

Die T_EXnische Komödie

DANTE
Deutschsprachige
Anwendervereinigung T_EX e.V.

15. Jahrgang Heft 3/2003 September 2003

3/2003

Impressum

„Die T_EXnische Komödie“ ist die Mitgliedszeitschrift von DANTE e.V. Der Bezugspreis ist im Mitgliedsbeitrag enthalten. Namentlich gekennzeichnete Beiträge geben die Meinung der Schreibenden wieder. Reproduktion oder Nutzung der erschienenen Beiträge durch konventionelle, elektronische oder beliebige andere Verfahren ist nur im nicht-kommerziellen Rahmen gestattet. Verwendungen in größerem Umfang bitte zur Information bei DANTE e.V. melden.

Beiträge sollten in Standard-L^AT_EX-Quellcode unter Verwendung der Dokumentenklasse **dtk** erstellt und an untenstehende Anschrift geschickt werden (entweder per E-Mail oder auf Diskette). Sind spezielle Makros, L^AT_EX-Pakete oder Schriften dafür nötig, so müssen auch diese mitgeliefert werden. Außerdem müssen sie auf Anfrage Interessierten zugänglich gemacht werden.

Diese Ausgabe wurde mit Hilfe folgender Programme erstellt: **pdfTeX 3.14159-1.00b-pretest-20020211 (Web2C 7.3.7x)**, **LaTeX2e (2001/06/01)**, **Acrobat Reader 5.0.5** und **xdvi(k) 22.40k** für die Bildschirmdarstellung. Als Standard-Schriften kamen die Type-1-Fonts Latin-Modern zum Einsatz.

Erscheinungsweise: vierteljährlich

Erscheinungsort: Heidelberg

Auflage: 2700

Herausgeber: DANTE, Deutschsprachige Anwendervereinigung T_EX e.V.
Postfach 10 18 40
69008 Heidelberg

E-Mail: dante@dante.de

dtk-redaktion@dante.de (Redaktion)

Druck: Konrad Triltsch Print und digitale Medien GmbH
Johannes-Gutenberg-Str. 1–3, 97199 Ochsenfurt-Hohe Stadt

Redaktion: Gerd Neugebauer (verantwortlicher Redakteur)

Luzia Dietsche	Gert Ingold	Volker RW Schaa
Rudolf Herrmann	Rolf Niepraschk	Herbert Voß
Moriz Hoffmann-	Günter Partosch	
Axthelm	Bernd Raichle	

Redaktionsschluss für Heft 4/2003: 4. Oktober 2003

ISSN 1434-5897

Die T_EXnische Komödie 3/2003

Editorial

Liebe Leserinnen und Leser,

oftmals kommt es anders als man denkt. So auch diesmal bei der Erstellung der Vereinszeitschrift...

DANTE e.V. hat die Erstellung der Latin-Modern-Schriften als Type1-Schriften mit T1-Kodierung gefördert. Da lag auch der Gedanke nahe, dass wir für „Die T_EXnische Komödie“ diese Schriften einsetzen – bisher kamen die CM-Super-Schriften zum Einsatz. Irgendwie wäre der Jahreswechsel ein guter Zeitpunkt für solch eine Neuerung gewesen, da wir in der Regel Änderungen am Layout immer mit einem neuen Jahrgang der Mitgliederzeitschrift verbinden.

Nun haben wir einen Beitrag über die Latin-Modern-Schriften erhalten und es wäre etwas merkwürdig, die neuen Fonts zu beschreiben und diesen Beitrag mit anderen Schriften zu setzen. Ganz abgesehen davon haben die Latin-Modern-Schriften beispielsweise eine fk-Ligatur, die auch in dem Beitrag gezeigt wird – und wie sollen wir eine Ligatur zeigen, die es in den benutzten Fonts nicht gibt?

So ist es dann am Ende doch dazu gekommen, dass wir diese Ausgabe mit den Latin-Modern-Schriften gesetzt haben. Dem aufmerksamen Leser wäre das aber sicher schnell aufgefallen. Spätestens bei der alten Form des ß scheiden sich die Geister. Hier haben die „Traditionalisten“ nun die ursprüngliche Form aus den CM-Schriften wieder.

Natürlich gibt es über das Schriften-Thema hinaus auch noch weitere interessante Beiträge in diesem Heft, auf die ich in der Kürze dieses Editorials nicht näher eingehen will – jeder mag selbst sehen, wieviel Nützliches dabei ist.

Damit verbleibe ich, wie immer,
mit T_EXnischen Grüßen

Ihr Gerd Neugebauer

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

wir hoffen, dass Sie uns die Länge dieses Grußwortes verzeihen und trotz allem bis zum Ende lesen, denn es gibt viel zu berichten.

Die 14. EuroT_EX-Tagung mit dem Motto „**Back to Typography**“ in Brest (Frankreich) liegt hinter uns. Außer unserem Schatzmeister nahm diesmal der gesamte Vorstand teil. Die Tagung war sehr gut besucht, fast 130 Teilnehmer aus 25 Ländern waren erschienen. Der von DANTE e.V. mitfinanzierte Bus, der von Polen über Deutschland und Belgien nach Brest fuhr, ermöglichte zudem etlichen T_EXies aus Polen, Ungarn, Tschechien und Dänemark eine kostengünstige Teilnahme. Näheres über das Tagungs- und Begleitprogramm können Sie demnächst im Bericht von Thomas Lotze nachlesen, hier wollen wir nur die interessantesten Informationen aus dem LUG-Meeting, Treffen der anwesenden Vertreter und Repräsentanten der T_EX-Local-User-Groups, weitergeben.

Insgesamt waren 15 Gruppen (bzw. Länder) vertreten und berieten über aktuelle Themen. Hier war insbesondere die bei der letzten EuroT_EX in Polen beschlossene Abfolge der Konferenzen ein Hauptpunkt der Diskussion. Durch die ohne Rücksprache mit den europäischen Gruppen erfolgte Vergabe der TUG-Tagung 2004 nach Griechenland, wäre die Regel verletzt worden, nach der im selben Jahr nie eine EuroT_EX und eine TUG-Tagung in Europa stattfinden sollen. Nach langen Beratungen, die sich noch über das Meeting selbst hinaus hingen, wurde dann eine für (fast) alle akzeptable Lösung gefunden. Sie wurde durch den Rückzug der dänischen T_EX-Users-Group (DK-TUG) ermöglicht, die sich für 2005 außer Stande sah, die finanziellen und organisatorischen Voraussetzungen für eine Konferenz zu schaffen.

Somit wird die 15. EuroT_EX-Tagung nach dem jetzigen Planungsstand von GUTenberg und DANTE e.V. im Rahmen des gemeinsamen 16. Geburtstags

beider User-Groups im Raum Straßburg Ende Februar/Anfang März 2005 ausgerichtet. Zum Abschluss der Konferenz in Brest wurde dann auch dieser Termin zusammen mit der 26. TUG-Tagung im September 2004 in Xanthi (Griechenland) angekündigt.

Die weitere Abfolge der EuroTeX-Tagungen wurde bei dem Meeting bestätigt: 2006 in Ungarn (MaTeX), 2007 in Irland und 2008 im Grenzgebiet Tschechien/Polen (CSTUG/GUST). Die dänische TeX-Users-Group (DK-TUG) bat um die Wiederaufnahme in die Liste für 2009.

Ein weiterer Punkt, der auf dem LUG-Meeting ausführlich diskutiert wurde, war das geplante „TeX Article Repository“. Hierin sollen neue Artikel abgelegt werden können, die über den Sprachraum, in dem sie veröffentlicht werden, hinaus von Interesse sind, als Demonstration für die Stärke von (L)TeX stehen oder Neues im Umgang mit TeX oder für Typographie aufzeigen. Alle LUGs sind eingeladen, diese Artikel zu veröffentlichen und Übersetzungen dafür ins Repository zu stellen. Das Motto lautet: „Provide, Share, and Translate“. Autoren sollen in Zukunft gefragt werden, ob sie mit einer solchen Nutzung ihres Artikels einverstanden sind. Darüber hinaus wurde die Möglichkeit eines internationalen Journals als „TeX and Typography Showcase“ diskutiert, das unregelmäßig (maximal einmal pro Jahr) einen Teil dieser Artikel in unterschiedlichen Sprachen aufnimmt und den Mitgliedern verschiedener Gruppen zusätzlich zu ihrer Mitgliederzeitschrift zugesandt wird.

Weitere Punkte der Tagesordnung waren zum einen die Vorstellung der zur Zeit geförderten Projekte im Rahmen der „European TeX Project Funds“ (wir berichteten in der letzten Ausgabe von „Die TeXnische Komödie“) und die Einladung an andere LUGs, sich diesem Fördervorhaben anzuschließen. Zum anderen war wiederum das Verhältnis zur TUG ein Thema. Beim letzten LUG-Meeting in Polen war ein offizieller Brief der europäischen Gruppen an die TUG ausgearbeitet und abgesandt worden. Dieser Brief wurde bis heute nicht beantwortet. Inzwischen hat sich allerdings durch die Neuwahl des Präsidiums der TUG einiges geändert. Seit März 2003 ist Karl Berry „President-elect“ und hat das Amt offiziell mit der Mitgliederversammlung im Rahmen der TUG-Tagung in Hawai‘i im Juli angetreten. Der Tagesordnungspunkt wurde vertagt, da sich durch Karl Berry einiges am Stil des TUG-Boards geändert hat und inzwischen ein intensiver Informationsaustausch stattfindet. Mittlerweile hat ein vereinbartes Telefongespräch zwischen Volker RW

Schaa und Karl Berry stattgefunden, in dem über das LUG-Meeting berichtet wurde und die weiteren Schritte der Zusammenarbeit besprochen wurden.

Ein weiterer Punkt der Tagesordnung betraf den „ $\text{\TeX}$ Calendar“, der den Mitgliedern von DANTE e.V. schon auf der letzten Tagung in Bremen vorgestellt wurde, bisher aber noch keine allgemeine Verbreitung fand. Dieser Web-Kalender bietet den einzelnen LUGs die Möglichkeit, ihre Termine frühzeitig in eigener Regie zu veröffentlichen. Der Kalender erlaubt die Darstellung von Terminen einzelner oder aller User-Groups in unterschiedlich detaillierter Wochen- und Monatssicht. Allgemein zugänglich ist der $\text{\TeX}$ Calendar unter der Adresse <http://texcalendar.dante.de>.

Nun zu einem anderen Thema: Am 15. Mai hatten Hans Hagen und Volker RW Schaa ein Treffen mit Donald Knuth während seines Aufenthaltes in Antwerpen. Zweck dieser Begegnung war zum einen die Vorstellung des „Latin Modern Font Projects“ (siehe Förderbericht in der letzten Ausgabe von „Die $\text{\TeX}$ nische Komödie“, Artikel von Bogusław Jackowski und Janusz Marian Nowacki in dieser Ausgabe und Latin-Modern-Fonts Version 0.86 unter `CTAN:/fonts/pstypel/1m`) und zum anderen die Planung eines Treffens von Donald Knuth mit Hermann Zapf in Darmstadt. Prof. Hermann Zapf hatte Volker RW Schaa um die Planung des Treffens gebeten, da er leider für ein Treffen in Antwerpen verhindert war, weil er sich zu diesem Zeitpunkt in den USA zur Verleihung der Ehrendoktorwürde aufhielt (Universität von Urbana-Champaign, Illinois).

Donald Knuth war von dem Font-Projekt angetan und empfahl nach kurzem Studium der überreichten Font-



Besuch bei Donald E. Knuth in Antwerpen und Vorstellung der „Latin-Modern-Fonts“

Broschüre einige Überarbeitungen. Insbesondere war es ihm wichtig, dass einige Fehler und Unschönheiten in den Computer-Modern-Fonts, die er schon in seinen Artikeln bemängelt hatte, aber wegen der sich sonst ändernden Metriken nie korrigieren wird, behoben werden. Zudem hatte er darüberhinaus etliche Änderungsvorschläge, die wir gern an Bogusław und seine Mitstreiter weitergeben werden.

Über die Verleihung der Ehrendoktorwürde an Prof. Hermann Zapf freute sich Donald Knuth sehr und betonte, dass es viel zu lange gedauert hätte, bis die Verdienste von Prof. Zapf gewürdigt worden sind. Auf ein Treffen in Deutschland angesprochen, sagte er, dass er gern kommen würde, wenn es seine Zeit erlaube. Auf die Frage nach einem möglichen Termin erwähnte er seinen Studien-Aufenthalt in Schweden von Januar bis Anfang März im kommenden Jahr. Da sich dies hervorragend mit der in Darmstadt geplanten Frühjahrstagung zum 15. Geburtstag von DANTE e.V. traf, wurde unser Ehrenmitglied sofort zu dieser Feier eingeladen.

Eine Woche nach diesem Treffen erreichte uns die folgende E-Mail von Donald Knuth:

One correction: I said that I would be living in Sweden during the first several months of 2004. But the truth is that I plan to live in Sweden during the first several months of 2005. At present I have no plans to travel to Europe [...] until then.

And of course that would be the 16th anniversary of Dante --- a most impressive anniversary, according to Digital Typography page 574.

So können wir uns nun vielleicht auf eine Begegnung mit Donald Knuth beim 16. Geburtstag und der dann stattfindenden EuroT_EX-Tagung zusammen mit GUTenberg im Jahre 2005 freuen.

Mit diesen Informationen schließen wir unser Grußwort und würden uns freuen, möglichst viele Mitglieder auf unser nächsten Tagung von DANTE e.V. und 29. Mitgliederversammlung in Rauschholzhausen begrüßen zu können, mit freundlichen Grüßen,

Volker RW Schaa Klaus Höppner
(Präsident) (Vizepräsident)

1. Bayerischer T_EX-Stammtisch in Nürnberg

Uwe Siart

Insgesamt zehn Mitglieder der beiden bayerischen T_EX-Stammtische in Erlangen und München trafen sich am Samstag, den 9. August 2003 zum ersten Mal zu einem gemeinsamen T_EX-Stammtisch in Nürnberg.

Anreise

Es war ein wolkenloser, glühend heißer Sommertag mit Lufttemperaturen jenseits von 35°C. Der eine oder andere der sechs Teilnehmer aus München stattete sich für die etwa eineinhalbstündige Fahrt mit dem PKW noch schnell mit großen Wasserflaschen und Eis am Stiel aus und dann konnte es losgehen. Ein unfallbedingter Stau auf der Autobahn war durch die Klimatisierung des Autos zum Glück einfach zu überstehen. In Nürnberg angekommen erwartete uns ein gut gekühlter Besprechungsraum und ebensolche Getränke, die uns von der Sommerhitze schnell erholen ließen. So konnten wir wie vorgesehen um 15.00 Uhr mit unserem Treffen beginnen.

Vorträge

Für den bayerischen T_EX-Stammtisch waren zwei Vorträge vorgesehen, die wegen der regen Frage- und Diskussionsbeteiligung auch fast drei Stunden in Anspruch nahmen. Zuerst gab Michael Niedermair die Einführung „Zeichnungen mit PSTricks erstellen“. Es war eine Übersicht über die Verwendung und die Möglichkeiten des Grafikpakets PSTricks. Der Vortrag zeigte das Erstellen grundlegender Grafikobjekte und die Optionen zur Steuerung grafischer Eigenschaften wie Linientyp, Linien- und Füllfarbe und vieles mehr. Im Hinblick auf wissenschaftliche Anwendungen enthielt der Vortrag auch die Möglichkeiten, Funktionen zu plotten, Knoten in einer Grafik festzulegen und auf vielseitige Weise zu verbinden. Die Einführung schloss mit einer Vorstellung möglicher Abläufe zur Verwendung von PSTricks mit pdf_latex und einem kurzen Überblick über verschiedene Ergänzungspakete für spezielle Anwendungen.

Im zweiten Vortrag stellte der Betreuer des PSNFSS-Paketes Walter Schmidt in seiner „L^AT_EX-Schriftengalerie“ verschiedene Kombinationen von Text- und

Mathematikschriften vor. Die vorgestellten Kombinationen konnten am Ausgang der zahlreichen Papierausdrucke in Augenschein genommen und individuell beurteilt und diskutiert werden. Er gab damit hilfreiche Vorschläge für geeignete Mathematikschriften, wenn für eine bestimmte Grundschrift entweder keine oder zumindest keine kostenfreien Mathematikschriften verfügbar sind. Sehr wertvoll waren dabei auch seine Hinweise und Erklärungen, zum einen über den verfügbaren freien Schriftenvorrat, zum anderen über den Schriftenzugriff unter L^AT_EX und unter PostScript.

Diskussionsabend

Als nach 18.00 Uhr die Vorträge abgeschlossen waren, die Diskussionen aber in keiner Weise zum Erliegen kamen, setzten wir unser Treffen im Freien auf der Terrasse bei Grillfleisch, Salaten und gekühlten Getränken fort. Vor allem die Gelegenheit, Walter Schmidt zu Interna über die Schrifteneinbindung und die L^AT_EX-Anpassung kommerzieller Schriften befragen zu können, wurde gerne und rege in Anspruch genommen. Es gab ansonsten einen sehr hilfreichen Austausch zu Stärken und Schwächen von T_EX, dessen Einsatz in der Firmenkorrespondenz, über die Qualitäten, Geschichte und angemessene Anwendung verschiedener Schriftfamilien sowie die Möglichkeiten von Ergänzungspaketen. Es war ein sehr interessanter Abend an dem die einen Fragen hatten und die anderen antworten mussten. Der Abbruch gegen 22.30 Uhr kam leider viel zu früh, aber ein Großteil der Münchener Teilnehmer musste schließlich noch die Rückfahrt antreten.

