entry, and a rating (1 to 5 stars) of the entry. Finally, attached files are listed here.

When PDFs are associated with an entry (using the `file` field), the "Annotations" tab displays highlighted text and PDF comments in a list format. Clicking on a particular item opens the PDF viewer at the respective location.

The "Abstract" tab is used to display the abstract field of the associated entry.

Occasionally, a BibTeX entry may contain fields that are not standard fields or recognized by JabRef. In such instances, these unrecognized fields are gathered and displayed under the "Other fields" tab.

The entry editor also offers literature recommendations, based on the currently selected article. Recommendations are provided by the Mr. DLib service [3], from a corpus with several million open access publications.[11] Mr. DLib uses several recommendation algorithms, including content-based filtering and collaborative filtering. The results are displayed inside the "Related articles" tab.

The "BibTeX source" tab displays the raw BibTeX code, which users can modify directly. Once the editing process is complete, JabRef parses the updated content and automatically populates the corresponding fields in the entry editor.

The "LaTeX citations" tab establishes a link between a LaTeX document and the citations used. JabRef recursively searches the directory of the BibTeX file for `*.tex` files and checks for `\cite` commands being included. If the given key matches the currently selected entry, JabRef displays the name of the `tex` file, with line and column number.

[11] `mr-dlib.org/`

The entry editor also provides a complete rendering of the entry. The layout can be configured either by a templating language custom to JabRef or by choosing a style from the Citation Style Language [19].[12] The custom template language utilized in JabRef comes with the significant advantage of being able to render fields that are not necessarily relevant to citations, such abstracts, comments, or other user-defined fields.

3 BibTeX-based internal data model

Since its inception, JabRef is implemented in Java and JabRef's internal data model is based directly on BibTeX.

On the technical side, the contents of a BibTeX entry is implemented as a map that links a `Field` to a `String`. A `Field` is a Java interface abstracting from concrete fields.[13] For instance, one concrete field class is `org.jabref.model.entry.field.StandardField`, which models all fields available in standard BibTeX and BibLaTeX. An example is `AUTHOR("author", FieldProperty.PERSON_NAMES)`. Based on the field property, the JabRef entry editor knows that the value editing component for that field should support person names. This is especially important for autocompletion. Besides `StandardField`s, there are `IEEEField`, `BiblatexSoftwareField`, and more. If JabRef does not recognize a field, it is stored in `UnknownField`, making JabRef extensible to unknown fields.

The same concept is implemented for entry types: The Java interface is `EntryType` with possible implementations `StandardEntryType`, `IEEETranEntryType`, `BiblatexSoftwareEntryType`, and `UnknownEntryType`.

4 Selected features

This section explains selected features of JabRef: Importing bibliographic data from files (Section 4.1) or the web (Section 4.2), quality assurance of entries (Section 4.3), and JabRef's feature for supporting systematic literature reviews (Section 4.4).

4.1 Importing MODS, EndNote, Citavi, and PDFs

JabRef can import other formats, such as MODS,[14] EndNote, and Citavi.

JabRef also offers the possibility to retrieve BibTeX data from PDF files, in four different ways:

1. reading embedded BibTeX data (e.g., generated by the `authorarchive` package [2]);
2. reading BibTeX text on the first page (e.g., generated by the `coverpage` package [8]);
3. interpreting the first PDF page and applying heuristics for the correct BibTeX data; and
4. using a self-hosted instance of the GROBID machine learning library [4].

In each case, the user drops a PDF between list entries in the main table, JabRef uses all these possibilities and merges the result using heuristics to retrieve the most complete BibTeX entry for the given PDF file. JabRef can also import a whole directory of PDF files and complete the current BibTeX library with the PDF files missing in that library.

4.2 Importing references using a web browser

JabRef provides a browser plugin that allows importing the reference represented by a website into JabRef.[15] An icon is displayed in the browser address bar. As soon as that icon is clicked, the plugin fetches the bibliographic data and sends it to JabRef. Then, JabRef presents an import inspection window, which allows for double-checking of the generated citation data and an assignment to one of the opened libraries.

4.3 Quality assurance of entries

JabRef offers three features to ensure the quality of entries: The integrity check,[16] the cleanup of entries,[17] and the automatic cleanup upon save.[18]

The *integrity check* checks the current BibTeX library for issues such as duplicate BibTeX keys, whether pages are connected via two hyphens in the case of BibTeX mode, and whether the field value contains HTML-encoded characters. Any issues are reported as a list in a separate window. When clicking on an issue, the respective entry is selected in JabRef's main table.

For some BibTeX issues, a cleanup is available: The user can start a cleanup dialog (Figure 2) and select actions to improve the quality of the bibliographic entries. For instance, some BibTeX entries have the DOI contained in the `note` field. JabRef

[12] `docs.jabref.org/setup/preview#layouts-styles`
[13] `github.com/JabRef/jabref/blob/refs/heads/main/src/main/java/org/jabref/model/entry/field/Field.java#L8`
[14] Metadata Object Description Schema by the Library of Congress: `www.loc.gov/standards/mods/`
[15] `docs.jabref.org/collect/jabref-browser-extension`
[16] `docs.jabref.org/finding-sorting-and-cleaning-entries/checkintegrity`
[17] `docs.jabref.org/finding-sorting-and-cleaning-entries/cleanupentries`
[18] `docs.jabref.org/finding-sorting-and-cleaning-entries/saveactions`

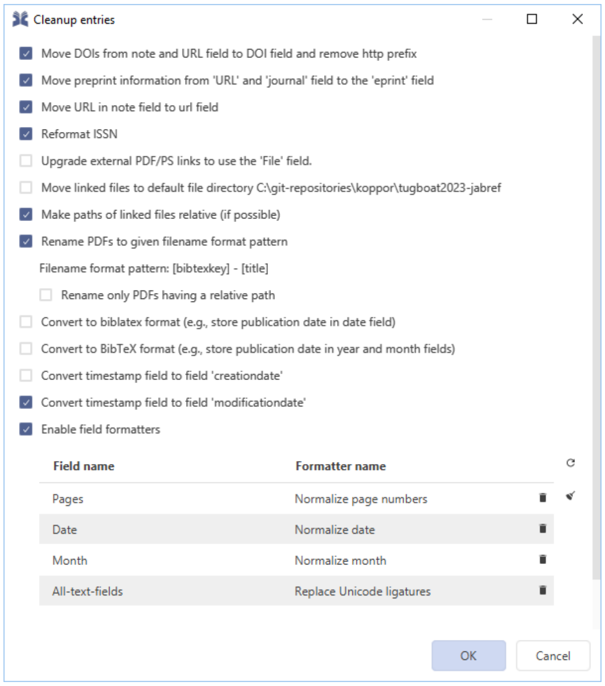


Figure 2: Options to clean up entries

can move the DOI from there to the `doi` field. Other cleanup actions include the normalization of page ranges and the normalization of name lists, which ensures that all names are formatted in the form "lastname, firstname".

