MAPS

NUMMER 31 • NAJAAR 2004

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

Voorzitter
Hans Hagen
pragma@wxs.nl

Secretaris
Willi Egger
w.egger@boede.nl

Penningmeester
Wybo Dekker
wybo@servalys.nl

Bestuursleden
Maarten Wisse
Maarten.Wisse@urz.uni-hd.de

Frans Goddijn
frans@goddijn.com

Karel Wesseling
k.h.wesseling@planet.nl

Postadres
Nederlandstalige TEX
Gebruikersgroep
Maasstraat 2
5836 BB Sambeek

Postgiro
1306238
t.n.v. NTG, Deil
BIC-code: PSTBNL21
IBAN-code: NL05PSTB0001306238

E-mail bestuur
ntg@ntg.nl

E-mail MAPS redactie
maps@ntg.nl

WWW
www.ntg.nl

De **Nederlandstalige TEX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft de kennis en het gebruik van TEX te bevorderen. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde documentopmaak in het algemeen en de ontwikkeling van 'TEX and friends' in het bijzonder. De doelstellingen probeert de NTG te realiseren door onder meer het uitwisselen van informatie, het organiseren van conferenties en symposia met betrekking tot TEX en daarmee verwante programmatuur.
De NTG biedt haar leden ondermeer:

- Tweemaal per jaar een NTG-bijeenkomst.
- Het NTG-tijdschrift MAPS.
- De 'TEX Live'-distributie op DVD/CDROM inclusief de complete CTAN softwarearchieven.
- Verschillende discussielijsten (mailing lists) over TEX-gerelateerde onderwerpen, zowel voor beginners als gevorderden, algemeen en specialistisch.
- De FTP server `ftp.ntg.nl` waarop vele honderden megabytes aan algemeen te gebruiken 'TEX-producten' staan.
- De WWW server `www.ntg.nl` waarop algemene informatie staat over de NTG, bijeenkomsten, publicaties en links naar andere TEX sites.
- Korting op (buitenlandse) TEX-conferenties en -cursussen en op het lidmaatschap van andere TEX-gebruikersgroepen.

Lid worden kan door overmaking van de verschuldigde contributie naar de NTG-giro (zie links); vermeld IBAN- zowel als SWIFT/BIC-code en selecteer shared cost. Daarnaast dient via `www.ntg.nl` een informatieformulier te worden ingevuld. Zonodig kan ook een papieren formulier bij het secretariaat worden opgevraagd.
De contributie bedraagt € 40; voor studenten geldt een tarief van € 20. Dit geeft alle lidmaatschapsvoordelen maar *geen stemrecht*. Een bewijs van inschrijving is vereist. Een gecombineerd NTG/TUG-lidmaatschap levert een korting van 10% op beide contributies op. De prijs in euro's wordt bepaald door de dollarkoers aan het begin van het jaar. De ongekorte TUG-contributie is momenteel $65.

MAPS bijdragen kunt u opsturen naar `maps@ntg.nl`, bij voorkeur in LaTEX- of ConTEXt formaat. Bijdragen op alle niveaus van expertise zijn welkom.

Productie. De Maps wordt gezet met behulp van een LaTEX class file en een ConTEXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdfetex 1.20a Web2C 7.5.3 draaiend onder Linux 2.6. De gebruikte fonts zijn Bitstream Charter, schreefloze en niet-proportionele fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

TEX is een door professor Donald E. Knuth ontwikkelde 'opmaaktaal' voor het letterzetten van documenten, een documentopmaaksysteem. Met TEX is het mogelijk om kwalitatief hoogstaand drukwerk te vervaardigen. Het is eveneens zeer geschikt voor formules in mathematische teksten.
Er is een aantal op TEX gebaseerde producten, waarmee ook de logische structuur van een document beschreven kan worden, met behoud van de letterzetmogelijkheden van TEX. Voorbeelden zijn LaTEX van Leslie Lamport, $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX van Michael Spivak, en ConTEXt van Hans Hagen.

Inhoudsopgave

Redactioneel

Voor u ligt het eenendertigste nummer van de Maps. Ook nu weer verlaat: het is het najaarsexemplaar, terwijl in de verte dankzij de geweldige hoeveelheid broeikasgassen die we met zijn allen weer hebben weten te produceren het voorjaar alweer lonkt.