Fazit

Der 1. Bayerische T_EX-Stammtisch in Nürnberg war ein gelungenes Treffen von T_EX- und Typographieinteressierten aus dem Nürnberger und Münchener Raum. Alle Beteiligten bedanken sich bei Peter Seitz für seine Vorbereitungen und bei der Firma Köhler + Seitz Beraten und Planen GmbH für die Überlassung ihrer Räumlichkeiten und Infrastruktur und für die freundliche Getränkespende. Es konnten viel Wissen und viele Tipps weitergegeben und ausgetauscht werden. Ich meine, es wurden jedes Teilnehmers Erwartungen an dieses Treffen vollständig erfüllt. Alle sind sich einig, dass ein solches Treffen im nächsten Jahr wiederholt wird.

Bretter, die die Welt bedeuten

Accents, accents, accents. . .

enhancing CM fonts with “funny” characters

Bogusław Jackowski, Janusz M. Nowacki

Dieser Artikel beschreibt die Geschichte, Entstehung und Schwierigkeiten bei der Entwicklung der „Latin Modern Fonts“, die als Projekt der European T_EX Users Groups (DANTE e.V., GUTenberg, NTG und GUST) im Rahmen der Projektfonds gefördert wurden. Die CM-Type-1-Fonts der AMS stellten die Basis für die LM-Fonts. In LM sind aber nun nicht nur alle akzentuierten Zeichen als eigenständige Glyphen (zur Zeit maximal 527 pro Font) enthalten, es werden auch Zeichen der EC-Fonts aufgenommen, um einen vollständigen Ersatz zu schaffen. Im Artikel werden die technischen Mittel und Wege zur Realisierung erläutert und die Entscheidungen bezüglich der Kompatibilität zu den CM-Fonts begründet. Die LM-Fonts stehen seit dem 5. August 2003 in der Version 0.86 im CTAN zur Verfügung und wurden in die T_EXLive 2003 eingebunden.

Introduction

Accented characters play the rôle of *enfants terribles* in the world of computers. Anybody who has to communicate with another computer system in a language other than English knows that using so called “funny characters” is not fun at all.

Those pesky diacritics

A giant step towards putting some order into the chaos was the Unicode standard (ISO/IEC 10646) published ten years ago. Unicode, obviously, does not remove all the problems from the font’s playground, and even adds a few

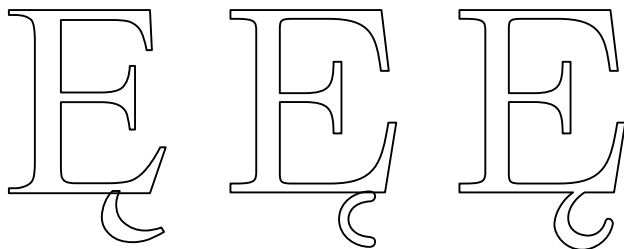


Figure 1: The letter *Eogonek* from Times New Roman for Windows XP (left), from `aer10` (middle), and from `lmr10` (right); only the latter form, i.e., having a single outline, is acceptable in professional applications.

specific ones (e.g., the problems with the size of fonts or with the registration of non-standard characters and languages). Nevertheless, one can believe that the world will become a bit better when Unicode turns from the standard *de nomine* to the standard *de facto*.

$\text{\TeX}$ with its 8-bit (i.e., 256 characters per font) paradigm is more and more obsolescent and enhancing it by multi-byte character codes seems unavoidable. Such efforts as Ω Project [11], developed by John Plaice and Yannis Haralambous, cannot be overestimated from this point of view. But the typesetting system itself is only one side of the coin. The other is a collection of fonts it uses.

Originally, $\text{\TeX}$ was equipped with Computer Modern family of fonts (CM) which did not contain diacritical characters. Those few $\text{\TeX}$ users who would need accented letters were supposed to employ the `\accent` primitive. The immense popularity of $\text{\TeX}$ in countries that use lots of diacritical characters proved this presumption invalid. At least three reasons can be set forth: (1) accented characters do not behave like “normal” ones, e.g., they interfere with important $\text{\TeX}$ algorithms such as hyphenation and insertion of implicit kerns; (2) the CM fonts do not contain all necessary diacritics, e.g., an ogonek accent (used in Polish, Lithuanian, Navaho) is missing; (3) such diacritical elements as cedilla and ogonek, when treated as “accents,” overlap with a letter, which precludes some applications, e.g., preparing texts for cutting plotters (see figure 1), even if outline fonts are used. The lesson is obvious—the CM family should be extended by a variety of diacritical letters.

In this paper we would like to present our approach to solving the problem, i.e., the open source family of fonts, *Latin Modern* (LM), in the PostScript Type 1 format [2], prepared using METATYPE1, a METAPOST-powered package [8] (see section “METATYPE1”, p. 16). We believe that the LM family is a decent alternative to the existing extensions of the CM family—we expect it to be a handy collection of fonts for typesetting in Latin-based alphabets. The fonts are also equipped with *Printer Font Metric* files (*.pfm) and therefore can be used as system fonts in GUI systems. We plan to release the METATYPE1 sources of the LM fonts soon after the 24th Annual Meeting and Conference of the T_EX Users Group, July 20th–24th, 2003, Big Island, Hawai‘i.

A gulp of history

Needless to say, the lack of diacritical letters in the CM family was recognized almost from the very beginning by T_EX users who had to struggle with the typesetting of languages other than English. Only in 1990, however, during the TUG meeting in Cork, Ireland, did the international T_EX community decide that fonts in so called Cork Encoding (EC or, in L^AT_EX lingo, T1) should be prepared for European T_EX users [6]. The work on EC fonts started soon after the Cork meeting. Norbert Schwartz designed a prototype, so called DC fonts. The work was then continued by the team led by Jörg Knappen. The final release of EC fonts was announced in 1997.

It was an important achievement. Nevertheless, the Cork Encoding conformed to T_EX’s 8-bit paradigm and therefore it was not able to comprise all characters occurring in European languages, not to mention other Latin alphabets, such as Vietnamese or Navaho.

For a few years, EC fonts were available only in a T_EX-specific bitmap form (pk). Nowadays, in the advent of electronic publishing, bitmaps are hardly acceptable. At least two factors can be pointed out: (1) the scaling of bitmap fonts is troublesome—they look nice only if their resolution matches the resolution of the device; (2) in many cases, outline fonts turn out to display much better and, paradoxically, faster on a screen, e.g., when used in PDF (although there exist programs which display bitmap fonts better than Adobe Acrobat Reader).

This inspired Lars Engbreetsen who prepared a set of T_EX virtual fonts containing basic diacritical characters [4]. The virtual fonts could refer to the

excellent outline version of the CM family which had appeared in the meantime. It had been created in 1988 by Blue Sky Research for the American Mathematical Society in PostScript Type 3 format, converted in 1990 by Y&Y into the hinted Type 1 format, and released in 1997 into the public domain by the American Mathematical Society. Engebretsen called his collection AE—“Almost EC.” His virtual fonts suffer, however, from the same limitation as $\TeX$ does, i.e., the number of characters is limited to 256; moreover, as we have mentioned, superimposing a diacritical element on a character reveals undesirable features when the character is stroked rather than filled (figure 1).

Only recently, automatically traced fonts in the PostScript Type 1 format, based on the EC fonts, have been published: Péter Szabó’s **Tt2001**, Vladimir Volovich’s CM-SUPER (both in 2001; [14] and [15], respectively), and a new-fangled Alexey Kryukov’s CM-LGC (March, 2003). Note, however, that Szabó courteously “recommends the wonderful CM-SUPER package instead of his own **Tt2001**.” Indeed, Volovich’s collection contains much more font variations and covers a broader character set than Szabó’s one. Kryukov’s collection is, in a way, a supplement to CM-SUPER. The creation of these packages was possible thanks to a marvellous tool provided by Martin Weber, namely, **autotrace** [16].

Volovich’s accomplishment seems to bring to an end the long-lasting endeavours to introduce diacritical characters into the $\TeX$ ’s realm. Do we really need one more collection of fonts?

Another viewpoint

Autotraced fonts, in spite of their many advantages, have drawbacks. Objectively, the most important one is perhaps the size of a font. Such fonts are usually larger than similar fonts having carefully designed outlines because of the greater number of nodes. Compare, for example, fairly tidy Volovich’s CM-SUPER fonts with $\mathcal{AMS}$ CM and LM: the number of bytes per character is 260, 200, and 135, respectively. Twice is not too much, but in connection with lots of magnifications included (see section “Too many font sizes”, p. 14) it makes a difference. Incidentally, the size of the CM-SUPER fonts can be reduced by circa 10 percent by using a subroutine compression module PACKSUBR from METATYPE1 (actually, it is a short **awk** script).

For us, however, more important are arguments of a rather imponderable nature. We stay firmly by the conception underlying the $\text{\TeX}$ and $\text{\METAFONT}$ design: *every detail*, be it a typesetting or a typeface design, *should be controllable and replicable*.

This is not the case with autotraced fonts. You must rely, e.g., on the nodes selected by the tracing engine. Volovich notes that **FontLab** program (very good but commercial) was used for improving the fonts, namely, for hinting and reducing the number of nodes; therefore, the process cannot be easily repeated somewhere else. In other words, there are actually no sources for the CM-SUPER family. The consequence is that **tfm** files have to be generated from **afm** ones (using, e.g., **AFM2TFM** program) which adds further uncontrolled factors. For example, one cannot suppress overshoots, i.e., characters ‘o’ and ‘x’ will usually have heights slightly different, unlike the original CM fonts.

Speaking about **AFM2TFM** converter, please note that it cannot produce mathematical fonts. One has to use **METAFONT** or **METAPOST** (or edit manually property lists generated by **tftopl** or **vftovp**) in order to exploit such features as **charlist** or **extensible**. Ignoring this aspect would mean, in our opinion, the waste of $\text{\TeX}$ equipment for mathematics.

Having said this, we would like to emphasize that we esteem the job Szabó, Volovich, and Kryukov did. Our predilection to another solution may be regarded as a natural, if not advisable, difference of viewpoints.

Too many font sizes

There is one more issue, related indirectly to the problem of ‘bitmaps versus outlines,’ namely, the number of font sizes for a given typeface, or more adequately—proportions. Donald E. Knuth, following the typographic praxis, implemented fonts having different proportions for different sizes (5, 6, 7, 8, 9, 10, 12 and 17 points). John Sauter attempted to go even further [13]. He prepared **METAFONT** programs that interpolate (and even extrapolate) Knuth’s font parameters to non-integer font sizes. We can accept Sauters’s approach as an interesting experiment, admissible for bitmap fonts. Nevertheless, using it for outline fonts is at least controversial.

We believe that, in general, four font proportions would suffice: heading (17 pt), normal (10 pt), script (7 pt), and second order script (5 pt, “scriptscript”). Because of the well-established tradition, we cannot refrain from using Knuth’s scheme, but we would strongly discourage extending it.

For these reasons, we accept with difficulty the enormous number of different sizes/proportions both in EC and CM-SUPER fonts. This is apparently the inheritance of Knuth's and Sauter's ideas. We would gladly discard most of fourteen alterations of a single typeface (corresponding to font sizes 5, 6, 7, 8, 9, 10, 10.95, 12, 14.4, 17.28, 20.74, 24.88, 29.86, and 35.83 points). The series proposed by Knuth plus `TEX scaled` and `at` operations provide sufficient means to deal with font scaling in most typical applications.

Too few typefaces

If anything, completely new typefaces are needed. The number of fonts prepared with METAFONT is surprisingly small compared, e.g., to what is available on the commercial market. Well, not so surprisingly. As we have already mentioned, METAFONT generates T_EX-oriented `pk` bitmap fonts which have not become popular outside the T_EX world. In principle, the conversion of `pk` bitmaps into PostScript Type 1 form is possible, as Szabó and others proved. Which does not mean that looking for alternative tools is impractical.

Alternative tools

In general, there are two classes of computer tools: visual (interactive) and logical (programmable). Perhaps someday both classes will converge into, say, “visual-and-logical” tools which will prevail, but at present, no doubt, interactive tools are in vogue. The majority of contemporary visual typographic programs are commercial products. Fortunately, George Williams launched (in 2000) an impressive open source project, `PfaEdit` [17]. This font editor is already a powerful tool and, being extensively developed, it promises even more, which countervails the grasping market up to a point. An interesting visual tool for generating PostScript Type 1 fonts is also Richard Kinch's `MetaFog` which enables visual tuning of METAPOST-generated PostScript files [9].

Programming tools are not so popular. Are they to go extinct some day? We hope they will not. It would be a pity, because in some applications programmability is better. Fortunately, there exist people who share our point of view. One of them is Włodek Bzyl who found a plausible application for the logical approach in typography. His amazing colour PostScript Type 3 fonts are no mean challenge for those who use visual tools [3].

Fonts are very complex structures. They are governed by the ample set of interdependent parameters, such as character dimensions, font-specific parameters (italic angle, x-height, typical stems), characteristic shapes (serifs and arcs), not speaking about such technicalities as hints or subroutines. And here an important aspect of programmability enters. By definition, programmable tools require sources in a human-readable text form. A plethora of text processing utilities around (**awk**, **perl**, **grep**, **diff**) can therefore be employed to crosscheck the consistency of the data describing the font. This can hardly be achieved with purely interactive programs, although it should be noted that some interactive typographic programs have implemented limited programmability.

METATYPE1

We prefer unlimited programmability. Being provoked by the irksome scarcity of fonts prepared using METAFONT, we contrived another font generating package, METATYPE1 [8], based on METAPOST, which produces results in the world-wide accepted PostScript Type 1 format. The package makes use of two sets of METAPOST macros (general purpose **plain_ex** and task-oriented **fontbase**) and a few other utilities, such as **awk** (for processing METAPOST output), **Tlutils** (for converting text data into a binary form), and **mft** (for neat proofing). Originally, METATYPE1 was developed for DOS; thanks to Włodek Bzyl, it is available also for Linux.

Those who are repelled by the sophisticated software and therefore are unwilling to experiment with METATYPE1 may find Han-Wen Nienhuys's opinion encouraging: "METATYPE1 is a very simplistic approach" [10].

One of the first results obtained with METATYPE1 was Donald E. Knuth's **logo** font and an electronic replica of a traditional Polish font, Antykwa Półtawskiego [7]. We also used METATYPE1 for auditing and enhancing selected fonts from the URW++ collection distributed with **GhostScript**.

In 2002, during the T_EX meeting in Bachotek, Poland, the representatives of European T_EX users group, having discussed the matters on email, came up with a proposal of converting AE virtual fonts into a more universal PostScript Type 1 format and augmenting them with a set of necessary diacritical characters. Thus the opportunity arose to embark METATYPE1 upon a new, unconventional task. We took up the gauntlet without hesitation.

lmb10	lmr17	lmss10	lmssq8
lmb010	lmr5	lmss12	lmssqbo8
lmbx10	lmr6	lmss17	lmssqbx8
lmbx12	lmr7	lmss8	lmssqo8
lmbx5	lmr8	lmss9	lmtcsc10
lmbx6	lmr9	lmssbo10	lmtt10
lmbx7	lmri10	lmssbx10	lmtt12
lmbx8	lmri12	lmssdc10	lmtt8
lmbx9	lmri7	lmssdo10	lmtt9
lmbxo10	lmri8	lmss010	lmtti10
lmbxi10	lmri9	lmss012	lmtto10
lmcsc10	lmro10	lmss017	lmvtt10
lmcsc010	lmro12	lmss08	lmvtt010
lmr10	lmro8	lmss09	
lmr12	lmro9		

Figure 2: The Latin Modern collection of fonts.

The LM family of fonts or details, details, details. . .

Our intention was to preserve the AE name, as we wanted to emphasize the rôle of Englebretsen’s idea in this enterprise. Soon it became clear, however, that the differences would be fundamental and that the change of the name would be necessary in order to avoid a mess. Therefore, we coined the name “Latin Modern” which is to betoken further development—we would like the final version of LM to comprise as many Latin-based alphabets as possible, e.g., Vietnamese which regrettably is not included yet.

The collection of AE fonts consisted of 50 fonts, reasonably selected from the abundance of Computer Modern. We decided to add a variable-width typewriter font and a few oblique derivatives, arriving finally at 57 fonts (see figure 2).

Observe two details:

1. We adopted a more regular (although unorthodox) font naming convention with respect to slant/italic variants: we have preserved the 8-character limit but we have used the letter ‘o’ as a suffix for oblique (slanted) fonts and the letter ‘i’—as a suffix for truly italic fonts.



Figure 3: The letter *I* from Knuth's `cmssq8` (left) and MacKay's `lcmss8` (right).

2. The LM family contains font `lmssqbx8` (i.e., the bold version of `lmssq8`); a corresponding font occurs neither in CM nor in EC. Actually, the respective AE fonts (`aessq8`, `aessqi8`, and `aessqb8`) refer to the fonts `lcmss8`, `lcmssi8`, and `lcmssb8`. These fonts, added by Pierre A. MacKay, were meant for using with $\text{\LaTeX}$. Their regular variants are nearly identical with Knuth's `cmssq8` and `cmssqi8`. The only difference is the capital 'I' (see figure 3).