Finally, the cleanup actions can also be triggered on save ("save actions"). For instance, the user can choose to always normalize the list of authors.

4.4 Systematic literature reviews

Systematic literature reviews (SLRs) are comprehensive examinations of existing research on a particular topic, conducted according to a well-defined and transparent methodology [7]. The most difficult challenge researchers face when conducting an SLR is during the search step [1, 16]. Commonly used e-libraries in the domain of computer science research, such as IEEE, arXiv, and ACM, do not support easy bulk access, which is key to the SLR method [1]. JabRef fills this gap with its special SLR feature [16]. The main concept of the implementation is to a) build on git and BibTeX to track the state of papers (excluded, included, ...); and b) transform a generic query syntax to the different query formats of each publisher.

JabRef provides an SLRwizard guiding the user through the creation of an SLR [16]. First, the user is asked for the intention of the SLR and in a next step JabRef transforms the queries and sends them to each catalog. The resulting BibTeX data is stored

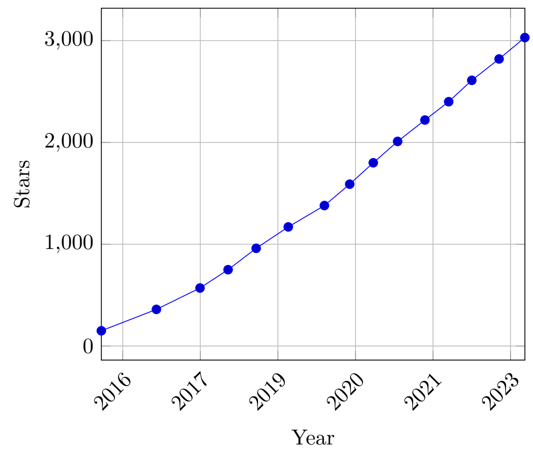


Figure 3: JabRef's stars on GitHub

in one BibTeX file per catalog. The user then reviews the fetched entries and removes entries if they meet the exclusion criteria. This is recorded as a git commit. When fetching is re-triggered, the deletions are honored and only "real" new data is added to the SLR repository.

5 JabRef as an open-source project

JabRef has a long history as an open-source project. The first commit was on October 14, 2003. It moved to GitHub in 2015 and gained popularity (Figure 3). For the last 60 months, JabRef shows the following statistics:[19]

- 6 new committers per month on average,
- 16 total active committers per month on average,
- 37 percent of committers were new committers,
- 95 commits per month on average,
- 5.94 commits per committer per month on average, and
- Cumulative total committers grew by 402 (from 261 to 663).

The constantly increasing number of new contributors is most likely due to the use of JabRef in software engineering education in universities.[20] JabRef positions itself as a project that is more complex than programming exercises for beginners, but also less complex than distributed systems. This allows students to focus on the software engineering aspects of a real-world program. Examples of such aspects include: proper requirements specification and a proper design document, and also proper test cases

[19] The committer statistics have been gathered by git-pulse, available at `github.com/git-pulse/tools`.

[20] A list of courses is provided at `devdocs.jabref.org/teaching`.

Title		Size	Main Focus	Issue Understanding Effort	Implementation Effort	Testing Effort	Status
Integrate check-bib-for...	#348	small	logic + UI	low	medium	low	Free to take
Add support for scite.ai	#375	small	logic + UI	low	high	medium	Free to take
Move data to wiki data	#49	small	logic	medium	low	low	Free to take

Figure 4: Issue board for selecting an issue to work on

and the process of using GitHub pull requests to contribute to an open-source project. In some courses, the schedule to contribute follows coordinated lectures, ensuring sustainability of the concepts taught (cf. [9]). We also try to make each project a success, which should also contribute to sustainability (cf. [17]).

One major issue for newcomers is to find suitable tasks [12]. In JabRef, we have tagged "good first issues" [6] for a long time. The intention of these issues was to provide the contributor with a self-guided tour through the concept of JabRef to be able to contribute more in a next issue. Since students participating in a course are primarily motivated to complete their assignment and then move on to other assignments, the required effort to understand JabRef was not usually made.

In 2022, we changed this approach. We moved away from recommending "good first issues" and turned to a more fine-grained categorization schema to present issues (Figure 4): We introduced a project board, where each issue is categorized by the size of the project, the main focus, the issue understanding effort, the implementation effort, the testing effort, and the status of the issue.

The size of a project is either small, medium, or large. We have seen that some students tend to like to work on the UI, while others prefer to do backend-like development. Thus, we introduced "main focus", which may be on one (or both) of the two main components of JabRef: The logic and the user interface.

Moreover, students tend to underestimate the task; in particular, they assume that they are able to understand a task in a very short time, implement it, and can get by with simple tests (if at all). Therefore, we aim to make these efforts more explicit.

The *issue understanding effort* captures the estimated effort to understand the context and the problem described in the issue. Although one might think that literature management is a simple task, there are details to think about. One example is adding PDF files that contain multiple papers.[21] Here, for instance, students need to understand that extracting BibTeX data from such a PDF is different from PDFs containing only one paper — and that this leads to significant code changes in JabRef.

The *implementation effort* captures the estimated effort to implement the main functionality. In general, effort estimates are not exact [10]. We merely view this category as a coarse-grained guideline for selecting an issue. For instance, an improved error message when downloading a file is quite different from developing and implementing a new concept for the entry editor in JabRef.

The *testing effort* is highlighted to stress that JabRef requires more than merely implementing a feature; it necessitates a comprehensive approach in line with the broader scope of software engineering, which goes far beyond writing code [18]. Furthermore, the inclusion of tests alongside new features is crucial for us, as it enables regression testing when subsequent changes may affect existing behavior.

Regarding other approaches, Vargovich et al. [14] take a different approach: they attempt to automatically label issues based on the domain of the APIs involved in the solution (e.g., User Interface, Test, Databases). Our manual approach distinguishes between "logic" and "UI" as we consider these domains to be the key distinguishing components (similar to backend and frontend development). We also distinguish according to the general size of the project and the difficulty of understanding an issue.

From a resource perspective, JabRef's software engineering education by JabRef is backed by the non-profit organization JabRef e.V., which provides the legal framework, the resources, and the network.

6 Collaboration

One of the most requested features is online collaborative work. Currently, JabRef offers collaboration based on the BibTeX file, because JabRef can detect if the file changed on disk. Thus, there

[21] `github.com/JabRef/jabref/issues/8128`

are currently two ways to collaborate directly:[22] 1) using version control systems such as `git`; 2) using file-synchronization services such as Dropbox or OneDrive. Both approaches require advanced users: regarding version control systems, not all LaTeX users use them [15]; and regarding file-synchronization services, each external update leads to a notification, which needs to be handled.

JabRef also offers collaboration using an SQL connection to a PostgreSQL database.[23] The changes are synchronized instantly to the server and then distributed to the other connected JabRef instances. It is possible to automatically write the current state to a local file, which can then be consumed by BibTeX tools. Additionally, JabRef offers connecting to a MySQL database. However, live synchronization is not supported there and the user has to push and pull changes manually.