Daar staat tegenover dat we nu de helft van de Maps in kleur hebben kunnen uitbrengen, zoals in de vorige Maps al werd aangekondigd. Daar willen we graag een gewoonte van maken, maar dat kan alleen als er ook kleurkopij wordt aangeleverd. Daarom herhaal ik mijn oproep uit de vorige Maps: wanneer u TEX-documenten in kleur produceert of hebt geproduceerd: laat ons het resultaat ook eens zien!

Dan wil ik u nu graag meenemen langs de bonte verzameling van artikelen in deze Maps:

- Piet van Oostrum (*Een uittreksel uit de recente bijdragen in het CTAN archief*) vraagt aandacht voor nieuwe pakketten, met nadruk op presentatie (BEAMER) en kleur in LaTEX, wiskunde in ConTEXt en vele andere nieuwe ontwikkelingen,
- Hans Hagen (*The State of ConTEXt*) vertelt hoe hij denkt dat ConTEXt zich zal gaan ontwikkelen,
- Taco Hoekwater (*MetaPost Developments*) vertelt over nieuwe initiatieven om MetaPost te destabiliseren,
- Giuseppe Bilotta (*The Aleph project*) vertelt over de ontwikkeling van het Aleph project, dat de mogelijkheden van Omega, ε-TEX en PDFTEX zal combineren,
- Maarten Sneep (*Producing graphs with MetaPost*) laat zien hoe figuren met MetaPost kunnen worden uitgelijnd en hoe je, als je veel fouten maakt, die in MetaPost met *error bars* kunt uitvergroten,
- Dwight Aplevich (*Circuit_macros*) beschrijft zijn CIRCUIT_MACROS, die PStricks gebruiken om de ingewanden van uw computer te tekenen,
- Frans Goddijn (*Een briefhoofd maken*) beschrijft zijn queeste naar mooi briefpapier,
- Brooks Moses (*MetaPlot, MetaContour, and Other Collaborations with MetaPost*) gaat in op methoden om berekeningen waar MetaPost niet (goed) voor geëquipeerd is door externe programma's te laten uitvoeren en ze daarna toch door TEX en MetaPost te laten zetten,
- Willi Egger en Hans Hagen (*Support for typesetting greek in ConTEXt*) vertellen hoe Griekse teksten in ConTEXt documenten kunnen worden opgenomen,
- Steve Grathwohl (*A Simple Book Design in ConTEXt*) gaat u ervan overtuigen dat het veel gemakkelijker is een boek in ConTEXt te ontwerpen dan in LaTEX,
- Adam Lindsay (*OpenType in ConTEXt*) doet uit de doeken hoe hij, door aanpassing van TeXFont OpenType fonts in ConTEXt beschikbaar maakte,
- Hans Hagen (*Fontgebruik*) laat twee curieuze zetsels zien uit de negentiende eeuw en daagt u uit mee te doen aan een wedstrijd om die in TEX te reproduceren,
- Frans Goddijn (*Conversies*) maakt duidelijk hoe het komt dat hij zoveel vrienden onder ons heeft,
- Siep Kroonenberg (*Exact layout with LaTEX*) leert ons een briefhoofd te ontwerpen,
- Wybo Dekker (*Boekdrukken en valkuilen*) ervoer dat het zetten van een boek niet zo moeilijk is als het netjes laten drukken ervan,
- Eckhart Guthöhrlein (*Object-Oriented Graphics with MetaObj*) maakt ons lekker met de prachtige plaatjes die met MetaObj (een OO pakket bovenop MetaPost) kunnen worden geproduceerd,
- Patrick Gundlach (*The ConTEXt Garden*) neemt u mee in de Tuin der Lusten, waar alles over ConTEXt te vinden zal zijn,
- Hans Hagen (*Fonts, more than a sample*) maakt u wegwijs in het kleurige font-boekje dat u onlangs ontving,
- Frans Goddijn en Willi Egger (*Bloei der decadence*) laten zien ze hoe een uitverkocht boek via ConTEXt weer tot leven brachten,
- Hendri Adriaens en Uwe Kern (*Keys and Values*) beschrijven hun `xkeyval`-pakket, een flexibele extensie van het `keyval`-pakket,
- Taco Hoekwater (*Boekbespreking Vormwijzer*) recenseert een boek.