The issue of font names was triggered by the slanted fonts that we decided to add: what name should we assign to the oblique variant of `lmvtt10`? The name `lmvttsl10` did not conform to the Knuthian 8-character canon, the name `lmvtti10` did not tell the truth. Having thought the problem over, we could not find the reason why oblique fonts, i.e., the mechanically skewed ones, received the designator 'i' in some cases (e.g., `cmssi10`) and the designator 'sl' in other (e.g., `cmbxsl10`), and why the designator appeared either at the end of the kernel of the name, as in the mentioned examples, or—in some cases—immediately after the prefix 'cm' (`cmslt10`, `cmitt10`). We could either uphold traditional Knuth's terminology (but how then should we call oblique `lmvtt10`?) or take an opportunity and introduce some regularity in font naming at the risk of commencing an incompatibility mess. We have chosen the latter solution...

The issue of an alternative letter 'I' necessitated, besides undertaking a decision whether to introduce it or not (we decided to introduce it as a variant letter), some extra work due to the addition of variant accented characters and a variant ligature *IJ*. The `lmssq*` fonts became thus somewhat exceptional. This is usually undesirable but sometimes cannot be avoided.

The reader may wonder why to dwell on such trifles? The answer is simple: it was the bulk of details of this kind that made the work on the LM family laborious, although individual tasks were relatively simple. In other words,

the problem with details is that each of them, even the tiniest one, has to be handled *somehow*—if the amount of details grows, the job becomes complex.

Enumerating all dilemmas, technicalities, subtleties or even puzzles with which we had to struggle is obviously pointless. On the other hand, our work consisted nearly exclusively of details—how to describe such a work? Perhaps the best method is to let the reader perceive the scent of the battleground by showing representative examples. Two such examples we have already indicated. The rest of the paper presents a few more of them.

From PostScript to METATYPE1 sources

The process of conversion of fonts from PostScript Type 1 form into METATYPE1 sources is moderately relevant since the potential users of the LM fonts are not expected to repeat this operation any more. The METATYPE1 sources are legible and can easily be modified, if necessary.

We used a stand-alone utility PF2MT1 (belonging to the METATYPE1 package) for the translation of `pfb + afm` couples from CM fonts into METATYPE1 code. The virtual AE fonts provided the necessary information for merging appropriately the results of conversion. A very convenient tool for such operations is `awk`. Thanks to it, the framework of the LM sources was ready after a few hours; amending the LM sources took a few months.

Tuning and augmenting the METATYPE1 sources of the LM fonts

The main part of the job, although the simplest one, was adding accents. METATYPE1 provides a `use_accent` operation, similar to the `TEX \accent` primitive, that can conveniently be used for this purpose. By default, `use_accent` aligns the centre of an accent with the centre of its accentee and raises the accent by $x - h$, where x is the value of the x-height parameter, and h is the height of the character. This is the procedure that is used by T_EX for accenting. Such an algorithm is not always eligible. Occasionally, the position of accent may have to be adjusted. The command `use_accent` enables an arbitrary shift of both accent and accentee. Moreover, a supplementary parameter can optionally be specified for each character—a glyph axis (see figure 4).

All in all, adding accented letters was child's play. Somewhat more difficult was adding extra characters.

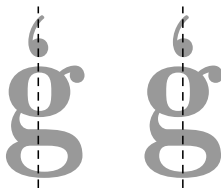


Figure 4: The optical axis of a glyph does not necessarily coincide with the center of the glyph. Compare the corrected placement of the accent in *gcommaaccent* (left) with the default one (right).

In the AE family, the characters were brought together from a few CM sources. Consider, e.g., *aer10*: *arrow left hook* (i.e., faked *ogonek*), *less*, and *greater* characters were taken from *cmmi10*; *bar*, *backslash*, *braceleft*, *braceright*, and *section*—from *cmmi10*; *sterling*—from *cmu10*. Some characters were drawn using rules, e.g., *visiblespace* (missing characters were marked by a rule having width and height equal to $\frac{1}{2}$ em), and some were assembled from a few components (*Aogonek*, *aogonek*, *Eogonek*, *eogonek*). We went even further: we “borrowed” characters *asciicircum* and *asciitilde* from *cmex10*; *mu*—from *cmmi10*; *dagger*, *daggerdbl*, and *paragraph*—from *cmsy10*.

It is debatable whether borrowing characters is acceptable. The *section* sign from *cmsy10* is certainly an alien in a sans serif font. Therefore, characters that seemed to us more important (*section*, *sterling*) were programmed from scratch. We used, of course, appropriate parameters from the CM driver files, but we did not follow Knuth’s recipe rigorously. This might have been done (see the comments on the *Euro* symbol below). We preferred, however, our shapes of glyphs. This may evoke some compatibility-related issues but, anyway, the full compatibility with CM, EC, and AE fonts cannot be achieved (see section “Compatibility issues”, p. 21).

Actually, some characters were borrowed not from CM fonts but from their PL counterparts (i.e., CM fonts equipped with Polish diacritical letters; note that the relevant METAFONT code from the PL fonts was incorporated into the EC sources). The acute and grave accents over capital and small letters in PL fonts diminutively differ, namely, accents over capital letters are flattened. The same approach we applied in LM fonts (see figure 5) which is consistent with EC and inconsistent with CM fonts.

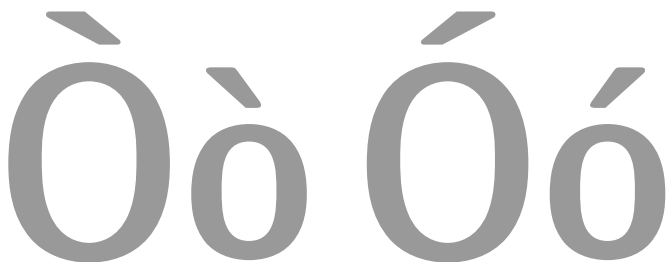


Figure 5: There are actually two acute accents in LM fonts: a flattened variant is used for capital letters. This idea was implemented in PL fonts and next in EC ones. Note that, in general, the flattening is neither a slanting nor a rotation.

Besides the accented, borrowed and newly programmed characters, a few glyphs had to be programmed as consistently as possible with the CM character set. A notable example is an *Euro* currency symbol. It looks as though it became so important recently that even Adobe assigned it a name beginning with a capital letter (cf. *dollar*, *yen*, *sterling*, etc., in *Adobe Glyph List For New Fonts* [1]). We attempted to exploit the METAFONT code for the letter *C* and—it worked (see figure 6).

The LM fonts contain also a few idiosyncratic symbols. We wanted, e.g., to have a ligature *f_k* in the repertoire of characters (see figure 7) because there are several words in Polish containing the digram ‘fk.’ They are less numerous than words with digrams ‘fi’ and ‘fl’ but more than words with trigrams ‘ffi’ and ‘ffl’ (which occur exclusively in words of foreign origin). The electronic *Collins English Dictionary* retrieved only three words containing the digram ‘fk’: Aufklärung (sic), Kafka, Kafkaesque. There are more candidates for nonstandard ligatures, e.g., ‘fb,’ ‘fh,’ ‘ffb,’ and ‘ffh.’ These groups of letters occur sporadically in English and German (they are absent from Polish). We are not in a position, however, to decide whether introducing the respective ligatures would make sense.

Compatibility issues

The answer to the question whether the LM fonts can serve as a replacement for CM or EC ones is obviously ‘no.’ First of all, the collection of fonts is different—LM is a subset of CM (except `lmssqb8` and a few oblique deriva-

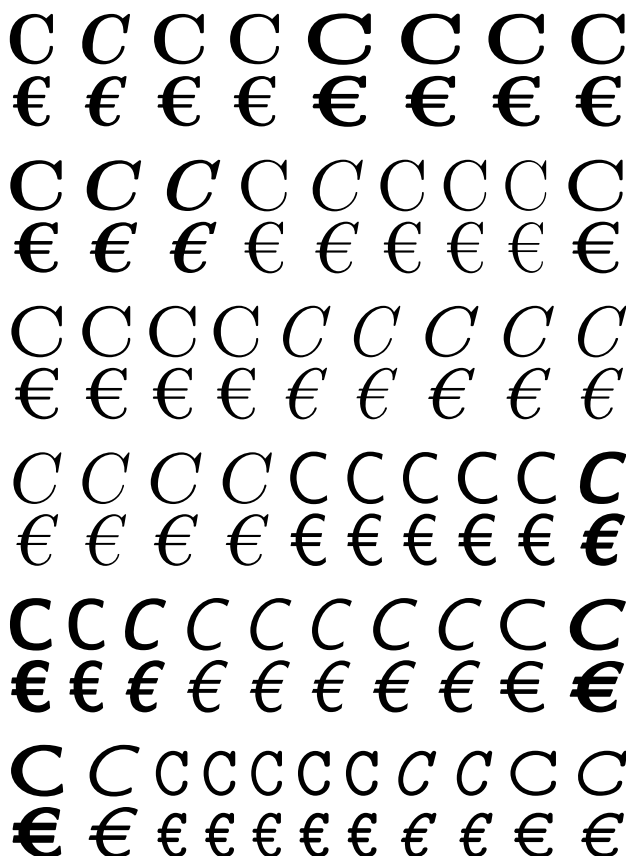


Figure 6: *Euro* symbols from the LM fonts; observe that an *Euro* symbol is narrower than the corresponding letter *C* (above), but that the stem sizes are preserved. Note, however, that there is no slot for an *Euro* symbol in the *Cork Encoding*.

tives), not speaking about EC. Therefore, not every text typeset with CM or EC fonts can be re-typeset using LM ones.

On the other hand, it should be noted that LM fonts are based on the data taken from CM driver files. Therefore, all relevant dimensions are (or at least should be) the same in LM and CM fonts within the accuracy of rounding errors. It is thus possible to use existing LM fonts as a replacement for CM

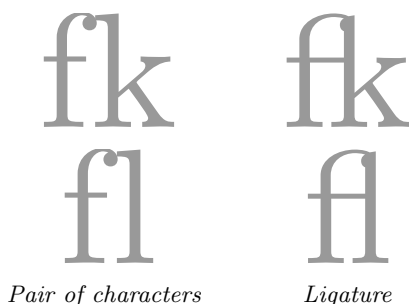


Figure 7: There are several words in the Polish language that contain the digram ‘fk’; therefore, we included the ligature *f_k* (top-right) in the LM character set for the sake of consistency with native CM ligatures, such as *fl* (bottom-right).

in dvips driver `psfonts.map` file—it suffices to prepare appropriate encoding (`*.enc`) files.

In order to reach this level of compatibility, we had to add two more characters, namely *arrowup* and *arrowdown* which, somewhat surprisingly, are present in `cmr5`, but not in other fonts in the `cmr*` series. At the moment, we resisted the temptation to include consequently a full quiver of other arrows. The main reason was that arrows are absent from the basic *Cork Encoding* (they appear only in the *Text Companion Encoding*—see, e.g., the file `dcdoc.tex` distributed with the EC sources); moreover, PostScript is anyway involved and therefore various transformations can easily be applied, if necessary. In future, however, we may change our opinion.

The METATYPE1 programs for the arrows are based on METAFONT sources excerpted from `sym.mf`. While adapting the code, we encountered a quandary which is a good example of a seemingly trivial yet embarrassing detail. It turned out, that the arrow programs produce questionable results for certain driver files, namely, sidebearings disappear. The arrow programs were perhaps never tested against all driver files. One can live with this, nevertheless, we decided to preserve minimal space at both sides—the result is certainly more palatable (see figure 8).

Another quandary of a similar kind is related to accents. For inexplicable reason, the caron accent in CM fonts is lowered in comparison to the remaining accents (see figure 9). We considered it a fault and decided to raise

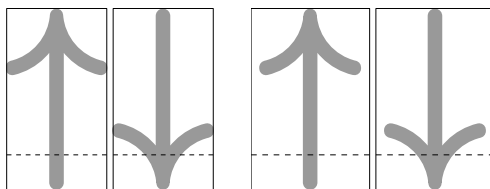


Figure 8: The METAFONT program for arrows (from `sym.mf`) would produce glyphs stripped of sidebearings for parameters from `cmssdc10` (left); arrows in LM fonts always have sidebearings (right).

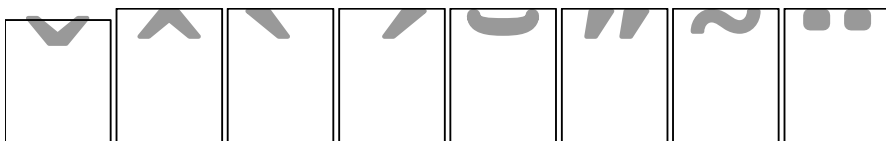


Figure 9: The caron alias hachek accent (the leftmost box) is slightly lowered in the CM fonts; in the LM fonts, all accents are aligned horizontally.

all carons appropriately. We relinquished thus the full compatibility between CM and LM families.

The game of names

Among many technicalities related to the representation of PostScript fonts, we would like to comment upon only one—the particularly upsetting problem of character names.

There exists a standard of glyph naming worked out by Adobe [1]: *Adobe Glyph List 2.0* (of 20th September 2002) and *Adobe Glyph List For New Fonts 1.0* (of 31st January 2003). Regretfully, the standard contains numerous entries that are at best dubious. We have already jeered at the name of the *Euro* symbol that singularly begins with a capital letter. But this is nothing. The excerpt from the *Adobe Glyph List For New Fonts* concerning characters with *commaaccent* is really astounding (see figure 10). Even more astounding is a part of this story pertaining to *Tcedilla* and *tcedilla*:

```

Gcommaaccent; LATIN CAPITAL LETTER G WITH CEDILLA
Kcommaaccent; LATIN CAPITAL LETTER K WITH CEDILLA
Lcommaaccent; LATIN CAPITAL LETTER L WITH CEDILLA
Ncommaaccent; LATIN CAPITAL LETTER N WITH CEDILLA
Rcommaaccent; LATIN CAPITAL LETTER R WITH CEDILLA
Scommaaccent; LATIN CAPITAL LETTER S WITH COMMA BELOW
gcommaaccent; LATIN SMALL LETTER G WITH CEDILLA
kcommaaccent; LATIN SMALL LETTER K WITH CEDILLA
lcommaaccent; LATIN SMALL LETTER L WITH CEDILLA
ncommaaccent; LATIN SMALL LETTER N WITH CEDILLA
rcommaaccent; LATIN SMALL LETTER R WITH CEDILLA
scommaaccent; LATIN SMALL LETTER S WITH COMMA BELOW

```

Figure 10: An excerpt from the up-to-date *Adobe Glyph List For New Fonts* [1]. How sweet. . .

- The version 1.1 of *Adobe Glyph List* mentioned the characters described as ‘T with cedilla’ and ‘t with cedilla’ and assigned them names *Tcommaaccent* and *tcommaaccent*, respectively; characters that could be described as ‘T with comma below’ or ‘t with comma below’ were just ignored.
- In the version 1.2 of the *Adobe Glyph List*, the names *Tcommaaccent* and *tcommaaccent* were assigned both to characters described as ‘T or t with cedilla’ and ‘T or t with comma below’.
- The up-to-date *Adobe Glyph List For New Fonts* says that one of the most recent changes was “renaming *tcommaaccent* to *tcedilla* and *Tcommaaccent* to *Tcedilla*.”

To untangle the “commaaccent story” a little bit, we would like to quote a more reliable opinion from Michael Everson’s web site devoted to European alphabets [5]:

- Concerning Latvian: “The [accented] characters g, k, l, n, r, G, K, L, N, and R must always be drawn with a *comma below*, although these characters are identified in ISO standards as *letters with cedilla*. Note particularly the reverse comma accent used with the *latin small letter g with cedilla*.” (Cf. figure 4.)

- Concerning Romanian: “Note that Romanian uses the characters *s with comma below* and *t with comma below*. In inferior Romanian typography, the glyphs for these characters are sometimes drawn with *cedillas*, but it is strongly recommended to avoid this practice.”

There were more pitfalls of this kind, not as ridiculous as the case of the *commaaccent*, but sufficiently confusing to make this part of the job arduous.

Given such a state of the art, we decided to copy some glyphs under different names—just in case. We repeated, e.g., the glyphs *scommaaccent*, *tcommaaccent*, *Scommaaccent*, and *Tcommaaccent* under the names *scedilla*, *tcedilla*, *Scedilla*, and *Tcedilla*, respectively. Altogether, there are approximately 10 duplicated characters per 400-character font.

Note that the duplication of glyphs does *not* lead to an enormous inflation of the size of font files because of a very efficient subroutine packing mechanism (cf. section “Another viewpoint”, p. 13). Actually, a duplicated character increases the size of a font by 30–40 bytes. This means that 10 duplicated characters would increase a font size by less than 1 percent, as the average size of an LM font (`pfb`) is 60 KB.

Beware of your friends

The basic tools we used (`METAPOST`, `tftopl`, `vftovp`, `awk`, `Tlutils`) worked nearly infallibly. Only once we met a truly intricate problem. It was a bug persistently offered by our friend, `METAPOST`.