Generally, there are several online services for collaboration with BibTeX: CiteDrive[24] provides a dedicated online service for BibTeX users, Overleaf [5] is the leading LaTeX online collaboration tool, and JabRef developers are also working on a web-based online collaboration platform.[25] In parallel, there are efforts to integrate JabRef with CiteDrive. Once accomplished, users will be able to utilize CiteDrive for BibTeX collaboration and take advantage of its Overleaf integration,[26] while also using JabRef for more advanced BibTeX management.

7 Contribution possibilities

JabRef offers multiple points of access for possible contributions. Besides direct code contributions, JabRef calls for improvements of the user documentation, which is currently maintained using Markdown in a repository on GitHub.[27] Lastly, there is a call for contributions page[28] describing even more possibilities to contribute.

Acknowledgements Our sincere gratitude goes to the users who have selected JabRef for their university work, demonstrating trust and confidence in our tool and our team. We extend heartfelt thanks to the dedicated volunteers who tirelessly support JabRef's users and code contributors. Their commitment to providing assistance in our online forum and meticulous review of pull requests is vital to our community. We also would like to thank all the users who constantly test the latest versions and give valuable feedback on the functionality. Lastly, we wish to express profound appreciation to all the Ph. D. students and postdocs who invest precious non-research time into JabRef. Their contributions enrich our project and reflect a commendable spirit of collaboration and innovation.

[22] `docs.jabref.org/collaborative-work/sharedbibfile`
[23] `docs.jabref.org/collaborative-work/sqldatabase`
[24] `www.citedrive.com/`
[25] `github.com/jabref/jabrefonline`
[26] `bibtex.eu/overleaf-citedrive/`
[27] `github.com/JabRef/user-documentation`
[28] `contribute.jabref.org/`

References

[1] A. Al-Zubidy, J.C. Carver. Identification and prioritization of SLR search tool requirements: an SLR and a survey. *Empirical Software Engineering*, 24(1):139–169, May 2018. `doi.org/10.1007/s10664-018-9626-5`

[2] A.D. Brucker. *The authorarchive package*, Feb. 2023. `ctan.org/pkg/authorarchive`

[3] S. Feyer, S. Siebert, et al. Integration of the Scientific Recommender System Mr. DLib into the Reference Manager JabRef. In *Advances in Information Retrieval*, Lecture Notes in Computer Science. Springer International Publishing, 2017. `doi.org/10.1007/978-3-319-56608-5_80`

[4] GROBID. `github.com/kermitt2/grobid`, 2008–2023.

[5] T. Hejda. TeX Live and Overleaf revisited. *TUGboat* 44(2), 2023. `doi.org/10.47397/tb/44-2/tb137hejda-overleaf-tl`

[6] H. Horiguchi, I. Omori, M. Ohira. Onboarding to open source projects with good first issues: A preliminary analysis. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021. `doi.org/10.1109/saner50967.2021.00054`

[7] B. Kitchenham, S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University, 2007.

[8] O. Kopp, M. Mühlich. *The coverpage package*, 2006. `ctan.org/pkg/coverpage`

[9] J. Ludewig. Erfahrungen bei der Lehre des Software Engineering. In *Software Engineering im Unterricht der Hochschulen, SEUH 11*. dpunkt, 2009.

[10] P. Matsubara, I. Steinmacher, et al. Trust yourself! Or maybe not: factors related to overconfidence and uncertainty assessments of software effort estimates. In *Brazilian Symposium on Software Engineering*. ACM, 2021. `doi.org/10.1145/3474624.3474643`

[11] F. Mittelbach, U. Fischer. *The LaTeX Companion*, ch. Bibliography Generation, pp. 375–468. Addison-Wesley, third ed., 2023.

[12] I. Steinmacher, C. Treude, M.A. Gerosa. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software*, 36(4):41–49, July 2019. `doi.org/10.1109/ms.2018.110162131`

[13] Technical University of Munich. Reference management software comparison – 9th update, 2022. `doi.org/10.14459/9.COMPARISON_REFERENCE_MANAGEMENT_2022`

[14] J. Vargovich, F. Santos, et al. GiveMeLabeledIssues: An open source issue recommendation system. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 2023. `doi.org/10.1109/msr59073.2023.00061`

[15] B. Veytsman. Using Overleaf for collaborative projects: First impressions and lessons learned. *TUGboat* 41(2):179–181, 2020. `tug.org/TUGboat/tb41-2/tb128veytsman-overleaf.pdf`

[16] D. Voigt, O. Kopp, K. Wild. Systematic literature tools: Are we there yet? In *ZEUS*, vol. 2839 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.

[17] M. Wetzel, H. Röder. Studentische Projekte: erfolgreich oder realistisch? In *Software Engineering im Unterricht der Hochschulen, SEUH 11*, pp. 17–28. dpunkt, 2009.

[18] T. Winters, T. Manshreck, H. Wright. *Software Engineering at Google: Lessons Learned from Programming Over Time*. O'Reilly Media, 2020.

[19] R.M. Zelle, B.M. Wiernik, et al. CSL 1.0.2 specification, 2015. `docs.citationstyles.org/en/stable/specification.html`

◇ Oliver Kopp
Sindelfingen, Germany
`https://github.com/koppor`
ORCID 0000-0001-6962-4290

◇ Carl Christian Snethlage
Zwiesel, Germany
`https://github.com/calixtus`

◇ Christoph Schwentker
Paderborn, Germany
`https://github.com/Siedlerchr`

The Treasure Chest

These are the new packages posted to CTAN (`ctan.org`) from April–October 2023. Descriptions are based on the announcements and edited for extreme brevity.

Entries are listed alphabetically within CTAN directories. More information about any package can be found at `ctan.org/pkg/`*pkgname*.

◇ Karl Berry
`https://tug.org/TUGboat/Chest`
`https://ctan.org/topic`

biblio

`biber-linux-aarch64` in `biblio/biber`
Biber binaries for the `linux-aarch64` platform.

fonts

`arsenal` in `fonts`
Semi-grotesque award-winning font Arsenal.

`culmus` in `fonts`
Hebrew fonts from the Culmus project.

`inconsolata-nerd-font` in `fonts`
Support for the Inconsolata Nerd Font.

`khatalmaqala` in `fonts`
An Arabic font especially for scientific texts.

`metsymb` in `fonts`
Provide (vector) meteorological symbols.

`parsimatnparsimatn` in `fonts`
Contemporary Persian font for scientific and formal writing.

`parsinevis` in `fonts`
Adaptation of SIL Scheherazade New for Persian typesetting and scientific writing.

`rit-fonts` in `fonts`
Malayalam font family from the Rachana Institute of Typography.

graphics/metapost/contrib/macros

`huffman` in `graphics/metapost/c/huffman`
Draw binary Huffman trees; based on `metaobj`.