De redactie vertrouwt erop dat u veel inspiratie zult opdoen voor een eigen verhaal in de volgende Maps.

Wybo Dekker
wybo@servalys.nl

Een uittreksel uit de recente bijdragen in het CTAN archief

Keywords
TeX, LaTeX, packages, CTAN, classes, beamer, slides,

Abstract
Dit artikel beschrijft een aantal recente bijdragen uit het CTAN-archief. De selectie ervan is gebaseerd op wat ik zelf interessant vind en op wat ik denk dat voor veel anderen interessant is en is dus een persoonlijke keuze. Het is niet de bedoeling om een volledig overzicht te geven. De uitgebreidere bijdragen zijn ook geen handleidingen. Beschouw dit artikel maar als een soort menukaart die de bedoeling heeft om de lezer lekker te maken.

Presentaties met LaTeX

De vorige keer heb ik een aantal LaTeX-packages voor het maken van presentaties de revue laten passeren. Deze keer wil ik beginnen met het vervolg ervan, en wel met het package 'beamer' waarmee ik de vorige keer eindigde. Ik heb de indruk dat in de laatste maanden dit package enorm in populariteit is gestegen, en dat het momenteel als de state-of-the-art oplossing wordt beschouwd. Je kunt je afvragen waarom dat zo is en hoe lang het zal duren. Tenslotte gebeurt het regelmatig dat een package wat op dit moment als het beste beschouwd wordt, plotseling een concurrent krijgt waar niet meer tegen op te boksen is. Vaak als gevolg van het feit dat de auteur van het eerste package intussen vanwege zijn werkzaamheden niet genoeg tijd meer heeft om het package in de vaart der volken op te stoten of zijn interesse erin verloren heeft. We zullen zien. Voor gebruikers van zo'n package is dat wel eens zuur, want ook zij willen mee met de ontwikkelingen maar conversie is niet altijd even simpel.

Beamer presentaties

Enige tijd geleden is versie 3.01 van beamer uitgekomen. Beamer heeft een aantal kenmerken die het erg aantrekkelijk maken voor het maken van presentaties:

□ Je kunt in één bestand een presentatie, notities voor de spreker, hand-outs voor de luisteraars en een normaal artikel samenbundelen. Dit heeft het voordeel dat je geen duplicatie van code krijgt. Als je bijvoorbeeld vergelijkingen, tabellen of diagrammen hebt dan komen die maar één keer voor en je loopt dus minder risico dat ze inconsistent worden.

□ Er zijn uitgebreide voorzieningen om de slides te animeren. Ook packages als pdfslides, texpower en zo hebben animatie-mogelijkheden, maar deze hebben verschillende beperkingen (bijvoorbeeld dat alleen in verticale mode – tussen alinea's – een onderbreking kan plaatsvinden). In beamer zijn deze beperkingen opgeheven.

□ Beamer werkt met *themes*, tegenwoordig een populair concept, bijvoorbeeld in user interfaces. Een theme bepaalt de layout van de slides, en heeft een kleurenschema, de vorm van boxen (rechthoekig of met afgeronde hoeken, wel of niet schaduwen), fonts, symbolen voor itemize en dergelijke. De themes van beamer zijn weer opgebouwd uit subthemes. Daardoor is het gemakkelijk onderdelen van themes te combineren.

□ Beamer documenten kunnen zowel met gewoon LaTeX als met PDFLaTeX verwerkt worden. Dit geeft flexibiliteit met betrekking tot het gebruik van figuren.

□ Voor gebruikers van andere packages zoals seminar, prosper en texpower zijn er packages om de conversie te vergemakkelijken. Een `\usepackage{beamerseminar}` (wanneer je van seminar over wilt stappen) met enkele tekstuele aanpassingen kan in voorkomende gevallen voldoende zijn.

Beamer komt met een uitgebreide gebruikershandleiding (200 pagina's). Deze handleiding bevat ook aanwijzingen over hoe een presentatie te structureren. Beamer is te vinden op CTAN: `macros/latex/contrib/beamer/` of op Sourceforge (`http://latex-beamer.sourceforge.net/`). Hier is ook het ondersteunende package pgf (zie hieronder) te vinden. Verder maakt beamer gebruik van het xcolor-package, ook hieronder genoemd.