One of the important operations in the process of font generation is determining the orientation of a path: anticlockwise-oriented paths are used for filling and clockwise-oriented—for unfilling. There is a function `turningnumber` in `METAFONT` and `METAPOST` that returns `+1` and `-1` for anticlockwise-oriented and clockwise-oriented paths, respectively. In `METAFONT` it works correctly, in `METAPOST`—unfortunately it does not. The bug manifests its presence even in such trivial cases as the following code:

```
path p;
p=(0,10)..controls (5,10) and (10,5)
..(10,0)..controls (10,-5) and (5,-10)
..(0,-10)..controls (-5,-10) and (-10,-5)
..(-10,0)..controls (-10,5) and (-5,10)
..cycle;
```

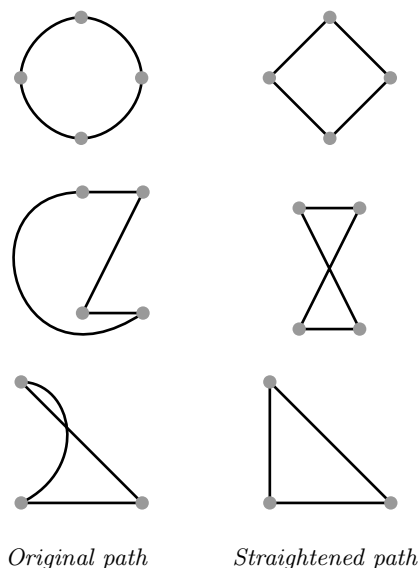


Figure 11: The operation of straightening of a path typically does not change the orientation of a path (top); in general, however, this may happen—middle and bottom pictures show how a non-zero turning number can be changed to zero and vice versa. The latter situations, fortunately, are rather unlikely in fonts.

This nearly circular 4-node path is evidently clockwise-oriented. Nevertheless, METAPOST maintains that `turningnumber p = 0`.

We did not analyse the METAPOST source code as we were not going to fix the bug, but circumventing it was crucial. The only method that proved to work was the “straightening” of a path prior to the application of the `turningnumber` function; in other words, each Bézier segment of a path was changed to a straight line and then the `turningnumber` function was applied to a modified path. It works so far, although the method is not general (see figure 11) and, moreover, frequently used straightening slows down the process of the generation of fonts.

Encodings

In olden days, there was a one-to-one correspondence between a *font name* and the name of a *font metric file* (**tfm**). This is not possible any longer. If there are more characters in a font than 256, like in CM-SUPER or LM, one has to select a subset of characters and to assign codes to every character. Even not knowing the precise results of combinatorial analysis, one may fancy how many such encoding may coexist. It seems that there is no choice—*metric files must not use the same name as the basic font*, otherwise a mess is bound to ensue.

One may think of a distinguished (main) encoding that would inherit the basic name, but we would rather equate all encodings. At present, we supply four encodings in the official distribution of the LM fonts: the *Cork Encoding* (also called T1), the *Text Companion Encoding* (TS1), the *T_EX'n'ANSI Encoding* (LY1) and the *QX Encoding*.

The first three encodings do not need explanation; the last one is actually a “double” encoding, i.e., there is a fixed collection of characters and two numberings—one to be used with T_EX and one to be used with GUI systems [12]. The *QX Encoding* was worked out a few years ago by the members of the Polish T_EX Users Group GUST as a difficult compromise between needs and abilities. In a nutshell: the *QX Encoding* for T_EX is a variant of the *Cork Encoding* with a few characters exchanged (e.g., *gbreve*, *Gbreve*, *uring*, and *Uring* are replaced by Lithuanian *iogonek*, *Iogonek*, *uogonek*, and *Uogonek*, respectively); the *QX Encoding* for GUI systems is a variant of the Code Page 1250 (and also includes Lithuanian characters with ogonek).

Recall that the complete list of the LM font names is shown in figure 2. The respective **tfm** file names are derived by adding the prefix **cork-**, **ts1-** (“textcompanion” seems lengthy), **texnansi-**, and **qx-** for the *Cork Encoding*, *Text Companion Encoding*, the *T_EX'n'ANSI Encoding*, and the *QX Encoding*, respectively. For example, the metric file for **lmr10** with the *Cork Encoding* has the name **cork-lmr10.tfm**.

This protocol is admittedly immature. Nevertheless, we do insist on recommending either this naming scheme or a similar one as a guideline for T_EX users at least as long as T_EX is not capable of handling multi-byte character codes.

Availability

And another tiny detail: the LM fonts are freely available on CTAN in the directory `/tex-archive/fonts/ps-type1/lm`.

Concluding remarks

We would like to emphasize once again that our aim was not only to provide a new family of fonts, but to provide it with METATYPE1 sources that can be maintained—adjusted, augmented, improved, etc. While it is rather difficult to write a font program from scratch, it is pretty simple to modify sources, e.g., as we have mentioned, adding accented letters is straightforward.

As concerns our plans regarding the LM family, we would like to enhance fonts: to extend the repertoire of characters (first of all by the *Text Companion* for the EC fonts), to improve kerning, hinting and shapes of certain glyphs, and, last but not least, to provide OpenType versions of the LM fonts for XP trailblazers. We consider, moreover, converting a few more CM programs from METAFONT to METATYPE1, as we would like to dismiss eventually the borrowed characters (see section “Tuning and augmenting. . .”, p. 20).

Before bringing the curtain down, we would like to draw the reader’s attention to a weak point of our approach: the CM parameterization is lost. The METATYPE1 sources can be enhanced, but they cannot be used for producing, say, light or condensed versions of sanserif fonts. An experiment with the programming of the *Euro* symbol and the arrows has shown that converting METAFONT sources to METATYPE1 ones without losing the parameterization is, in general, possible but rather time-consuming. It is an open question whether such a venture, being extremely attractive, is reasonable.

Acknowledgements

The project was supported by European T_EX Users Groups, in particular by the German-speaking T_EX Users Group DANTE e.V., the French-speaking T_EX Users Group GUTenberg, and the Dutch-speaking T_EX Users Group NTG—very many thanks. We are also grateful to Volker RW Schaa and Stefan Sokołowski for their valuable comments concerning the draft version of the paper.

References

- [1] *Adobe Solutions Network: Unicode and Glyph Names*,
<http://partners.adobe.com/asn/developer/type/unicodegn.html>
- [2] *Adobe Type 1 Font Format*. Addison-Wesley, 1990,
http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF
- [3] Włodzimierz Bzyl, *The Tao of Fonts*.
Proceedings of TUG 2002, 4th–7th September, 2002, Trivandrum,
India, TUGboat 23 (1), 2002, p. 27.
- [4] Lars Engbrechtsen, *AE fonts*,
<http://www.tug.org/tex-archive/fonts/ae/>
- [5] Michael Everson, *The Alphabets of Europe* (ver. 3.0)
<http://www.evertype.com/alphabets/>
- [6] Michael Ferguson,
Report on multilingual activities,
TUGboat 11 (4), November 1990, p. 514.
- [7] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk,
Antykwa Półtawskiego: A Parameterized Outline Font.
Proceedings of EuroT_EX 1999, 20th–24th September, 1999, Heidelberg,
Germany, pp. 109–141.
- [8] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk,
*METATYPE1: A METAPOST-based Engine for Generating Type 1
Fonts*. Proceedings of EuroT_EX 2001, 23rd–27th September, 2001,
Kerkrade, the Netherlands, pp. 111–119,
<http://www.ntg.nl/eurotex/metatyp1.pdf> and
<http://www.ntg.nl/eurotex/JackowskiMT.pdf>
- [9] Richard J. Kinch,
MetaFog: Converting METAFONT Shapes to Contours.
TUGboat 16 (3), pp. 233–243, 1995.
- [10] Han-Wen Nienhuys, *MFTrace—Scalable Fonts for METAFONT*,
<http://www.cs.uu.nl/~hanwen/mftrace/>
- [11] John Plaice and Yannis Haralambous, *Omega System*,
<http://sourceforge.net/projects/omega-system/>

- [12] *QX encoding tables for T_EX and for window systems*,
<http://www.gust.org.pl/fonty/qx-table1.html>,
<http://www.gust.org.pl/fonty/qx-table2.html>
- [13] John Sauter, *Building Computer Modern Fonts*,
 TUGboat 7 (3), October 1986, p. 151.
- [14] Péter Szabó, *T_EXtrace*,
<http://www.inf.bme.hu/~pts/texttrace/>
- [15] Vladimir Volovich, *CM-super Font Package*,
<ftp://ftp.vsu.ru/pub/tex/font-packs/cm-super/>
- [16] Martin Weber, *Autotrace*, <http://autotrace.sourceforge.net/>
- [17] George Williams, *PfaEdit—a PostScript Font Editor*,
<http://pfaedit.sourceforge.net/>

Appendix: The contents of the Latin Modern family of fonts, ver. 0.86

Note that some characters do not occur in all fonts, e.g. there are no *f*-ligatures in the typewriter fonts. Actually, there are five classes of charsets:

1. The basic class (516 glyphs); this class consists of `lmb10`, `lmb010`, `lmbx10`, `lmbx12`, `lmbx5`, `lmbx6`, `lmbx7`, `lmbx8`, `lmbx9`, `lmbxi10`, `lmbxo10`, `lmr10`, `lmr12`, `lmr17`, `lmr5`, `lmr6`, `lmr7`, `lmr8`, `lmr9`, `lmri10`, `lmri12`, `lmri7`, `lmri8`, `lmri9`, `lmro10`, `lmro12`, `lmro8`, `lmro9`, `lmss10`, `lmss12`, `lmss17`, `lmss8`, `lmss9`, `lmssbo10`, `lmssbx10`, `lmssdc10`, `lmssdo10`, `lmssso10`, `lmssso12`, `lmssso17`, `lmssso8`, `lmssso9`, `lmvtt10`, and `lmvtto10`.
2. The class ‘ssq’ (527 glyphs); besides the characters present in the basic class, it contains *varI*, *varIacute*, *varIcircumflex*, *varIdieresis*, *varIdotaccent*, *varIgrave*, *varIJ*, *varImacron*, *varIogonek*, *varItilde*, and *varIvardieresis*. The following fonts belong to this family: `lmssq8`, `lmssqbo8`, `lmssqbx8`, and `lmssqo8` (cf. also figure 3 and the relevant comments in section “The LM family of fonts. . .”, p. 18).
3. The class ‘typewriter’ (501 glyphs); the following glyphs are missing in comparison with the basic class: *f_k*, *ff*, *ffi*, *ffl*, *fi*, *fl*, *Germandbls*, *IJ*, *ij*, *permyriad*, *servicemark*, *suppress*, *trademark*, *varcopyright*, and *varregistered*. The class consists of `lmtt10`, `lmtt12`, `lmtt8`, `lmtt9`, `lmtti10`, and `lmttto10`.

4. The class containing only `lmcsc10` and `lmcsc010` (508 glyphs); the following glyphs are missing in comparison with the basic class: *dquoteright*, *f_k*, *ff*, *ffi*, *ffl*, *fi*, *fl*, and *tquoteright*.
5. The class containing only `lmtcsc10` (499 glyphs); the set of missing characters is as in class 3 plus *dquoteright* and *tquoteright*.

Tipps und Tricks: Mal anders herum – *excludeonly*

Rolf Niepraschk

Erstellt man mit $\text{\LaTeX}$ ein umfangreiches Dokument, sollte man von der Möglichkeit Gebrauch machen, das Dokument in Teildokumente aufzuteilen. Man erreicht dadurch eine bessere Übersicht und ist nicht gezwungen, bei jedem Übersetzungslauf den gesamten Quelltext zu bearbeiten. Die einzelnen Kapitel speichert man dazu in eigene Dateien, die per `include`-Anweisung dem Hauptdokument bekannt gemacht werden. Im Dokumentenkopf können mit Hilfe der Anweisung `includeonly` die aktuell gültigen Kapitel ausgewählt werden, ohne dass die Seitennummerierung und die Verweise gestört werden. Das Hauptdokument könnte beispielsweise wie im Folgenden skizzenhaft gezeigt aussehen:

```
\documentclass[a4paper]{scrbook}
\includeonly{theorie}
\title{Die neue Physik} \author{Hugo Wunderlich}
\begin{document}
  \frontmatter
  \maketitle \tableofcontents
  \mainmatter
  \include{einleitung}
  \include{theorie}
  \include{experiment}
  \include{auswertung}
  \backmatter
  \include{anhang}
\end{document}
```

Einzig der Inhalt der Datei `theorie.tex` wird in der Ausgabe wirksam. Bis hier handelt es sich um eine weitgehend bekannte Vorgehensweise.

Gelegentlich würde man sich wünschen, dass statt der zu aktivierenden Teildokumente besser die zu unterdrückenden anzugeben sein sollten. Ein Grund dafür könnte sein, dass man alle bereits fertigen Abschnitte eines umfangreichen Dokuments jederzeit komplett zusammenstellen möchte, wobei die noch in Arbeit befindlichen zwar bezüglich der Seitenzählung berücksichtigt werden sollen, jedoch ohne dass ihr noch fehlerhafter Inhalt erscheint. In derartigen Fällen könnte eine Anweisung `excludeonly` die Übersichtlichkeit erhöhen. Das $\text{\LaTeX}$ -Paket `excludeonly` [1] bietet eine solche Anweisung sowie die zugehörige Logik. Wollte man im oben gezeigten Beispiel erreichen, dass der Inhalt des gesamten Dokuments mit Ausnahme der in `theorie.tex` enthaltenen Teile in die Ausgabe gelangt, müsste man `\includeonly{theorie}` durch `\excludeonly{theorie}` ersetzen sowie im Dokumentenkopf

```
\usepackage{excludeonly}
```

ergänzen. Auch eine Kombination aus `excludeonly`- und `includeonly`-Anweisungen ist möglich, wobei dann nur die von beiden Anweisungen erlaubten Teildokumente berücksichtigt werden. Verwendet man die Paketoption „only“ bei der `usepackage`-Angabe, so werden dagegen alle `includeonly`-Anweisungen unwirksam.

Literatur

- [1] Dan Luecking und Donald Arseneau: *Excludeonly Package*; März 2003; CTAN: `tex-archive/macros/latex/contrib/supported/ltxmisc/excludeonly.sty`.

Erstellen von Schaltbildern mit *pst-circ*

Herbert Voß

Mit diesem Artikel wird die Beschreibung der Teilpakete, die alle unter dem Synonym `pstricks` zusammengefasst werden, mit einem Paket zur Darstellung von Schaltbildern fortgesetzt. Derartige Abbildungen gehören zu den Standardthemen in Schule und Universität und können daher für einige

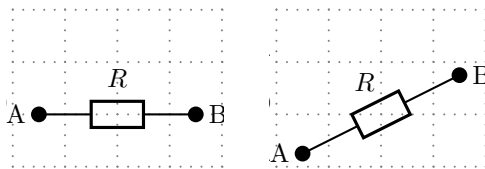


Abbildung 1: Prinzip zur Darstellung eines Objekts

interessant sein, insbesondere wenn es gilt, für Veröffentlichungen oder Arbeitsblätter auf einfache Art und Weise Ersatzschaltbilder zu erstellen ohne auf Vektorzeichenprogramme zugreifen zu müssen.

Das Prinzip

pst-circ baut im Prinzip auf **pst-node** auf, indem es zwischen vorgegebene Bezugspunkte (Knoten) ein grafisches Objekt setzt (Abbildung 1), wobei die Ausrichtung des Objekts dabei sekundär ist. Wenn keine Notwendigkeit besteht, explizit ein Koordinatenpaar als Knoten zu definieren, so kann grundsätzlich auch die normale $(x|y)$ -Form verwendet werden.

Die Beschriftung bezieht sich immer auf die horizontale Achse, kann jedoch durch eigene Parameter ebenfalls gedreht werden. Das Programmlisting 1 zeigt das Prinzip, wobei hier und für alle folgenden Beispiele das Zeichnen des Koordinatengitters und der Knotenpunkte nicht mit angegeben wird.

Listing 1: Prinzip zur Darstellung eines Objekts

```

1 \begin{pspicture}(-2,-1)(2,1.5)
2   \pnode(-1.5,0){A}
3   \pnode(1.5,0){B}
4   \resistor(A)(B){R$}
5 \end{pspicture}\hspace{0.5cm}
6 \begin{pspicture}(-2,-1)(2,1.5)
7   \pnode(-1.5,-0.75)
8   \pnode(1.5,0.75){B}
9   \resistor(A)(B){R$}
10 \end{pspicture}
```

Die Objekte

Die folgenden Tabellen 1 bis 3 zeigen eine Zusammenstellung der möglichen Objekte, wobei sie sich auf die derzeit aktuelle Version 1.1b bezieht. Formal kann folgende Unterscheidung vorgenommen werden:

Dipol benötigt zwei Knotenpunkte

Multidipol benötigt zwei Knotenpunkte, besteht aber aus mehreren Dipolen

Tripol benötigt drei Knotenpunkte

Quadrupol benötigt vier Knotenpunkte

In den Tabellen wird zur besseren Übersicht jedes Objekt ohne mögliche Optionen dargestellt, auf die dann im weiteren Verlauf dieses Artikels eingegangen wird. Der Aufruf eines Makros geschieht in der Regel durch

`\<Objektname>(<Knoten1>)(<Knoten2>) ... {<Bezeichner>}`

Das letzte Argument (Bezeichner oder Label) kann auch leer bleiben. Einige der Tripole verfügen nicht über dieses Argument, so dass eine Bezeichnung selbst vorgenommen werden muss.