`wordcloud` in `graphics/metapost/c/wordcloud`
Draw wordclouds with MetaPost and Lua.

graphics/pgf/contrib

`circularglyphs` in `graphics/pgf/contrib`
Circular glyph alphabet.

`papiergurvan` in `graphics/pgf/contrib`
Commands to work with Gurvan paper sheets.

doi.org/10.47397/tb/44-3/tb138chest

`playcards` in `graphics/pgf/contrib`
TikZ template for drawing playing cards.

`profsio` in `graphics/pgf/contrib`
Work with French BTS SIO math concepts.

`quiver` in `graphics/pgf/contrib`
Draw commutative diagrams exported from `q.uiver.app`.

`rouequestions` in `graphics/pgf/contrib`
Draw a “question wheel” with answers inside.

`string-diagrams` in `graphics/pgf/contrib`
Create string diagrams.

`tikz-osci` in `graphics/pgf/contrib`
Draw oscilloscope screen captures.

`tikz2d-fr` in `graphics/pgf/contrib`
Work with some 2D TikZ commands.

`tikz3d-fr` in `graphics/pgf/contrib`
Work with some 3D figures.

`tilings` in `graphics/pgf/contrib`
Tiles and tilings; updates and extends `penrose`.

`tkz-bernoulli` in `graphics/pgf/contrib`
Draw Bernoulli trees with TikZ.

`tkz-elements` in `graphics/pgf/contrib`
Lua library for drawing Euclidean geometry.

`wordle` in `graphics/pgf/contrib`
Create wordle grids.

---

info

`antique-spanish-units` in `info/spanish`
Short description of units used in Spain et al. from the 16th to 19th centuries.

---

macros/generic

`advice` in `macros/generic`
Extend commands and environments.

`collargs` in `macros/generic`
Collect arguments of any command.

`etoolbox-generic` in `macros/generic`
Loader for `etoolbox` in non-LaTeX formats.

`memoize` in `macros/generic`
Externalization of graphics and memoization of compilation results in general; supports all major engines and formats.

---

macros/latex/contrib

`beautybook` in `macros/latex/contrib`
Template for math and science books, with both Chinese and English versions.

`cahierprof` in `macros/latex/contrib`
Schedule and grade books for French teachers.

`codedescribe` in `macros/latex/contrib`
Describe document- and class-level commands.

`couleurs-fr` in `macros/latex/contrib`
French color names.

`counterz` in `macros/latex/contrib`
Stealth prefixes, and random counters.

`creationboites` in `macros/latex/contrib`
Creating and customizing `tcolorbox` boxes.

`curriculum-vitae` in `macros/latex/contrib`
Lightweight class for CVs.

`customenvs` in `macros/latex/contrib`
Multiple choice, list with chosen items, other custom environments.

`defoldfonts` in `macros/latex/contrib`
Define old font commands.

`examz` in `macros/latex/contrib`
Randomized exams with multiple versions.

`expex-glossonly` in `macros/latex/contrib`
Help `gb4e` and `linguex` users use the `expex` glossing macros.

`fail-fast` in `macros/latex/contrib`
Turn all warnings into errors.

`fitch` in `macros/latex/contrib`
LaTeX macros for Fitch-style natural deduction.

`floatbytocbasic` in `macros/latex/contrib`
Enhancements for `float` from `tocbasic`.

`floatrowbytocbasic` in `macros/latex/contrib`
Enhancements for `floatrow` from `tocbasic`.

`glossaries-norsk` in `macros/latex/contrib`
Norsk Bokmål language module for `glossaries`.

`hebrew-fonts` in `macros/latex/contrib`
Input encodings, font encodings and font definition files for Hebrew.

`hep-graphic` in `macros/latex/contrib`
Extensions for graphics, plots and Feynman graphs in high energy physics.

`highlightx` in `macros/latex/contrib`
Handwriting effects in formulas or paragraphs.

`homework` in `macros/latex/contrib`
Class for homework, with multilingual support.

`isosafety` in `macros/latex/contrib`
ISO signs and colors according to standards 7010 and 3864.

`isphysicalmath` in `macros/latex/contrib`
Write math formulas following scientific notation.

`iwonamath` in `macros/latex/contrib`
LaTeX support for scaled Iwona math fonts.

`joinbox` in `macros/latex/contrib`
Join boxes vertically or horizontally.

`jourr` in `macros/latex/contrib`
Template for journal rebuttal letters.

`logoetalab` in `macros/latex/contrib`
Insert the “Licence Ouverte Etalab 2.0” logo.

`macros/latex/contrib/codedescribe`

`mitthesis` in `macros/latex/contrib`
LaTeX template for MIT theses.

`movement-arrows` in `macros/latex/contrib`
Draw movement arrows on linguistic example sentences.

`newfloat` in `macros/latex/contrib`
Define new floating environments.

`non-decimal-units` in `macros/latex/contrib`
Support for historical non-decimal units.

`oststud` in `macros/latex/contrib`
Templates for the student organization at OST FH, Switzerland.

`panneauxroute` in `macros/latex/contrib`
Display French road signs (vector graphics).

`postit` in `macros/latex/contrib`
Display Post-it notes.

`profmaquette` in `macros/latex/contrib`
Use exercises in various documents, for French math teachers.

`ptlatexcommands` in `macros/latex/contrib`
LaTeX commands in Portuguese.

`quizztex` in `macros/latex/contrib`
Create quizzes as in TV shows.

`robust-externalize` in `macros/latex/contrib`
Cache anything in a robust, efficient, and pure way.

`rorlink` in `macros/latex/contrib`
Create ROR symbols linking to their given ROR-IDs.

`setspaceenhanced` in `macros/latex/contrib`
Enhancement of the `setspace` package.

`starray` in `macros/latex/contrib`
Structured arrays of properties, using `expl3`.

`stellenbosch-2` in `macros/latex/contrib`
Stellenbosch University thesis bundle.

`ucph-revy` in `macros/latex/contrib`
Typeset lyrics and dialogue for theatrical productions.

`unicode-math-input` in `macros/latex/contrib`
Allow entering Unicode math symbols without modifying the output encoding.

`unifront` in `macros/latex/contrib`
Give a unique front page to each chapter and section of notes.

`updatemarks` in `macros/latex/contrib`
Extract and update marks from boxes.

m/l/c/beamer-contrib/themes

`beamertheme-rainbow` in `m/l/c/b-c/themes`
Alternate theme colors between frames.

macros/luatex/generic

`addtoluatexpath` in `macros/luatex/generic`
Add to search paths for Lua packages and TeX input files.

`opbible` in `macros/luatex/generic`
Create a study Bible with OpTeX.

`penlightplus` in `macros/luatex/generic`
Additions to the Penlight Lua libraries.

macros/luatex/latex

`luahttp` in `macros/luatex/latex`
Insert interactive web features into PDF output from LuaLaTeX.

`luanumint` in `macros/luatex/latex`
Numerical integration using Lua, inside LaTeX documents.

`luaplot` in `macros/luatex/latex`
Plotting graphs using Lua and either MetaPost or Ti*k*Z.

`sympycalc` in `macros/luatex/latex`
Work with SymPy and PyLuaTeX.

macros/plain

`mlawriter` in `macros/plain/contrib`
Write MLA style documents in plain TeX.

macros/unicodetex/latex

`adobeornaments` in `macros/unicodetex/latex`
Using ornaments from Adobe fonts.

support

`easydtx` in `support`
DTX variant eliminating the need for `macrocode` environments.

`runtexshebang` in `support`
Running TeX documents with a shebang (`#!`).

`wmf2epsc` in `support`
Windows metafile conversion.

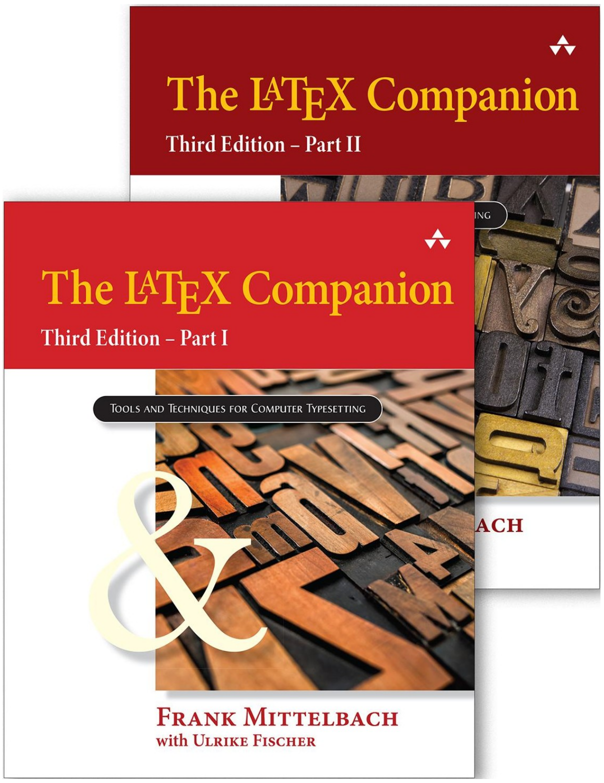
web

`javascripthttp` in `web`
Insert buttons and text fields into PDF output from LaTeX.

Book review: *The LaTeX Companion*, third edition, by Frank Mittelbach with Ulrike Fischer (PDF edition)

John D Lamb

Frank Mittelbach with Ulrike Fischer, *The LaTeX Companion*, 3rd edition. Addison-Wesley, 2023, two volumes, 944 pp. and 970 pp., electronic (PDF) edition, ISBN 978-0-13-465894-0 and 978-0-201-36300-5. tug.org/l/tlc3



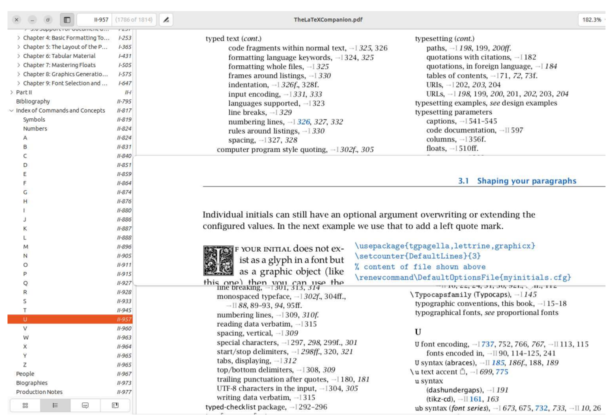
This is a review of the electronic (PDF) edition of *The LaTeX Companion*, third edition (TLC3) [2]. If you are looking for a general discussion of what's new in the third edition, see the earlier *TUGboat* review [3].

The first thing to say about this edition is that it is in a downloadable PDF format. It isn't currently available in any e-reader format and would likely be difficult to adapt for e-readers, since it has two colours (black and blue) and is a reference book you search in rather than read from cover to cover.

There are two reasons you might want the PDF edition instead of (or as well as) the hardback edition. First, at 3.7 kg, the hardback edition is only a little more portable than a saxophone, while the PDF weighs in at only 131.5 MB, small enough to comfortably fit on a mobile phone or tablet. Second, it is quicker to search through a PDF document or navigate from the index to a page in the PDF format.

I tested how easy it is to use TLC (PDF edition) on several devices using several PDF readers. Most mobile phone screens will be too small, but the book is easily readable on a 25 cm (10 inch) tablet screen. The main issue with the tablet was that searching the PDF takes longer, though it is straightforward to use the table of contents or index to navigate to what you want to read about.

TLC3 has a comprehensive index, because it is designed to be used as a reference text (and not just because LaTeX makes this easy!). It also has a lot of cross references. The PDF format makes it particularly easy to handle these, though you may want to choose a PDF reader that allows you to move back to the previous page in case the index entry you clicked on wasn't the one you were looking for. I tested several PDF readers including Adobe Acrobat Reader, Qoppa PDFStudio, Evince and the built-in PDF readers in Chrome and Firefox. All of them worked well. The web browsers have a back button and Evince (see picture below; cropped a little) shows a preview of a hyperlink target if you hover over it before clicking it. All of them also allow you to search for a phrase within the book, either with a button or with Ctrl+F. This is very useful for the occasions when the index is not enough.



One advantage of being able to download the PDF edition of TLC3 is that you can add bookmarks and annotations. For example, I find it difficult to remember how to identify font features in OpenType fonts in LuaLaTeX. TLC3 explains it on page I.715 with an example:

```
otfinfo -f `kpsewhich XCharter-Roman.otf`
```

This is difficult to remember and only works for fonts that `kpsewhich` knows how to find. So, I add a bookmark to the PDF file. Then I put in an annotation explaining what to do if the OpenType file is not one that `kpsewhich` will find. Finally I save the PDF file so that in addition to the table of contents there is

John D Lamb
doi.org/10.47397/tb/44-3/tb138reviews-mittelbach-tlc-pdf

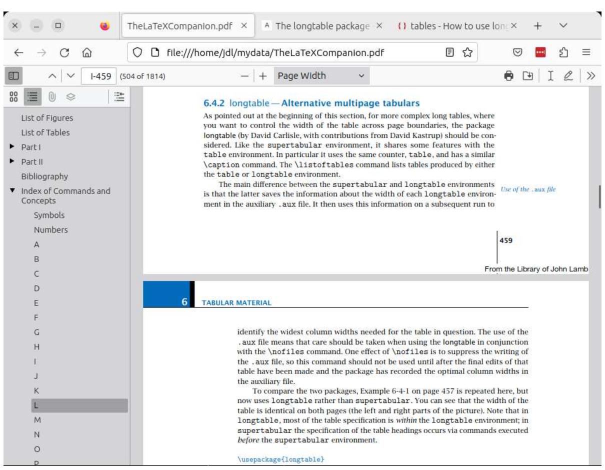
now a list of bookmarks. I can now quickly find this detail just as if I had added some bookmarks to the hardback version of the book, but not quite as messy. The one limitation I found is that bookmarks created in one viewer are not always visible in another.