Dipole

Dipole, im technischen Sprachgebrauch auch als Zweipole bezeichnet, stellen den größten Anteil an verfügbaren Objekten. Bis auf `wire` können alle Dipole mit einem Bezeichner (Label) versehen werden.

Tabelle 1: Liste der vordefinierten Schaltsymbole für Dipole

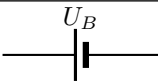
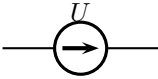
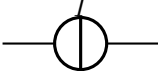
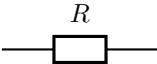
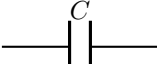
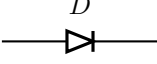
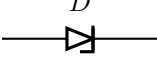
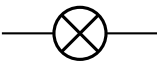
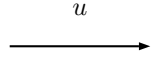
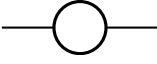
Bezeichnung	Makroname	Grafik
Batterie	<code>battery</code>	
Spannungsquelle	<code>Ucc</code>	
Stromquelle	<code>Icc</code>	

Tabelle 1: Liste der vordefinierten Schaltsymbole für Dipole (Fortsetzung)

Bezeichnung	Makroname	Grafik
Widerstand	<code>resistor</code>	
Kondensator	<code>capacitor</code>	
Spule	<code>coil</code>	
Diode	<code>diode</code>	
Z-Diode (Zener-Diode)	<code>Zener</code>	
LED	<code>LED</code>	
Lampe	<code>lamp</code>	
Schalter	<code>switch</code>	
Verbindung	<code>wire</code>	
Pfeillinie	<code>tension</code>	
Kreis	<code>circledipole</code>	

Das Symbol `circledipole` kann insbesondere für die Darstellung von Strom- und Spannungsquellen verwendet werden, die abweichend vom Batteriesymbol sind. Mit der Option `labeloffset=0` erreicht man, dass das Label genau

```

1 \begin{pspicture}(-2.75,-0.5)(2.75,1)
2   \pnode(-2.75,0){A}
3   \pnode(2.75,0){B}
4   \multipole(A)(B)%
5     \coil{$L$}%
6     \resistor{$R_L$}.%!! the dot
      finishes the multidipol !!
7 \end{pspicture}

```

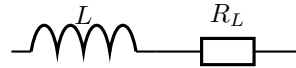


Abbildung 2: Definition eines Multidipols

zentriert eingefügt wird, wie es für das Anwendungsbeispiel in Abbildung 11 vorgenommen wurde.

Multidipole

Dies sind prinzipiell nichts weiter als eine lineare Verkettung von Dipolen, was eine Vereinfachung bei der Erstellung sein kann. So könnte man beispielsweise das Ersatzschaltbild einer Spule statt der Definition eines neuen Makros einfach als Multidipol definieren (Abbildung 2).

Zu beachten ist, dass der Punkt die Definition eines Multidipols beendet. Die Anzahl der Dipole ist formal nicht begrenzt, jedoch durch die endliche Zeichenbreite praktisch limitiert.

Tripole

In der Tabelle 2 sind zum besseren Verständnis jeweils die Namen der Knotenpunkte mit angegeben, hier als A, B und C bezeichnet. Damit wird die Bedeutung der Reihenfolge deutlich, denn alle Tripole werden mit ... (A) (B) (C) aufgerufen. Nur bei den Schalter besteht die Möglichkeit, einen Bezeichner (Label) anzugeben. Bei den anderen Tripolen muss dies über normale `uput` Befehle erfolgen.

Quadrupole

In der Tabelle 3 sind zum besseren Verständnis wieder die Namen der Knotenpunkte mit angegeben, hier als A, B, C und D bezeichnet. Damit wird die

Tabelle 2: Liste der vordefinierten Schaltsymbole für Tripole

Bezeichnung	Makroname	Grafik
Operationsverstärker	OA	
PNP-Transistor	transistor	
Schalter	Tswitch	

Bedeutung der Reihenfolge deutlich, denn alle Quadropole werden hier mit ... (A) (B) (C) (D) aufgerufen.

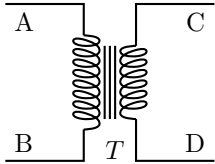
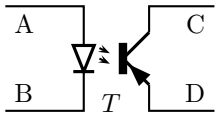
Optionen

Das Paket `pst-circ` selbst verfügt über keinerlei Paketoptionen. Dagegen gibt es eine erhebliche Zahl an verfügbaren Optionen für die Makros. Hierbei sind vor allem auch die Optionen für die farbliche Darstellung von Interesse, was für PDF-Ausgaben von Vorteil ist.

Strompfeile

Jedes Objekt kann in seinen Verbindungsleitungen mit einem Strompfeil versehen werden. Das Beispiel in Abbildung 3 zeigt eine Anwendung aller verfügbaren Optionen für die Kennzeichnung von Strompfeilen, wobei die Stan-

Tabelle 3: Liste der vordefinierten Schaltsymbole für Quadropole

Bezeichnung	Makroname	Grafik
Transformator	transformer	
Optokoppler	optoCoupler	

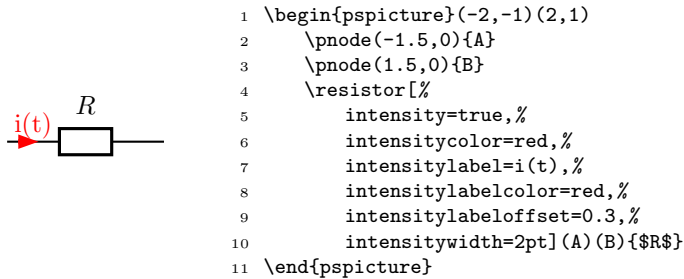
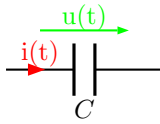


Abbildung 3: Möglichkeiten zur Darstellung von Strompfeilen

dardwerte für die Optionen der Dokumentation von `pst-circ` entnommen werden können.

Die Stromrichtung ist jeweils durch die Reihenfolge der Knoten beim Makroaufruf festgelegt. Der Strompfeil zeigt demnach von `Knoten1` nach `Knoten2`, was jedoch über die Option `directconvention=false` zusätzlich beeinflusst werden kann.



```

1 \begin{pspicture}(-2,-1)(2,1.5)
2   \pnode(-1.5,0){A}
3   \pnode(1.5,0){B}
4   \capacitor[%
5     labeloffset=-0.75,%
6     tension=true,%
7     tensioncolor=green,%
8     tensionoffset=0.75,%
9     tensionlabel=u(t),%
10    tensionlabelcolor=green,%
11    tensionlabeloffset=1,%
12    tensionwidth=\pslinewidth] (A) (B) {\$C\$}
13 \end{pspicture}

```

Abbildung 4: Möglichkeiten zur Darstellung von Spannungspfeilen

Spannungspfeile

Analog zu den Strompfeilen kann jedes Objekt parallel dazu mit einem Spannungspfeil versehen werden. Das Beispiel in Abbildung 4 zeigt wieder eine Anwendung aller verfügbaren Optionen für die Kennzeichnung von Spannungspfeilen, deren Standardwerte wieder der Dokumentation von *pst-circ* entnommen werden können. Das Listing zeigt allerdings nur die für die Spannung relevanten Optionen, die hier zusätzlich zu Abbildung 3 aufgenommen wurden.

Der Abbildung 4 kann ebenfalls entnommen werden, dass das Label für das Objekt, welches standardmäßig oberhalb erscheint, durch Modifikation des Offsets nach unten gesetzt wird. Weiterhin entspricht die Pfeilrichtung der Standardvorgabe, sie ist wie beim Strom von B nach A gerichtet, womit das Bauteil formal Energie aufnimmt. Dies lässt sich einfach mit der Option

`dipoleconvention=generator|receptor`

ändern, wobei `receptor` die Standardvorgabe ist. Beide Pfeile zusammen können ebenfalls über eine Option richtungsmäßig umgedreht werden:

`directconvention=false|true`

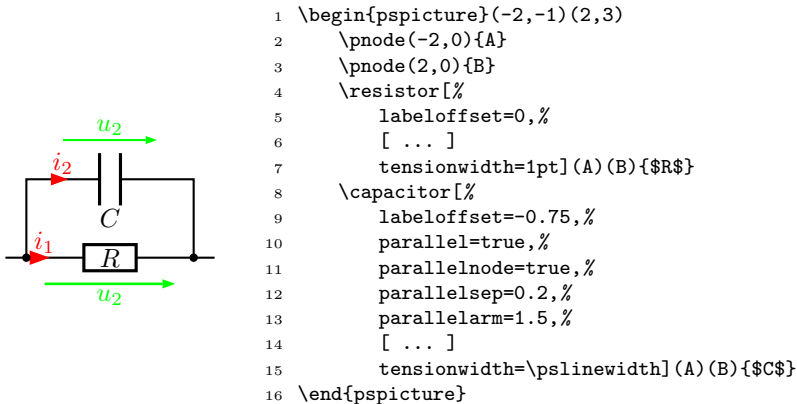


Abbildung 5: Parallelschaltung

Parallelschaltungen

Dieser Fall tritt sehr häufig auf, beispielsweise beim Ersatzschaltbild für einen Kondensator, sodass *pst-circ* hierfür extra einige Optionen aufweist. Abbildung 5 zeigt ein einfaches Beispiel mit sämtlichen Optionen, wobei wieder einige der vorhergehenden übernommen wurden, ohne dass sie im angegebenen Programmcode erscheinen.

Das Prinzip ist, dass beide Objekte dieselben Knotenpunkte haben, nur dass einer von beiden mit der Option **parallel** darüber oder darunter angeordnet wird. Letzteres ist mit einem negativen Wert für **parallelarm** möglich, beispielsweise -2 , dann wird dieses Objekt im Abstand von 2 Längeneinheiten nach unten gezeichnet. Damit sind auch Parallelschaltungen mit drei Objekten möglich.

Darstellungsformen

Insbesondere zwischen europäischen und amerikanischen Symbolen bestehen sehr häufig Unterschiede in der Darstellung. Fast alle können per Option berücksichtigt werden und sind in Tabelle 4 zusammengefasst.

Tabelle 4: Alternative Darstellungsformen

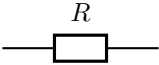
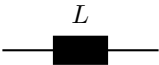
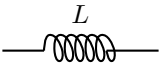
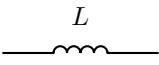
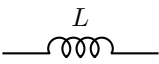
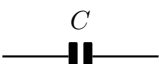
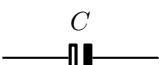
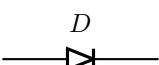
Makroname	Option	Grafik
resistor	–	
	dipolestyle=zigzag	
coil	–	
	dipolestyle=rectangle	
	dipolestyle=curved	
	dipolestyle=elektor	
	dipolestyle=elektorcurved	
capacitor	–	
	dipolestyle=chemical	
	dipolestyle=elektor	
	dipolestyle=elektorchemical	
diode	–	

Tabelle 4: Alternative Darstellungsformen (Fortsetzung)

Makroname	Option	Grafik
		<i>T</i>
	dipolestyle=thyristor	
		<i>GTO</i>
	dipolestyle=GTO	
		<i>Triac</i>
	dipolestyle=triac	

Regelbare Widerstände, Spulen und Kondensatoren

Mit der Option `variable=true` lassen sich die Elemente mit einem diagonalen Pfeil versehen, sodass sie als regelbar gekennzeichnet sind (Abbildung 6).

Transistoren

Ohne weitere Angaben wird der in Tabelle 1 dargestellte PNP-Transistor gezeichnet. Eine Anwendung möglicher Optionen zeigt Abbildung 7. Mit `transistorinvert=false|true` vertauscht man die Anschlüsse Emitter und Kollektor. Die übergeordnete Option `intensity=true` setzt automatisch alle drei Strompfeile für Basis/Emitter/Collector auf `true`. Die Farbe für Strompfeil und Label kann über die allgemeinen Optionen `intensitycolor=<Farbe>` und `intensitylabelcolor=<Farbe>` global gesteuert werden.

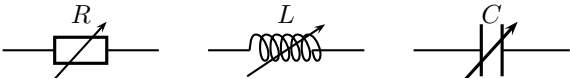
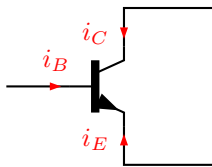


Abbildung 6: Regelbare Objekte



```

1 \begin{pspicture}(4,3)
2   \pnode(0,1.5){A}\uput[-45](A){A}
3   \pnode(4,3){B}\uput[-135](B){B}
4   \pnode(4,0){C}\uput[135](C){C}
5   \transistor[%
6     transistorcircle=false,%
7     transistortype=NPN,%
8     transistoribase=true,%
9     transistoricollector=true,%
10    transistoriemitter=true,%
11    transistoribaselabel={\red $i_B$},%
12    transistoricollectorlabel={$i_C$},%
13    transistoriemitterlabel={$i_E$},%
14    intensitycolor=red,%
15    intensitylabelcolor=red](A)(B)(C)
16 \end{pspicture}

```

Abbildung 7: Transistor-Optionen

Operationsverstärker

Tabelle 2 zeigt die Standarddarstellung für den idealen Operationsverstärker mit unendlich großer Verstärkung. Die Abbildung 8 zeigt eine Darstellung mit allen möglichen Optionen. Auch hier kann wieder zur Vereinfachung die übergeordnete Option **intensity** benutzt werden, wenn alle drei Strompfeile gezeichnet werden sollen.

Sehr häufig findet sich mittlerweile eine andere Darstellung für OPs, die in Abbildung 9 gezeigt ist und über die Option **tripolestyle=french** erreicht wird.

Die letzte Option **OAinvert** ermöglicht ähnlich wie beim Transistor die Vertauschung der beiden Eingänge.

Transformator

Als letztes Objekt soll der Transformator mit seinen Optionen behandelt werden, deren Auswirkung in Abbildung 10 zu erkennen ist. Die Markierung der Strompfeile kann wieder über die übergeordnete Option **intensity** gesetzt werden, wenn beide Pfeile gezeichnet werden sollen. Weiterhin kann das in Deutschland übliche Symbol mit rechteckigen Spulen benutzt werden. Dazu

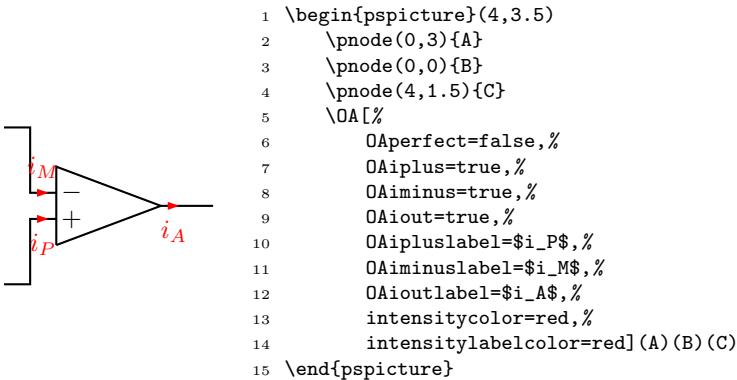


Abbildung 8: Operationsverstärker-Optionen

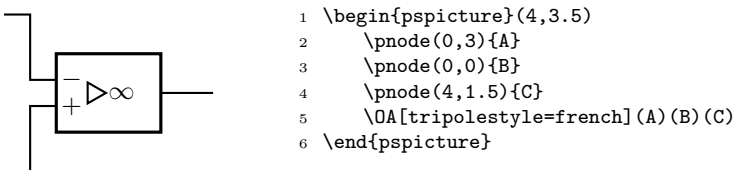


Abbildung 9: Operationsverstärker in anderer Darstellung

wird die Option `dipolestyle=rectangle` verwendet, auch wenn es sich hier um einen Quadrupol handelt.

Anwendungsbeispiel

Mit Sicherheit wird man mit *pst-circ* nicht umfangreiche Schaltungen entwerfen. Dennoch lassen sich insbesondere kleinere Schaltbilder oder Ersatzschaltbilder leicht erstellen, wenn man während der Erstellung mit einem Koordinatengitter arbeitet, welches mit *psgrid* einfach erstellt werden kann. Das in Abbildung 11 dargestellte Ersatzschaltbild eines Gleichstromstellers wurde mit der in Listing 2 angegebenen Befehlsfolge erstellt.