TLC3 cannot cover every detail of every LaTeX package. There are vast web resources such as [4, 6, 7] so that we can find an answer to most questions we have about LaTeX using a simple search. So it is reasonable to ask whether we really need another electronic resource.

The practical answer is yes. In principle, all the LaTeX packages on CTAN [1] are documented. But even if the documentation is beautifully written it is often cluttered with explanation of how it works, changes from earlier versions and features you are unlikely to need. And, while CTAN is comprehensive, it offers little in discussion of which packages you should use.

By contrast, a web search is often more helpful. Typically it will lead you to general documentation sites like [6] and [7] or to sites answering specific questions like StackExchange [4]. These are often very good — LaTeX package writers post answers on such sites. But the answers are not always comprehensive, accurate or up-to-date.

TLC3 overcomes many of the limitations of web resources. The authors know LaTeX inside out and so their advice is invariably good and usually up-to-date. They discuss alternative methods of achieving what you want and point out limitations in various packages. And their explanations are usually longer and better illustrated than anything you will find online.



In practice, then, you will probably want to use TLC3 (electronic edition) in combination with online resources. The picture above shows how you might do this in Firefox, with separate tabs for TLC3 and the results of various web searches. For example, suppose you wish to create a table that is too long to fit on a single page. A web search quickly identifies the `longtable` package and gives some simple examples of its use. Looking through the index (or table of contents) of TLC3 you find more detailed documentation on `longtable`. In particular, it explains that it uses the `table` counter and shares some behaviour with the `table` environment. But you also find discussion of alternatives such as `supertabular` and `xltabular` that might turn out to be better for your needs.

The electronic edition of TLC3 is easier for quick reference than the print edition [2]. But the print edition is easier to read for those learning LaTeX or when you need to read more extensively than is easy on a screen.

There are a few limitations of the electronic edition. The index entries take you to the top of the page indicated, and not to a specific point on the page where that would be more useful, though the table of contents entries work better. It would be nice if there were links from footnote text back to the footnote and not just the other way round. And the PDF file contains no tags, which limits its accessibility. These are current limitations of using LaTeX to write a book that could only be written in LaTeX. And there is much ongoing (though still experimental) work in this area [5], so that we may hope that many of these limitations are resolved by the time the fourth edition of *The LaTeX Companion* is published.

References

[1] CTAN, *Comprehensive TeX Archive Network*. `ctan.org`

[2] Frank Mittelbach with Ulrike Fischer, *The LaTeX Companion*, 3rd edition (parts I and II), Addison-Wesley, 2023. `tug.org/l/tlc3`

[3] John D Lamb, *Book review: The LaTeX Companion, third edition, by Frank Mittelbach with Ulrike Fischer*, *TUGboat*, Vol. 44, No. 2, 2023. `tug.org/TUGboat/tb44-2/tb137reviews-mittelbach-tlc.pdf`

[4] StackExchange, *TeX - LaTeX*. `tex.stackexchange.com`

[5] Ulrike Fischer, *The tagpdf package*. `github.com/u-fischer/tagpdf` `ctan.org/pkg/tagpdf`

[6] Overleaf, *Documentation*. `overleaf.com/learn`

[7] Wikibooks, *LaTeX*. `en.wikibooks.org/wiki/LaTeX`

◇ John D Lamb
`j.d.lamb (at) johndlamb dot net`

Book review: *LaTeX Graphics with TikZ* by Stefan Kottwitz

samcarter

Stefan Kottwitz, *LaTeX Graphics with TikZ*. Packt Publishing, 2023, 304 pp., softcover, US$39.99 (print+ebook; other options available), ISBN 1804618233. packtpub.com/product/latex-graphics-with-tikz/9781804618233

The book *LaTeX Graphics with TikZ* by Stefan Kottwitz [1] comprises 285 pages featuring numerous colourful illustrations and examples. The printed version includes guidelines on obtaining a PDF version at no additional charge.

The sources of all the examples presented in the book are available for download from the publisher's GitHub repository.[1] This conveniently saves the reader from either retyping the code from the printed book or having to copy from the PDF. In cases where the book only contains a short code fragment, the examples online still show a complete example, including all the necessary packages and libraries.

For beginners

The book sets out to be a "practical and fast-paced introduction" to producing graphics with Ti*k*Z — and it lives up to this promise.

The target audience is users who already have some experience with LaTeX, albeit not necessarily with Ti*k*Z. For readers completely new to the world of Ti*k*Z, the book starts with the very basics, e.g. how to install the package. It then introduces the basic commands and concepts of Ti*k*Z in a well-structured manner.

Compared to the also excellent tutorials in the Ti*k*Z user manual [2], this book is a bit more handholding, which might be appreciated by new Ti*k*Z users looking for more guidance in their endeavour to explore the wonders of Ti*k*Z.

Intermediate Ti*k*Z users

An intermediate Ti*k*Z user may wish to skip over some of the more basic chapters and selectively read only the chapters they are interested in. The individual chapters work well as stand-alone texts. They each start with a short summary and repeat some generic information at the start, like where to find the source code of the examples online. They are also each concluded by their own short summary and a collection of useful links for further reading.

[1] github.com/PacktPublishing/LaTeX-graphics-with-TikZ

For this group of users, *LaTeX Graphics with TikZ* is a good companion to the Ti*k*Z user manual. Due to the focus on only the most commonly used macros and concepts, the book by Stefan Kottwitz can explain certain things in more detail and with helpful visualisations. Concepts such as the `even odd rule` and `nonzero rule` for filling paths benefit from this more in-depth explanation and may be easier to understand than in the user manual.

Advanced Ti*k*Zperts

The scope of *LaTeX Graphics with TikZ* goes beyond the core functionalities of Ti*k*Z. It also introduces the reader to useful third-party packages and libraries, like the `spline` library (which might not be broadly known as it is not available from CTAN) or the `tikz-ext` library which can drastically simplify certain tasks. So if an experienced Ti*k*Z user comes across a copy of the book in a library or on a colleague's bookshelf, they might still find some useful information while thumbing through the book.

Summary

The book is nice to read and well-structured. Even though it is a technical book, it is far from humourless. The author tries to come up with interesting examples, e.g. why teach a user how to draw circles and arcs when you can instead teach them how to draw a smiley (cf. Figure 1)?

Figure 1: One of the example images from *LaTeX Graphics with TikZ* (source code taken from github.com/PacktPublishing/LaTeX-graphics-with-TikZ/blob/main/02-First-steps-creating-TikZ-images/09-filling-colors.tex).

One tiny fly in the ointment: For future editions, I would wish for a different typewriter font to avoid short code pieces (like '`to`') from blending in too much with the surrounding text.

References

[1] S. Kottwitz. *LaTeX Graphics with TikZ: A practitioner's guide to drawing 2D and 3D images, diagrams, charts, and plots.* Packt Publishing, 2023.

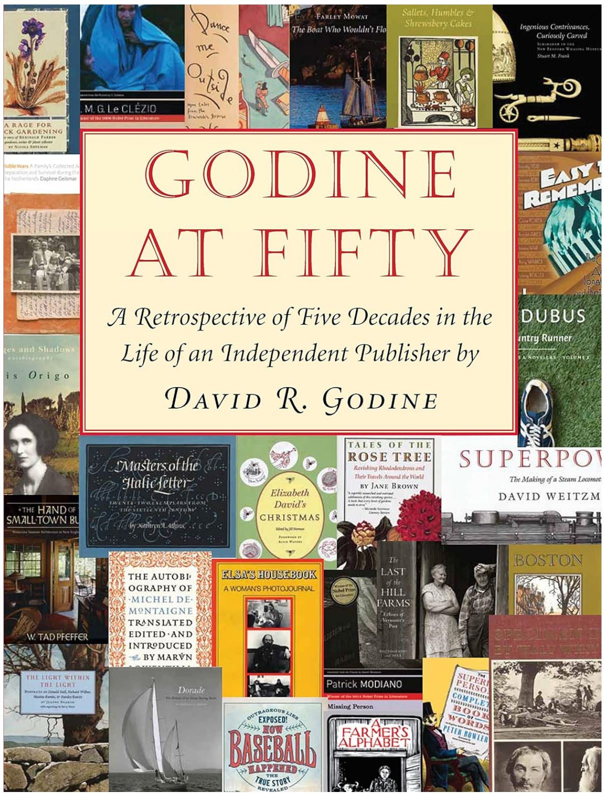
[2] The PGF/Ti*k*Z Team, T. Tantau, et al. *The TikZ and PGF Packages.* ctan.org/pkg/pgf

◇ samcarter

Book review: *Godine at Fifty* by David R. Godine

Karl Berry

David R. Godine, *Godine at Fifty: A Retrospective of Five Decades in the Life of an Independent Publisher*, Godine, 2021, 292 pp., hardcover (9x12), US$60.00, ISBN 978-1-56792-676-7. godine.com/book/godine-at-50



This beautiful oversized volume celebrates a half-century of David R. Godine's publishing ventures. For an overview and brief history of Godine (as the publishing company is now known), please see Dave Walden's "Note on the publisher of the Bodoni book: David R. Godine" in *TUGboat* 37:1 (tug.org/TUGboat/tb37-1/tb115walden.pdf).

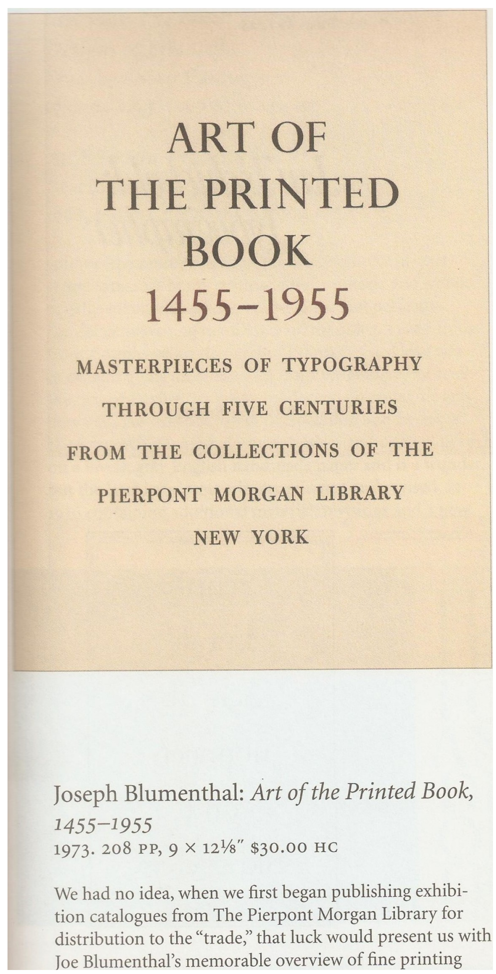
For this retrospective, Godine himself selected nearly 300 titles from his catalog. The cover of the book, shown above, shows a sampling — every book on the cover is included (and many, many, more).

Each entry includes an image or two, usually the cover (as shown here in the entry for Joseph Blumenthal's masterpiece), often also an interior page from the book, and a few paragraphs of commentary by Godine.

His text is engaging, usually remarking on how he came to publish the book, with notes on the design, typesetting, illustrations, sales, and any other interesting aspects of the particular book. He often points out salient aspects of his particular edition, and does not shy away from giving personal opinions, e.g., regret that a book did not sell as well as he had hoped, or gratitude that an author reached out to him to propose a publication.

Godine also wrote an introduction with a brief history of the press. He started his company (as printer as well as publisher) just as hot metal typesetting was reaching its end, through the brief interregnum of phototypesetting, and on into digital, and so is well placed to comment on printing and publishing through the last five decades. The book also includes a few welcome photographs of Godine, his family and associates at work, and some of the equipment used over the years.

Godine himself retired from the company in 2019; it is now ably led by David Allender. We can only be glad he created this book, a superb capstone to his remarkable career.


ART OF THE PRINTED BOOK 1455–1955

MASTERPIECES OF TYPOGRAPHY THROUGH FIVE CENTURIES FROM THE COLLECTIONS OF THE PIERPONT MORGAN LIBRARY NEW YORK

Joseph Blumenthal: *Art of the Printed Book, 1455–1955*
1973. 208 PP, 9 × 12⅛″ $30.00 HC

We had no idea, when we first began publishing exhibition catalogues from The Pierpont Morgan Library for distribution to the "trade," that luck would present us with Joe Blumenthal's memorable overview of fine printing

◇ Karl Berry
https://tug.org/TUGboat

doi.org/10.47397/tb/44-3/tb138reviews-godine50

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`. If you'd like to be listed, please visit that page.

Boris Veytsman Consulting
132 Warbler Ln.
Brisbane, CA 94005
+1 703-915-2406
Email: `borisv (at) lk.net`
Web: `www.borisv.lk.net`

TeX and LaTeX consulting, training, typesetting and seminars. Integration with databases, automated document preparation, custom LaTeX packages, conversions (Word, OpenOffice etc.) and much more.

I have about two decades of experience in TeX and three decades of experience in teaching & training. I have authored more than forty packages on CTAN as well as Perl packages on CPAN and R packages on CRAN, published papers in TeX-related journals, and conducted several workshops on TeX and related subjects. Among my customers have been Amnesty International, Annals of Mathematics, ACM, FAO UN, Google, Israel Journal of Mathematics, No Starch Press, Philosophers' Imprint, Res Philosophica, US Army Corps of Engineers, US Treasury, and many others.