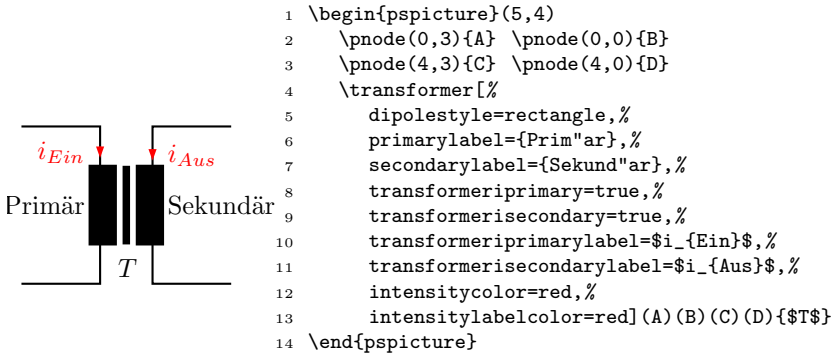
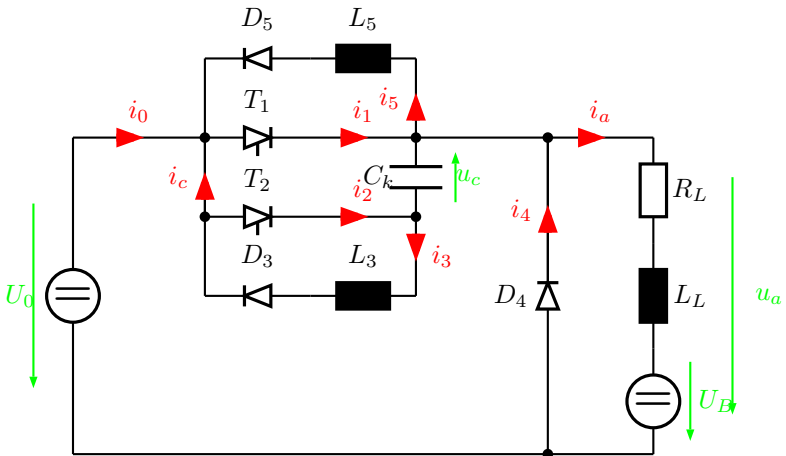


Abbildung 10: Transformator-Optionen

Abbildung 11: Anwendungsbeispiel für *pst-circ*

Listing 2: Befehlsfolge für Abbildung 11

```

1 \psset{intensitycolor=red,intensitylabelcolor=red,%
2   tensioncolor=green,tensionlabelcolor=green,%
3   intensitywidth=3pt}
4 \begin{pspicture}(-1.25,0)(13.5,9)
5   \psgrid[griddots=5,gridlabels=7pt,subgriddiv=0]
6   \circledipole[
7     tension,%
8     tensionlabel=$U_0$,%
9     tensionoffset=0.75,%
10    labeloffset=0](0,0)(0,6){\LARGE\textbf{=}}
11   \wire[intensity,intensitylabel=$i_0$](0,6)(2.5,6)
12   \diode[dipolestyle=thyristor](2.5,6)(4.5,6){$T_1$}
13   \wire[intensity,intensitylabel=$i_1$](4.5,6)(6.5,6)
14   \multidipole(6.5,7.5)(2.5,7.5)%
15     \coil[dipolestyle=rectangle,labeloffset=-0.75]{$L_5$}%
16     \diode[labeloffset=-0.75]{$D_5$}.
17   \wire[intensity,intensitylabel=$i_5$](6.5,6)(6.5,7.5)
18   \wire(2.5,7.5)(2.5,3)
19   \wire[intensity,intensitylabel=$i_c$](2.5,4.5)(2.5,6)
20   \qdisk(2.5,6){2pt}\qdisk(6.5,6){2pt}
21   \diode[dipolestyle=thyristor](2.5,4.5)(4.5,4.5){$T_2$}
22   \wire[intensity,intensitylabel=$i_2$](4.5,4.5)(6.5,4.5)
23   \capacitor[tension,tensionlabel=$u_c$,%
24     tensionoffset=-0.75,tensionlabeloffset=-1](6.5,4.5)(6.5,6){$C_k$}
25   \qdisk(2.5,4.5){2pt}\qdisk(6.5,4.5){2pt}
26   \wire[intensity,intensitylabel=$i_3$](6.5,4.5)(6.5,3)
27   \multidipole(6.5,3)(2.5,3)%
28     \coil[dipolestyle=rectangle,labeloffset=-0.75]{$L_3$}%
29     \diode[labeloffset=-0.75]{$D_3$}.
30   \wire(6.5,6)(9,6)\qdisk(9,6){2pt}
31   \diode(9,0)(9,6){$D_4$}
32   \wire[intensity,intensitylabel=$i_4$](9,3.25)(9,6)
33   \wire[intensity,intensitylabel=$i_a$](9,6)(11,6)
34   \multidipole(11,6)(11,0)%
35     \resistor{$R_L$}
36     \coil[dipolestyle=rectangle]{$L_L$}%
37     \circledipole[labeloffset=0,%
38       tension,tensionoffset=0.7,%
39       tensionlabel=$U_B$]{\LARGE\textbf{=}}.
40   \wire(0,0)(11,0)\qdisk(9,0){2pt}
41   \tension(12.5,5.5)(12.5,0.5){$u_a$}
42 \end{pspicture}

```

PDF-Ausgabe

pst-circ [1] baut wie bei allen **pstricks**-Paketen [7] üblich, vollständig auf PostScript [3] auf, kann somit auch nicht direkt mit pdf $\TeX$ benutzt werden. Für eine PDF-Ausgabe gibt es allerdings verschiedene Möglichkeiten:

- Verwendung von **pdftricks** [6], was aber sehr häufig wegen der Verwendung von dvips zu Problemen mit der „Bounding Box“ führen kann.
- Verwendung von **ps4pdf** [5], was allerdings eine Installation des L $\text{\AA}$ T $\text{\E}$ X Pakets **preview** [2] voraussetzt. Vergleiche dazu den entsprechenden Artikel in diesem Heft.
- Verwendung des für Linux und OS/2 freien Programms V $\text{\T}$ E $\text{\X}$ [4].
- Verwendung von **ps2pdf** mit der Konvertierungskette **dvi**→**ps**→**pdf**.

Die Methode mit **ps4pdf** bietet zudem die Möglichkeit, sämtliche mit **pstricks** generierten Abbildungen auch als einzelne PDF- oder EPS-Dateien abzuspeichern.

Zusammenfassung

Dieser Artikel sollte die Anwendung des Pakets **pst-circ** zeigen. Einiges blieb unerwähnt, kann jedoch in der Dokumentation zum Paket nachgelesen werden. Dies betrifft beispielsweise das Kreuzen von Verbindungen und Anwendungen von Schaltern. **pst-circ** ist eine echte Hilfe, wenn es um Schaltbilder geht, die eine gewisse Komplexität nicht überschreiten, wie sie fast immer für Ersatzschaltbilder gegeben ist.

Literatur

- [1] Christophe Jorssen und Herbert Voß: *pst-circ - PostScript macros for drawing electronic circuits*; CTAN:/graphics/pstricks/contrib/pst-circ/; 2003.
- [2] David Kastrup: *preview-latex*; CTAN:/support/preview-latex/; 2003.
- [3] Nikolai G. Kollock: *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*; IWT; Vaterstetten; 1989.

- [4] Micropress: *VT $\TeX$ /Linux*; <http://www.micropress-inc.com/linux/>; 2003.
- [5] Rolf Niepraschk: *ps4pdf*; CTAN:/macros/latex/contrib/ps4pdf/; 2003.
- [6] Herbert Voss: *PSTricks Support for pdf*; <http://www.pstricks.de/pdf/pdftricks.phtml>; 2002.
- [7] Timothy Van Zandt: *pstricks - PostScript macros for Generic TeX*; <http://www.tug.org/application/PSTricks>; 1993.

PDF und PostScript – das $\LaTeX$ -Paket *ps4pdf*

Rolf Niepraschk

Es wird ein Weg gezeigt, wie man mit relativ geringem Aufwand PostScript-Code innerhalb eines mit pdf $\LaTeX$ zu bearbeitenden Dokuments verwenden kann.

Einleitung

Vor einiger Zeit wurde in dieser Zeitschrift das $\LaTeX$ -Paket *preview* vorgestellt [3]. Der Artikel endete mit den Worten:

„... Mit dem Gezeigten sind die Möglichkeiten von *preview* lange noch nicht ausgeschöpft. Insbesondere kann es die Grundlage bilden für künftige $\LaTeX$ -Pakete, die die Grafikeinbindung noch mehr vereinfachen.“

Dieser Ankündigung folgend ist der Gegenstand der Betrachtung diesmal genau ein solches $\LaTeX$ -Paket, nämlich *ps4pdf* [4].

Der übliche Weg

Die meisten Grafiken können von T_EX nicht direkt verarbeitet werden. Stattdessen wird dieser T_EX-fremde Code intern innerhalb von so genannten `\special`-Anweisungen gekapselt. Einzig die Abmessung der Grafik muss während des Textsatzes bekannt sein. Durch Sichtbares wird der Inhalt der `\special`-Anweisungen erst vom Ausgabetreiber ersetzt. Im Falle von PostScript-Code bedeutet dieses Ersetzen einen großen Aufwand, da es sich bei PostScript um eine umfangreiche Programmiersprache handelt, deren Anweisungen – abgesehen von Spezialfällen – einen sehr leistungsfähigen Interpreter verlangen. Der in pdfT_EX eingebaute Ausgabetreiber ist nicht in der Lage, diese Aufgabe zu erfüllen. Nachfolgend wird davon ausgegangen, dass die Entscheidung für pdfL^AT_EX gefallen ist, auch wenn die Dokumente normalerweise „unverträglichen“ PostScript-Code enthalten. Gründe für die Wahl von pdfT_EX bzw. pdfL^AT_EX zur Erzeugung eines pdf-Dokuments können sein:

- Umbrechbare Hyperlinks
- Direkt ladbare Pixelgrafikformate
- Einfügen von Einzelseiten aus externen pdf-Dokumenten
- Möglichkeit des optischen Randausgleichs
- Möglichkeit der Font-Beeinflussung
- Sehr weitgehende Einflussnahme auf die erzeugte pdf-Datei
- Kontinuierliche Weiterentwicklung
- Freie Software

Im Einzelnen kann der eine oder andere Punkt von geringem Gewicht bei einem Vergleich mit anderen Verfahren sein. Das soll aber an dieser Stelle nicht näher diskutiert werden.

Um den in einer EPS-Grafik enthaltenen PostScript-Code in einem pdfL^AT_EX-Dokument verwenden zu können, geht man üblicherweise wie folgt vor:

- Wandlung der EPS-Grafik ins PDF-Format:
 - ▷ `epstopdf --outfile=grafik.pdf grafik.eps`

- Bearbeitung des Dokuments (Mit `\includegraphics{grafik}` wird die Grafik eingefügt):

▷ `pdflatex dokument.tex`

Dieser Weg ist überschaubar. Bei sehr vielen Grafiken ist der Aufwand allerdings nicht unerheblich, wobei natürlich eine Automatisierung per Skripten möglich ist.

Erheblich komplizierter ist die Behandlung von PostScript-Code jedoch dann, wenn er nicht in Form einer Grafikdatei vorliegt, sondern direkt im Quelltext des Dokuments benutzt wird, wie das beispielsweise bei PSTricks-basierten Paketen [6] der Fall ist. Fasst man sämtliche derartigen Grafiken in einem Hilfsdokument zusammen, lässt sich unter Zuhilfenahme von dvips (Option -E) versuchen, EPS-Dateien zu erzeugen, die wiederum wie oben beschrieben behandelt werden können. Ganz offensichtlich ist dies aber ein sehr unbequemer Weg, der – noch wichtiger – auch nicht immer fehlerfrei ist.

Ein anderer Weg

Die im vorigen Abschnitt angedeutete Idee eines Hilfsdokuments, welches ausschließlich die Grafiken enthält, ist besonders deshalb sehr aufwändig zu realisieren, weil man es selbst erzeugen und auf etliche Besonderheiten achten muss. Um später Einzelgrafiken zu bekommen, muss genau eine Grafik pro Seite angeordnet sein. Will man, dass Referenzen auf das Hauptdokument Bestandteil von Grafiken werden, erfordert dies weitere Überlegungen. Auch ist darauf zu achten, dass das Hilfsdokument dieselben Schriftarten und -größen wie das Hauptdokument benutzt. Es wäre daher eine große Erleichterung, wenn man das eigentliche Dokument zusätzlich auch zum Erzeugen der gewünschten Grafiken nutzen könnte.

Um dies zu erreichen, ist das L^AT_EX-Paket `preview` von großem Nutzen. Wie bereits in [3] beschrieben, kann `preview` beliebige Dokumentbestandteile systematisch extrahieren und in einer DVI-Datei aufsammeln. Die Forderung „ein Objekt pro Seite“ ist bereits erfüllt. Sorgt man nun noch dafür, dass es sich bei den „Objekten“ um die problematischen PostScript-haltigen Grafiken handelt, ist eine anwenderfreundliche Lösung des Gesamtproblems bereits in greifbare Nähe gerückt.

Der Ablauf mit *ps4pdf*

Das L^AT_EX-Paket *ps4pdf* setzt diese Überlegungen um. Alle Anweisungen, die direkt oder indirekt PostScript-Code beinhalten, müssen Parameter der von dem Paket definierten Anweisung `\PSforPDF` werden. Ein in dieser Art ergänztes Dokument wird nun folgendermaßen bearbeitet (es soll hier den Namen `beispiel.tex` haben):

- Erzeugung einer Grafikcontainer-Datei `beispiel-pics.pdf` („pics“ als Teil des Dateinames¹ ist zwingend!):
 - ▷ `latex beispiel.tex` (preview ist in Aktion!)
 - ▷ `dvips -o beispiel-pics.ps beispiel.dvi`
 - ▷ `ps2pdf beispiel-pics.ps beispiel-pics.pdf`
- Erzeugung des eigentlichen Dokuments:
 - ▷ `pdflatex beispiel.tex`

Während des pdfL^AT_EX-Laufs ändert sich die Bedeutung der Anweisung `\PSforPDF` in der Weise, dass automatisch intern per

```
\includegraphics[page=n]{beispiel-pics.pdf}
```

die zugehörige PDF-Grafik eingefügt wird. Ein interner Zähler liefert den Wert *n*, sodass gewährleistet ist, dass jeweils die richtige Seite bzw. Grafik aus der Grafikcontainer-Datei verwendet wird. Ändert sich Inhalt oder Anzahl der Grafiken, muss sie neu erzeugt werden.

Ein konkretes Beispiel

Der nachfolgende Quelltext führt zu dem in den Abbildungen gezeigten Ergebnis. Es gelingt, wie man sieht, auch für diese mit pdfL^AT_EX hergestellte Zeitschrift ohne Probleme. Die eingefügte Datei `4-10-8.inl` aus [5] enthält eine PStricks-Grafik aus dem Buch „The L^AT_EX Graphics Companion“:

```
% beispiel.tex
\documentclass{scrartcl}
\usepackage{ps4pdf,calc,graphicx,psfrag}
\PSforPDF{%    Vorbereitende Definitionen für die PStricks-Grafik.
```

¹ Einen anderen Namen für die Grafikcontainer-Datei kann man mit der Anweisung `\containerName{neuer Name}` definieren.

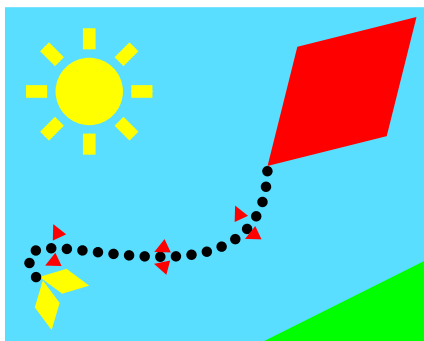


Abbildung 1: Eine PSTricks-Grafik

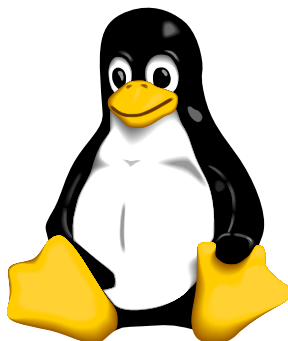


Abbildung 2: Eine EPS-Datei

```

\usepackage{pstcol,pst-node,pst-tree,multido}
\definecolor{lightblue}{cmyk}{0.65,0.13,0,0}
\begin{document}
\begin{figure}
\begin{minipage}[b]{.5\textwidth-.5\columnsep}%
\PSforPDF{\resizebox{\linewidth}{!}{\input{4-10-8.inl}}}
\caption{Eine PSTricks-Grafik}\label{fig:beispiel1}
\end{minipage}\hfill
\begin{minipage}[b]{.5\textwidth-.5\columnsep}\centering
\PSforPDF{\includegraphics[width=.66\linewidth]{penguin}}
\caption{Eine EPS-Datei}\label{fig:beispiel2}
\end{minipage}
\end{figure}
\begin{figure}
\hfill\fbbox{%
\PSforPDF{\includegraphics[width=.3\textwidth]{escher}}}
\hfill\fbbox{%
\PSforPDF[trim=-10mm 0mm 12mm 2mm]{\Large%
\psfrag{T}{c}[t]{Oben}\psfrag{L}{r}[r]{Links}%
\psfrag{R}{l}[l]{Rechts}%
\includegraphics[width=.3\textwidth]{escher}}}
\hfill\mbox{}
\caption{Der \texttt{trim}-Parameter bei einer mit
\textsf{psfrag} modifizierten EPS-Datei}\label{fig:beispiel3}%
\end{figure}
\end{document}

```

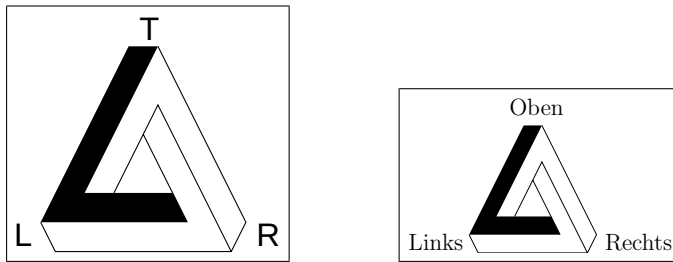


Abbildung 3: Der `trim`-Parameter bei einer mit `psfrag` modifizierten EPS-Datei

Die Paketooptionen `inactive` und `draft`

Ein Dokument, welches von `\PSforPDF`-Anweisungen Gebrauch macht, sollte trotzdem nicht unbedingt auf pdf_TE_X angewiesen sein. Aus diesem Grunde gibt es die Paketooption `inactive`. Sie verhindert die spezielle Behandlung der `\PSforPDF`-Anweisungen durch `preview`. Der „klassische“ Ablauf sieht dann folgendermaßen aus, wenn man auf die Modifikation des Quelltextes verzichtet:

- `latex '\PassOptionsToPackage{inactive}{ps4pdf}' \`
`'\input{beispiel.tex}'`
- `dvips -o beispiel.ps beispiel.dvi`

Die Notwendigkeit und Anzahl der Hochkommas ist vom verwendeten Kommandozeileninterpreter abhängig. Der T_EX-Compiler V_TE_X kann, anders als hier dargestellt, direkt benutzt werden, da er automatisch die Option `inactive` erzwingt.

Ähnlich wie beim L^AT_EX-Paket `graphicx` kann die Paketooption `draft` zum Unterdrücken der Inhalte von `\PSforPDF`-Anweisungen eingesetzt werden. Es werden dann nur Rahmen in der jeweiligen Größe gezeichnet. `draft` kann auch als Parameter beim Aufruf von `\PSforPDF` gezielt einzelne Ausgaben verhindern.

Der `trim`-Parameter

Die Größe der beiden ersten Grafiken entspricht genau der Größe, die auch T_EX bekannt ist (durch die vom PSTricks-Code definierte Box bzw. durch

die Angabe in der `BoundingBox`-Zeile der EPS-Datei). In manchen Fällen enthält eine PostScript-Grafik jedoch Anweisungen, die außerhalb des Bereiches, für den T_EX Platz reserviert hat, Zeichenoperationen ausführen. Um solche Grafiken trotzdem korrekt platzieren zu können, kann der `\PSforPDF`-Anweisung der optionale Parameter „trim“ übergeben werden. Er wirkt in ähnlicher Weise wie bei `\includegraphics` aus dem Paket `graphicx`, wenn er dort zusammen mit „clip“ verwendet wird [1]. Die Angabe

```
\PSforPDF[trim=-3mm 0mm 0mm 4mm]{...}
```

würde beispielsweise einen links um 3 mm und oben um 4 mm größeren Bereich in die Grafikcontainer-Datei einfügen. Beim späteren pdfL^AT_EX-Lauf wird die gesamte Grafik derart skaliert, dass der T_EX bereits bekannte Platz optimal ausgefüllt wird. Abbildung 3 zeigt die Anwendung des `trim`-Parameters an einem konkreten Beispiel, bei dem Textbestandteile der Grafik mittels `psfrag`-Anweisungen [2] ersetzt wurden. Ohne ihn würden sich die neuen breiteren Texte teilweise außerhalb der Position der T_EX-Box befinden und dort nicht zu sehen sein.

Weitere Möglichkeiten

Gelegentlich kann der Wunsch aufkommen, die grafischen Inhalte der `\PSforPDF`-Anweisungen auch außerhalb der T_EX-Welt zu verwenden, beispielsweise innerhalb eines Grafikprogramms. Auch für diesen Zweck kann die Grafikcontainer-Datei als Basis dienen. Soll die Grafik mit dem „freundlichen Drachen“ in Form einer EPS-Datei zur Verfügung gestellt werden, kann dies folgendermaßen geschehen:

```
pdftops -f 1 -l 1 -eps beispiel-pics.pdf drachen.eps
```

(Die Nummer der betreffenden Seite im Grafikcontainer muss bekannt sein – es ist die erste Grafik, daher hier die Angabe „1“ für die Seitenauswahl.)

Gedacht ist die Anweisung `\PSforPDF` eigentlich dazu, PostScript-Code als Parameter aufzunehmen. Es können aber genauso gut beliebige andere Bestandteile eines L^AT_EX-Dokuments sein. Eine umfangreiche mathematische Formel kann somit auch in den Grafikcontainer gelangen und, wie eben beschrieben, zu einer EPS-Grafik gewandelt werden. Auf diese Weise lässt sich die mit T_EX mögliche gute Qualität des Satzes von mathematischen Formeln in weniger kompetente Programme exportieren.

Fazit

Das L^AT_EX-Paket `ps4pdf` bietet eine weitere Möglichkeit, auf PostScript basierenden Code innerhalb eines pdfL^AT_EX-Dokuments nutzen zu können, ohne diesen aufwändig einzeln ins PDF-Format wandeln zu müssen. Die dabei erzeugte Grafikcontainer-Datei bietet darüber hinaus einen einfachen Weg des Exports in andere Anwendungen.

Literatur

- [1] D. P. Carlisle: *Packages in the ‘graphics’ bundle*; Jan. 1999; CTAN: `macros/latex/required/graphics/grfguide.tex`.
- [2] Michael C. Grant und David Carlisle: *The PSFrag system, version 3*; Apr. 1998; CTAN: `macros/latex/contrib/supported/psfrag/pfpguide.tex`.
- [3] Rolf Niepraschk: *Anwendungen des L^AT_EX-Pakets preview*; *Die T_EXnische Komödie*; 1/2003, S. 60–65; Febr. 2003.
- [4] Rolf Niepraschk: *The ps4pdf Package*; Mai 2003; CTAN: `macros/latex/contrib/ps4pdf/`.
- [5] Sebastian Rahtz: *Most of the PSTricks examples of The L^AT_EX Graphics Companion*; CTAN: `graphics/pstricks/doc/lgc/`.
- [6] Timothy Van Zandt: *PSTricks*; März 1993; CTAN: `graphics/pstricks/`.

T_EX-Beiprogramm

Typograf der Zeit – Hans Peter Willberg ist tot¹

Die Hamburger „Zeit“ erhob ihn einmal zu Deutschlands „Typografiepapst“. Sein Rang war damit treffend bezeichnet, im Übrigen aber hatte Hans Peter Willberg nichts Päpstliches an sich. In Mainz, wo er Buchgestaltung lehrte, ermunterte er die Studenten zu Kritik und Diskussion. „Glauben Sie mir kein Wort!“, mahnte er sie selbstironisch. Für ihn gab es nur eine einzige typografische Doktrin: „Ein Buch muss funktionieren.“ Damit allerdings war es ihm bitterernst. Ein Roman, der schön gemacht aussieht, aber schlecht lesbar ist – dergleichen fand keine Gnade vor seinen Augen.

Dutzendweise sind Willbergs gestalterische Arbeiten im Wettbewerb „Die schönsten Bücher“ von der Stiftung Buchkunst in Frankfurt prämiert worden, die Stuttgarter Ausgabe der Werke und Briefe von Christian Morgenstern etwa und natürlich seine eigenen Werke, allen voran „Buchkunst im Wandel“, ein Klassiker der typografischen Zunft. Denn Willberg war nicht nur ein exzellenter Buchgestalter, sondern auch ein vorzüglicher Schriftsteller, der sein als trocken geltendes Lehrfach lebendig, ja packend behandelte. HPW machte nicht alles mit, aber war für alles offen. Zusammen mit seiner Frau und Berufskollegin Brigitte, mit der ihn eine produktive Symbiose verband, konnte er für ein evangelisches Gesangbuch eine geradezu antimodernistische, verzopfte Noten- und Schrifttype wählen – und zugleich die expressive Wucht eines von HAP Grieshaber gestalteten Bandes oder die Künstlerbücher des Avantgardisten Dieter Roth preisen.

Schulbildend zu wirken hat Willberg nie interessiert. Umso mehr Schüler hat er gewonnen. Seine Arbeiten sind überall präsent: der Umschlag der Reclam-Universalbibliothek zum Beispiel oder das Signet des Manesse-Verlags, der

¹ Stuttgarter Zeitung 06.06.2003; Abdruck mit freundlicher Genehmigung der Stuttgarter Zeitung.

Löwe, der gediegen und historisch aussieht, aber doch mit moderner Raster-typografie gestaltet und erst zwanzig Jahre alt ist. Viele kennen diese Schöpfungen, ihren Schöpfer kennen sie nicht – Typografenschicksal. Am 29. Mai hat, wie erst jetzt bekannt wurde, Hans Peter Willberg im Alter von 72 Jahren den Kampf gegen ein schweres Leiden verloren.

gün

Rezensionen

„ $\text{\LaTeX}$ für Dummies“ Christian Baun

Hilmar Preuße

Die Bücher der „Dummies“-Reihe hat man schon in den Regalen der großen Ladenketten stehen sehen, sich aber nie eines gekauft. Kürzlich wurde „ $\text{\LaTeX}$ für Dummies“ DANTE e.V. zugesandt, welches hier rezensiert werden soll. Alle Fehler dieses Buchs aufzuzählen würde den Rahmen dieser Kurzkritik sprengen, weshalb nur die gravierendsten genannt werden. Schon im Inhaltsverzeichnis fallen URLs der Form `http://hostname/ user` auf. Auf der dazugehörigen Seite im Buch sind diese aber korrekt geschrieben.

Die im Kapitel 1 beschriebene $\text{\TeX}$ -History liest sich etwas merkwürdig: Die Programmierung von $\text{\TeX}$ 78 wurde 1977 begonnen, 10 Jahre später wurde es released, aber schon 1984 war $\text{\LaTeX}$ fertig? Ein echter Insider-Tipp ist, dass $\text{\TeX}$ zu 1/3 Compiler, 1/3 Interpreter und 1/3 Textverarbeitung ist. Leider wird vermieden, dies genauer zu erläutern. Auf Seite 37 stößt man dann auf den ersten fachlichen Fehler: Weder erzeugt METAFONT `pk`-Files, noch kann man aus ihnen die `tfm`-Files gewinnen. Auf Seite 40 geht es dann los, leider etwas verquer: Der Autor erklärt zweimal die Grundstruktur eines minimalen $\text{\LaTeX}$ -Dokuments und geht mittendrin darauf ein, wie man mit dem fertigen `dvi`-File umgeht. Zwei Seiten später folgt kommentarlos das erste komplexe Dokument mit Bildern, Formeln und Schmankerln, wie `\usepackage{umlaut}`.

Im Kapitel 2 lässt sich der Autor über feste und elastische Maße, Änderung von Längenwerten, Akzente, Sonderzeichen, Ligaturen und Gruppenbildung aus. Das Paket (n)german wird nur erwähnt und auch verraten, wie man $\mathcal{M}\mathcal{S}\text{\TeX}$ setzt, falls man einmal ein Buch über $\text{\LaTeX}$ schreiben will. Dass man dazu ein zusätzliches Paket benötigt, wird nicht erwähnt. Ganz nebenbei wird das Kommando `\usepackage` eingeführt, wobei weder globale noch Paketoptionen erklärt werden.

In Kapitel 3 erfahren wir etwas zum Thema Dokumentklassen und Klassenoptionen. Der $\text{LaTeX}2.09$ -Kompatibilitätsmodus wird kurz vorgestellt und zuviel(?) über die Änderung des Seitenlayouts philosophiert. Danach erfährt der geneigte Leser, dass die „Stilfiles“ in $\text{LaTeX}2.09$ die Endung `.sty` haben, in $\text{LaTeX}2_{\epsilon}$ hingegen die Endung `.cls`: „Bei $\text{LaTeX}2_{\epsilon}$ -Distributionen sind die `.sty`-Dateien meistens fast vollständig leer und erhalten nur einen Verweis auf die gleichnamigen `.cls`-Dateien.“ Auf Seite 87 kommen wir endlich zur Dokumentstrukturierung – Inhaltsverzeichnis, Gliederungen, Titelseite usw. Es darf getrost bezweifelt werden, dass der Autor die vollständigen Auswirkungen des Befehls `\section{\textsl{Geneigte Überschrift}}` getestet hat. Insgesamt gibt es in diesem Kapitel sehr viele Einzelheiten zum Thema, die zu diesem Zeitpunkt noch uninteressant sind und auch nicht ausreichend oder unverständlich erläutert werden, zumal sich das Buch an Anfänger wendet.

Das Kapitel 4 heißt „Formatierungshilfen“. Hier wird `\\[...]` als Kommando zum Zeilenumbruch eingeführt – von `\par` ist nirgendwo die Rede. Es werden lustig plain- TeX -Kommandos, wie `\bigskip` eingestreut. Weiter geht es in Kapitel 5 mit Texthervorhebungen. Dass der Autor anscheinend den Unterschied zwischen `\textsl` und `\textit` nicht sieht, ist zu verschmerzen. Ob das Fontheading in $\text{LaTeX}2.09$ sehr viel einfacher war, weil man die Font-Attribute nicht kombinieren konnte, sei dahingestellt. Wenn man `textcomp` erwähnt und dessen Symbolliste abdruckt, sollte man auch die „Comprehensive LaTeX Symbol List“ erwähnen. Auch ist interessant, dass man `dropping.dtx` „ausführen“ muss, um `dropping.sty` zu generieren. Weiterhin gibt es einen Abschnitt „Fußnoten“. Wie man mittels `\fnsymbol` das Fußnotensymbol ändert, habe ich auch nach fünfmal Lesen nicht verstanden. Die Hervorhebung der semantischen Auszeichnung gegenüber der generischen ist unzureichend bzw. nicht vorhanden.

Kapitel 6 behandelt dann den Tabellensatz. Warum die Angabe `p{breite}` in der Tabellenpräambel eine „Fuddelei“ sein soll, die zu vermeiden ist, wird nicht verraten. Auch nicht, dass man damit mehrzeilige Zellen erzeugen kann. Weiterhin wird extensiv auf `tabularx` und farbige Tabellen eingegangen. Andere Tabellenpakete, wie beispielsweise `longtable` und `supertabular` sucht man vergeblich. Das nächste Kapitel dreht sich um die Bildeinbindung. Ob man den Anfänger gleich mit der `\picture`-Umgebung, `\put` und `\qbezier` traktieren sollte, ist fragwürdig. Auf die Möglichkeiten des Pakets `graphicx` wird nur unzureichend eingegangen und es wird auch keine weiterführende Literatur genannt.

Weiter geht's im Kapitel 8 mit Boxen in allen Variationen. Strings, die öfter vorkommen, in eine Box zu packen und sie mit `\usebox` wiederzuverwenden, anstatt dies mittels `\newcommand` zu tun, ist sicher nicht zu empfehlen.

Kapitel 9 ist mit „Der Mathematikmodus“ betitelt und erstaunlich fehlerarm. Zu bemängeln gibt es eigentlich nur, dass `\vec` zur Vektorendarstellung besser geeignet ist als `\overrightarrow` und dass der Autor den T_EXnischen Unterschied in der Verwendung von `\mid` und `|` anscheinend nicht kennt. Es ist zwar nicht Thema dieses Buches, dass Formeln, die man beispielhaft setzt, fachlich korrekt sein müssen, man könnte aber trotzdem darauf achten.

Im Kapitel 10 dreht sich alles um Chemie. Der Hauptteil besteht aus der Beschreibung des Paketes `chemtex`. Die Erwähnung des Paketes mit Verweis auf die Dokumentation hätte gereicht, da es den Anfänger zunächst kaum interessieren dürfte.

Kapitel 11 behandelt dann PDF. Hier finden sich Weisheiten, wie: „PDF ist (...) im Grunde nichts anderes als ein objektorientiertes Postscript.“ Einige Tools zum Anzeigen von PS und PDF sowie zum Generieren von PDF (`ps2pdf` und `dvipdf`) werden vorgestellt. `pdfLATEX` und `hyperref` sind einen längeren Abschnitt wert. Nach ein paar Worten zum Thema Konvertierung EPS→PDF folgt auch schon ein größeres Beispiel mit `\ifpdf`-Konstrukten ohne genauere Erläuterung. Die Schriftproblematik (Postscript-Type1 vs. Postscript-Type3) hat der Autor offensichtlich nicht verstanden, denn der zugehörige Absatz strotzt vor Fehlern.

Kapitel 12 enthält eine relativ ausführliche Beschreibung der beiden Klassen `g-brief` und `dinbrief`, die man mit einem Verweis auf die Dokumentation auch hätte einsparen können. Dasselbe gilt für Kapitel 15 und 16, die die Vorstellung einer Sammlung von (Nichtbrot)Schriften und die Beschreibung des Paketes `chess` enthalten.

Das nächste Kapitel zum Thema Literaturverzeichnisse und `BibTEX` enthält keine gravierenden Fehler. Danach kommen wir auch schon zur „Oberliga“, also grundlegende Dinge, wie Querverweise. Außerdem gibt es Befehle zum Thema Textfarbe, das Paket `draftcopy`, das Unterteilen des T_EX-Files in einzelne Teile, sowie drei Seiten zum Thema Indexerstellung (sehr unvollständig) bunt gemixt.

Kapitel 17 enthält drei Beispiele und Vorlagen in denen sich der Autor produziert. Diese sind entweder zu komplex für Anfänger oder elementar aus der Dokumentation erstellbar. Im vorletzten Kapitel gibt es die „zehn wichtigs-

ten Dinge, die Sie (...) beachten sollten“. Diese Hinweise sind sehr allgemein gehalten und eher verwirrend als wirklich zielführend.

Eines der hilfreicheren Kapitel ist Anhang B, der mit „Fehlerbehandlung“ betitelt ist und ein paar Hinweise gibt, wie mit (L^A)T_EX-Fehlern und -Warnungen umgegangen werden könnte. Die Pakete **syntonly** und **tracefmt** werden kurz vorgestellt.