We recently expanded our staff and operations to provide copy-editing, cleaning and troubleshooting of TeX manuscripts as well as typesetting of books, papers & journals, including multilingual copy with non-Latin scripts, and more.

Dangerous Curve
Email: `khargreaves (at) gmail.com`

Typesetting for over 40 years, we have experience in production typography, graphic design, font design, and computer science, to name a few things. One DC co-owner co-authored, designed, and illustrated a TeX book (*TeX for the Impatient*).

We can: ▪ convert your documents to LaTeX from just about anything ▪ type up your handwritten pages ▪ proofread, copyedit, and structure documents in English ▪ apply publishers' specs ▪ write custom packages and documentation ▪ resize and edit your images for a better aesthetic effect ▪ make your mathematics beautiful ▪ produce commercial-quality tables with optimal column widths for headers and wrapped paragraphs ▪ modify bibliography styles ▪ make images using TeX-related graphic programs ▪ design programmable fonts using METAFONT ▪ and more! (Just ask.)

Our clients include high-end branding and advertising agencies, academics at top universities, leading publishers. We are a member of TUG, and have supported the GNU Project for decades (including working for them). All quote work is complimentary.

Hendrickson, Amy
57 Longwood Avenue Apt. 8
Brookline, MA 02446
+1 617-738-8029
Email: `amyh (at) texnology.com`
Web: `www.texnology.com`

Full time LaTeX consultant for more than 30 years; have worked for major publishing companies, leading universities, and scientific journals. Our macro packages are distributed on-line and used by thousands of authors. See our site for many examples: `texnology.com`.

▪ *LaTeX Macro Writing:* Packages for books, journals, slides, posters, e-publishing and more; Sophisticated documentation for users.

▪ Design as well as LaTeX implementation for e-publishing, print books and journals, or specialized projects.

▪ Data Visualization, database publishing.

▪ Innovative uses for LaTeX, creative solutions our speciality.

▪ LaTeX Training, customized to your needs, on-site or via Zoom. See `https://texnology.com/train.htm` for sample of course notes.

Call or send email: I'll be glad to discuss your project with you.

Latchman, David
2005 Eye St. Suite #6
Bakersfield, CA 93301
+1 518-951-8786
Email: `david.latchman (at) texnical-designs.com`
Web: `www.texnical-designs.com`

LaTeX consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized LaTeX packages and classes to meet your needs. Contact us to discuss your project or visit the website for further details.

LaTeX Typesetting
Auckland, New Zeland
Email: `enquiries (at) latextypesetting.com`
Web: `www.latextypesetting.com`

LaTeX Typesetting has been in business since 2013 and is run by Vel, the developer behind `LaTeXTemplates.com`. The primary focus of the service is on creating high quality LaTeX templates and typesetting for business purposes, but individual clients are welcome too.

I pride myself on a strong attention to detail, friendly communication, high code quality with extensive commenting and an understanding of your business needs. I can also help you with automated document production using LaTeX. I'm a scientist, designer and software developer, so no matter your field, I've got you covered.

I invite you to review the extensive collection of past work at the showcase on my web site. Submit an enquiry for a free quote!

Monsurate, Rajiv
Web: `www.rajivmonsurate.com`
`latexwithstyle.com`

I offer: design of books and journals for print and online layouts with LaTeX and CSS; production of books and journals for any layout with publish-ready PDF, HTML and XML from LaTeX (bypassing any publishers' processes); custom development of LaTeX packages with documentation; copyediting and proofreading for English; training in LaTeX for authors, publishers and typesetters.

I have over two decades of experience in academic publishing, helping authors, publishers and typesetters use LaTeX. I've built typesetting and conversion systems with LaTeX and provided TeX support for a major publisher.

Warde, Jake
90 Resaca Ave.
Box 452
Forest Knolls, CA 94933
+1 650-468-1393
Email: `jwarde (at) wardepub.com`
Web: `myprojectnotebook.com`

I have been in academic publishing for 30+ years. I was a linguistics major at Stanford in the mid-1970s, then started a publishing career. I knew about TeX from editors at Addison-Wesley who were using it to publish beautifully set math and computer science books.

Long story short, I started using LaTeX for exploratory projects (see website above). I have completed typesetting projects for several journal articles. I have also explored the use of multiple languages in documents extensively. I have a strong developmental editing background in STEM subjects. If you need assistance getting your manuscript set in TeX I can help. And if I cannot help I'll let you know right away.

Calendar

2023

Nov 15 — Award Ceremony: The Updike Prize for Student Type Design, Speaker: Katie Garth, Providence Public Library, Providence, Rhode Island, USA.
provlib.libcal.com/event/11026344

Dec 1 – 2 — "Face/Interface", Conference on non-Latin typography, Stanford, California, USA.
eventbrite.com/e/faceinterface-global-type-design-and-human-computer-interaction-tickets-720692238887

2024

Feb 4 – 7 — CODEX IX, Oakland, California, USA.
codexfoundation.org

Mar 24 — *TUGboat* **45**:1, submission deadline.

Apr 16 – 20 — Association Typographique Internationale, ATypI Brisbane 2024, "Crafted Technology", Brisbane, Australia.
atypi.org/conferences-events/atypi-brisbane-2024

Jun 4 – Jul 12 — Type Paris Summer 24, intensive type design program, Paris, France. typeparis.com

Jun 6 – 8 — XML Prague 2024, a conference on markup languages and data on the web. University of Economics, Prague, Czech Republic. xmlprague.cz

Jun 12 – 13 — Year of Printing Heritage Conference, Centre for Printing History & Culture, CPHC, Birmingham, UK.
cphc.org.uk/events

Jun 26 – 28 — Twenty-second International Conference on New Directions in the Humanities, "Traveling Concepts: The Transfer of Ideas in the Humanities", Sapienza University of Rome, Rome, Italy, and online.
thehumanities.com/2024-conference

Jul 1 – 5 — SHARP 2024, Society for the History of Authorship, Reading & Publishing, Reading, UK.
sharpweb.org/main/conferences

TUG 2024 — Prague, Czech Republic

Jul — The 44th annual meeting of the TeX Users Group. Presentations covering the TeX world
tug.org/tug2024

Jul 28 – Aug 1 — SIGGRAPH 2024, [location not yet announced]
siggraph.org/siggraph-events/conferences

Aug 6 – 9 — Digital Humanities 2024, Alliance of Digital Humanities Organizations, "Reinvention & Responsibility", Arlington, Virginia, USA, and online.
dh2024.adho.org

Aug 7 – 10 — TypeCon 2024, [location not yet announced]
typecon.com

Status as of 1 November 2023

For additional information on TUG-sponsored events listed here, contact the TUG office by email: office@tug.org). For events sponsored by other organizations, please use the contact address provided.

User group meeting announcements are posted at tug.org/meetings.html. Interested users can subscribe and/or post to the related mailing list, and are encouraged to do so.

Other calendars of typographic interest are linked from tug.org/calendar.html.