Dem Buch liegt eine CD-ROM bei, die eine „Kopie“ einer T_EX-Live zu sein scheint. Leider wusste der DANTE-Vorstand nichts von einer Autorisierung der Kopie. Die CD-ROM wurde um die Beispiel-Listings aus dem Buch angereichert und unter Windows gemastert. Dabei sind sämtliche Softlinks, die auf der CD-ROM waren, zerstört worden, was die Linux-Programme teilweise unbrauchbar macht. Erst im Anhang A wird, neben den Hinweisen auf ein paar Tools aus dem L^AT_EX-Umfeld, darauf eingegangen, welche T_EX-Implementierungen genau auf der CD-ROM enthalten sind. Nur unzureichend wird beschrieben, wo man eine passende Distribution für sein Betriebssystem erhält. Da sich dieses Buch explizit an Anfänger wendet, von denen nur erwartet wird, dass sie ein Betriebssystem haben, für das eine T_EX-Implementierung existiert, könnte wenigstens kurz auf die Installations-Oberfläche der T_EX-Live und die Bedienung des installierten Programmpakets eingegangen werden.

Eine besondere Spezialität des Autors besteht im Weglassen von \ an einigen Stellen und anscheinend wahllosen Einfügen dieses Zeichens an anderen Stellen, wo es fehlplatziert ist. Auch sonst sind Tipp- und Flüchtigkeitsfehler an der Tagesordnung. Der Befehls-Index zeigt nicht immer an die Stelle im Buch, an der auf dieses Kommando eingegangen wird. Man hat das Gefühl, der Autor hat die Tipps, die er so gibt, nicht immer selber ausprobiert – nicht alle Beispiel-Listings sind fehlerfrei. Der Fachkorrektor ist offenbar nicht seiner Rolle als solcher gerecht geworden. Ab und zu wird auch auf ein Paket, einen Befehl oder eine Umgebung verwiesen, auf die im ganzen Buch nicht eingegangen wird, sowie werden Pakete erwähnt, die inzwischen obsolet sind und durch offizielle Pakete vom L^AT_EX3-Team ersetzt wurden. Moderne Entwicklungen, wie die EC-Fonts oder KomaScript sucht man vergeblich. Im ganzen Buch ist nirgendwo von Gleitobjekten die Rede. Weder die Umgebung **table** wie auch die Umgebung **figure** werden auch nur erwähnt. Sehr mangelhaft ist der Verweis auf weiterführende Literatur: Weder die T_EX-FAQ von DANTE e.V. noch spezielle (paketspezifische) Dokumentationen oder weitere einführende Dokumente werden erwähnt. Stattdessen sind die URLs der

Homepage von Donald Knuth und zweier Anleitungen, wie Truetype-Fonts in $\text{\LaTeX}$ verwendet werden, im letzten Kapitel aufgeführt. Wenigstens findet sich hier die Homepage vom DANTE e.V. Die für diese Buchreihe typische Symbolik im Fließtext erleichtert das Lesen nicht unbedingt.

Der Schreiber dieser Rezension ist der Meinung, dass der Buchautor der Verbreitung von $\text{\LaTeX}$ eher geschadet als genützt hat.

Christian Baun
 $\text{\LaTeX}$ für Dummies
Mitp-Verlag, 2001
ISBN 3-826-63035-1
24,95 €

Leserbriefe

Zu Torsten Brongers Artikel „Einfaches Setzen von Texten in Fraktur mittels `blackletter1`“

Moriz Hoffmann-Axthelm

Zu Torsten Brongers interessantem und verdienstvollem Artikel „Einfaches Setzen von Texten in Fraktur mittels `blackletter1`“ (Die $\TeX$ nische Komödie 2/2003) möchte ich einige Anmerkungen beisteuern.

Auszeichnung

Es erscheint mir problematisch, Textur (Gotisch), Schwabacher und Fraktur in einem Text zu mischen, zumal, wenn es zur Auszeichnung, analog zu kursivem und fettem Antiquasatz, geschieht. Zwar ist auch die Kursive als eigenständige Schrift (ungefähr zur selben Zeit wie die Fraktur) entstanden, aber das interessiert nur historisch, sie ist traditionell die für Auszeichnung zuständige Schwester ihrer Grundschrift. Textur, Schwabacher und Fraktur waren und sind dagegen verschiedene Welten.

Ich kann zwar auf keine wissenschaftlichen Kenntnisse bezüglich des Satzes mit gebrochenen Schriften zurückgreifen wie offenbar Torsten Bronger, aber meiner Erfahrung als Leser (ich habe einige Meter in gebrochenen Schriften gesetzter Bücher konsumiert) wie Setzer zufolge ist es ebenso üblich wie angezeigt, es möglichst bei einer Schrift zu belassen.

Fetten und halbfetten Satz gibt es auch bei gebrochenen Schriften (für Überschriften sowie zur Auszeichnung für konsultatives Lesen), und Sperrung als Form der Auszeichnung hat vielleicht einzig bei gebrochenen Schriften seine Berechtigung (auf jeden Fall aber Tradition). Für Überschriften gibt es ansonsten die Mittel Schriftgröße und Durchschuss – was braucht man mehr?

Le fontaineux tout ce que l'ay
 du en ces ouvrages à l'enlaine
 de l'ant. Siege Apollonique.

Abbildung 1: Antiqua von Claude Garamond um 1540

Wer partout mischen will, könnte es auch mit Antiqua-Schriften in gebrochenem Satz versuchen.

Langes und rundes s

Die Differenzierung von langem und rundem s in gebrochenen Schriften ist zusätzlicher Reichtum, über den auch die Antiquaschriften der Renaissance (von denen wir noch heute ‚leben‘) verfügten, wie Originalschriftbeispiele von Nicolaus Jenson, Aldus Manutius oder Claude Garamond (siehe Abbildung 1) beweisen.

Historisch geht die Differenzierung in langes und rundes s mindestens bis zur karolingischen Minuskel zurück, Anfang 9. Jh. bis Ende 11. Jh. im gesamten westlichen Abendland die vorherrschende Schrift und ‚Mutter‘ von runden (Antiqua-) wie gebrochenen Schriften (Abbildung 2).

A a b b c d d e e
 f g g h i l l m m
 n o p p q q r r t t
 u v x y c t f t a

Abbildung 2: Karolingische Minuskel, vergrößert

Þ þ

Ð ð

(a) Þorn in gebrochener Schrift

(b) eð mit spezifischen Serifen

Abbildung 3: Isländische Sonderzeichen

Reichtum deshalb, weil die Differenzierung nicht nur das Schriftbild belebt, ‚natürlicher‘ macht, sondern auch – wie auch Ligaturen – durch ihre Kontextabhängigkeit die Lesbarkeit fördert: Die Wortfuge in *aussetzen* ist eben nicht nur eine mögliche Trennstelle, sondern für Aussprache (gerade im Deutschen: einerseits mit hartem, andererseits mit weichem s) und Verständnis essenziell.

Umgekehrt machen die Konsonantenhäufungen, die uns die Rechtschreibreform beschert (*dass* statt *daß*, *Schlusssatz* statt *Schlußsatz*), das Schriftbild starrer und unübersichtlicher. Folglich ist die Empfehlung Fritz Jörns, für *dass* *daß* zu setzen, nicht nur naheliegend und vernünftig, sondern auch historisch legitimiert, ist doch unser ß von Haus aus eine Ligatur eben aus langem und rundem s. Entsprechend erscheint mir diese Schreibweise nicht nur nicht ungewohnt, geschweige denn schwer lesbar, vielmehr sehe ich bei schnellem Lesen hier unwillkürlich ein – ß.

Und so empfehle ich allen Gegnern der neuen Rechtschreibung, die zu deren Verwendung gezwungen werden, das lange s in den Antiquaschriften wieder einzuführen und entsprechend den Regeln für die gebrochenen Schriften zu verwenden. Da mir dazu die T_EXnischen Kenntnisse fehlen, bleibt mir hier nur die Bitte: Kann das jemand ermöglichen? Die eigentliche Arbeit hat Torsten Bronger doch bereits geleistet, oder?

Sonderzeichen

Die isländischen Sonderzeichen Þorn (Þ, þ) und eð (Ð, ð) dürfen auf keinen Fall mit Th/th wiedergegeben werden, nur weil sie im Englischen diese Entwicklung im Laufe der Jahrhunderte genommen haben – was weiß der gemeine Isländer davon, der dann seine Texte nicht mehr lesen kann? Für das eð empfehle ich jedenfalls die auf Island gerne benutzte spezifische Gestaltung der Serifen. Zu den polnischen und tschechischen Sonderzeichen sollen sich die Slawisten äußern.

Spielplan

Termine

- 11.–12. 9. 2003** Journée GUTenberg 2003
Annual Meeting of the French TeX Users' Group
Chateau Pierreclos, Mâcon, Frankreich
Kontakt: Secrétariat GUTenberg
- 13. 11. 2003** 32. Mitgliederversammlung NTG
Arnhem, Niederlande
Kontakt: NTG
- März 2004** 30. Mitgliederversammlung und 15. Geburtstag von DANTE e.V.
Technische Universität Darmstadt, Darmstadt
Kontakt: DANTE e.V.
- 28. 4.–2. 5. 2004** BachoT_EX 2004
10th annual meeting of the Polish T_EX Users' group
GUST
Bachotek, Brodnica Lake District, Poland
Kontakt: Jolanta Szelatyńska
- 30. 8.–3. 9. 2004** TUG 2004
International Conference on T_EX, XML and Digital Typography
Democritus University of Thrace
Xanthi, Greece
<http://obelix.ee.duth.gr/tug2004/>
Kontakt: Apostolos Syropoulos

Stammtische

In verschiedenen Städten im Einzugsbereich von DANTE e.V. finden regelmäßig Treffen von T_EX-Anwendern statt, die für jeden offen sind. Im WWW gibt es aktuelle Informationen unter <http://www.dante.de/events/stammtische/>.

Berlin

Rolf Niepraschk
Tel.: 030/3 48 13 16
niepraschk@ptb.de
Hausprojekt K9
Kinzigstraße 9
10247 Berlin
Zweiter Donnerstag im Monat, 19.00 Uhr

Bremen

Martin Schröder
Tel.: 04 21/2 23 94 25
martin@oneiros.de
Wechselnder Ort
Erster Donnerstag im Monat, 18.30 Uhr

Chemnitz

Ralf König
Tel.: 03 41/4 11 58 00
ralf.koenig@s1998.tu-chemnitz.de
Universitätsteil 1, Straße der Nationen 62,
Raum 1/068
Dritter Mittwoch im Monat, 18.00 Uhr

Darmstadt

Karlheinz Geyer
karlheinz.geyer@LHSystems.com
Restaurant „Bölle“
Darmstadt, Böllenfalltor
Nieder-Ramstädter-Straße 251/Klappacher
Straße
Erster Freitag im Monat, ab 19.30 Uhr

Dresden

Carsten Vogel
lego@wh10.tu-dresden.de
Studentenwohnheim, Borsbergstraße 34,
Dresden, Ortsteil Striesen
ca. alle 8 Wochen, Donnerstag, 19.00 Uhr

Erlangen

Walter Schmidt, Peter Seitz
was@VR-Web.de,
Gaststätte „Erlanger Gärtla“
Marquardsenstraße 1

Dritter Dienstag im Monat, 19.00 Uhr

Freiburg

Heiko Oberdiek
Tel.: 07 61/4 34 05
oberdiek@uni-freiburg.de
Wechselnder Ort
Dritter Donnerstag im Monat, 19.30 Uhr

Hannover

Mark Heisterkamp
heisterkamp@rrzn.uni-hannover.de
Seminarraum RRZN
Schloßwender Straße 5
Zweiter Mittwoch von geraden Monaten,
18.30 Uhr

Heidelberg

Luzia Dietsche
Tel.: 062 21/54 45 27
luzia.dietsche@urz.uni-heidelberg.de
China-Restaurant „Palast“
Lessingstraße 36
Letzter Mittwoch im Monat, 20.00 Uhr

Karlsruhe

Klaus Braune
Tel.: 07 21/6 08 40 31
braune@rz.uni-karlsruhe.de
Universität Karlsruhe, Rechenzentrum
Zirkel 2, 3. OG, Raum 316
Erster Donnerstag im Monat, 19.30 Uhr

Köln

Bruno Hopp
b.hopp@lepkes-frings.de
Institut für Kristallographie
Zülpicher Straße 49b
Letzter Mittwoch im Monat, 19.30 Uhr

Konstanz

Matthias Weisgerber, Hraban Ramm
weisgerb@fmi.uni-konstanz.de,
hraban@f1ee.net
Restaurant Rheingold

*Spanierstraße 3
unregelmäßig*

München

Michael Niedermair

m.g.n@gmx.de

Gastwirtschaft „Rhætenhaus“

Luisenstraße 27

Erster Dienstag im Monat, 19.00 Uhr

Münster

Johannes Reese

reese@linguist.de

Gaststätte „Sabroso“

Mauritzstraße 19

nach Vereinbarung

Stuttgart

Bernd Raichle

bernd.raichle@gmx.de

Gaststätte „Alte Mira“

Büchsenstraße 24

Zweiter Dienstag im Monat, 19.30 Uhr

Wuppertal

Andreas Schrell

Tel.: 02 02/50 63 81

schrell@wupperonline.de

Restaurant Croatia „Haus Johannisberg“

Südstraße 10

an der Schwimmoper Wuppertal-Elberfeld

Zweiter Donnerstag im Monat, 19.30 Uhr

Zürich

Johannes Reese

reese@spw.unizh.ch

nach Vereinbarung

Adressen

DANTE, Deutschsprachige Anwendervereinigung T_EX e.V.
Postfach 10 18 40
69008 Heidelberg

Tel.: 0 62 21/2 97 66 (Mo, Mi–Fr, 10.00–12.00 Uhr)
Fax: 0 62 21/16 79 06
E-Mail: dante@dante.de

Konten: Volksbank Rhein-Neckar eG
BLZ 670 900 00
Kontonummer 2 310 007
IBAN DE67 6709 0000 0002 3100 07
SWIFT-BIC GENODE61MA2
Postbank Karlsruhe (Auslandsüberweisungen)
BLZ 660 100 75
Kontonummer 213 400 757
IBAN DE93 6601 0075 0213 4007 57
SWIFT-BIC PBNKDEFF

Präsidium

Präsident:	Volker RW Schaa	president@dante.de
Vizepräsident:	Klaus Höppner	vice-president@dante.de
Schatzmeister:	Tobias Sterzl	treasurer@dante.de
Schriftführer:	Günter Partosch	secretary@dante.de
Beisitzer:	Thomas Koch	
	Bernd Raichle	advisor@dante.de

Server

ftp: [ftp.dante.de](ftp://ftp.dante.de) [134.100.9.51]
E-Mail: ftpmail@dante.de
WWW: <http://www.dante.de/>

Autoren/Organisatoren

GUST secretary Univwersyteckie Centrum Komputeryzacji UMK ul. Gagarina 7 87-100 Toruń, Poland secretary@gust.org.pl	[67]	Rolf Niepraschk Persiusstr. 12 10245 Berlin niepraschk@ptb.de	[32, 49]
Secrétariat GUTenberg 2, rue des Boutons-d'or F-05000 Gap France secretariat@gutenberg.eu.org	[67]	Janusz M. Nowacki Foto-Alfa, Grudziadz Poland J.Nowacki@gust.org.pl	[10]
Klaus Höppner siehe Seite 70	[4]	Hilmar Preuße Max-Planck-Str. 2 64807 Dieburg hille42@web.de	[59]
Moriz Hoffmann-Axthelm Oranienstraße 19 10999 Berlin	[64]	Volker RW Schaa siehe Seite 70	[4]
Bogusław Jackowski BOP s.c., Gdańsk Poland B.Jackowski@gust.org.pl	[10]	Uwe Siart Aribonenstr. 1 81669 München	[7]
Gerd Neugebauer Im Lerchelsbühl 5 64521 Groß-Gerau gene@gerd-neugebauer.de	[3]	Apostolos Syropoulos 366, 28th October Str. GR-671 00 Xanthi, Hellas apostolo@ocean1.ee.duth.gr	[67]
		Herbert Voß Wasgenstr. 21 14129 Berlin voss@perce.de	[33]

Die T_EXnische Komödie

15. Jahrgang Heft 3/2003 September 2003

Impressum

Editorial

Hinter der Bühne

- 4 Grußwort
- 8 1. Bayerischer T_EX-Stammtisch in Nürnberg

Bretter, die die Welt bedeuten

- 10 Accents, accents, accents... enhancing CM fonts with “funny” characters
- 32 Tipps und Tricks: Mal anders herum – *excludeonly*
- 33 Erstellen von Schaltbildern mit `pst-circ`
- 49 PDF und PostScript – das L^AT_EX-Paket ps4pdf

T_EX-Beiprogramm

- 57 Typograf der Zeit – Hans Peter Willberg ist tot

Rezensionen

- 59 „L^AT_EX für Dummies“ Christian Baun

Leserbriefe

- 64 Zu Torsten Brongers Artikel „Einfaches Setzen von Texten in Fraktur mittels `blackletter1`“

Spielplan

- 67 Termine
- 68 Stammtische

Adressen

- 71 Autoren/Organisatoren