Tijdens het schrijven van dit artikel is versie 3.01 uitgekomen. Een belangrijke wijziging in deze versie is de mogelijkheid om in slides met verbatim tekst of verbatim-achtige constructies ook animaties te gebrui-

ken. Dit gebeurt door de optie `fragile` aan een frame (slide) mee te geven. In vorige versies was dit niet mogelijk. Helaas zit er een bug in, waardoor op deze slides het nummer met twee verhoogd wordt in plaats van met één.

Met de nieuwe versie is het ook gemakkelijker om oudere packages te emuleren. Ik ben zelf met mijn college-slides begonnen met het gebruik van seminar en vrij snel daarna overgestapt op foils (toen ging het nog om op transparanten afgedrukte slides). Toen het gebruik van beamers meer gangbaar werd besloot ik pdfslides in combinatie met texpower te gebruiken en heb toen een eigen class geschreven die voortbouwde op deze twee, zodat ik mijn slides zelf nauwelijks hoefde te veranderen (voornamelijk de class-naam). Nu heb ik de class omgeschreven om beamer te gebruiken en kunnen mijn slides praktisch zonder wijzigingen hergebruikt worden met de beamer-layout. Bij de vorige versie van beamer zaten daar nog wat haken en ogen aan; bij de nieuwste is het een stuk gemakkelijker.

pgf

Pgf (Portable Graphics Format) is een LaTeX-package voor het maken van tekeningen. De tekeningen worden opgebouwd uit LaTeX-commando's net zoals bij de LaTeX `picture` omgeving, pstricks, en vele andere packages.

Het package bevat een kern voor het maken van basisfiguren, en hierop gebouwd een aantal extensies voor het tekenen van pijlen (pgfarrows), diagrammen met knopen en verbindingen ertussen (pgfnodes), en kleurgradiënten (pgfshade). Het bijzondere van pgf is dat het zowel Postscript (via dvips specials) als PDF kan genereren. Hoewel het minder krachtig is dan pstricks weegt deze portabiliteit in veel gevallen op tegen de beperking van pstricks dat het in een Postscript omgeving gebruikt moet worden of met kunst- en vliegwerk via packages als ps4pdf.

Versie 0.64 van pgf had een hinderlijke bug als het samen met het calc package gebruikt werd. In versie 0.65, die ten tijde van dit schrijven werd uitgebracht is deze verholpen.

Het package is te vinden op CTAN: `/graphics/pgf/` of op bovengenoemde Sourceforge site.

xcolor

Met het xcolor package kunnen gemakkelijk meer kleuren gebruikt worden dan in standaard LaTeX. Bovendien kan dit op een manier die (min of meer) onafhankelijk is van de gebruikte output-driver. De huidige versie ondersteunt dvips, xdvi, dvipdf, dvipdfm, pdftex, dvipsone, dviwindo, emtex, dviwin, oztex, textures, pctexps, pctexwin, pctexhp, pctex32, truetex, tcidvi, vtex. Ik moet bekennen dat er verschillende drivers in dit lijstje zitten waar ik nog nooit van gehoord heb.

Xcolor ondersteunt verschillende kleurmodellen zoals RGB en CMYK. Er zijn voorgedefinieerde kleuren en er kunnen eigen kleuren gedefinieerd worden. Ook kunnen kleurensubstituties uitgevoerd worden. Bijvoorbeeld met het commando:

```
\def\xcolorcmd{\colorlet{black}{red}}
```

kan alle zwart in een document veranderd worden in rood (althans in de LaTeX-code). Ik neem aan dat de kleuren in ingevoegde tekeningen niet veranderen. `\xcolorcmd` is een commando dat gedefinieerd kan worden voordat het package geladen wordt, en dat dan bij initialisatie van het package uitgevoerd wordt.

Xcolor kent ook kleur-*expressies*. Zo selecteert `\color{-red}` het complement van de kleur rood, en `\color{green!20!blue}` een mengsel van 20% groen en 80% blauw. Iets ingewikkelder is het met `\color{red!40!green!20!blue}`. In dit geval wordt eerst 40% rood met 60% groen gemengd, en van dit mengsel wordt 20% gemengd met 80% blauw. `\color{rgb:red,4;green,2;yellow,1}` mengt 4 delen rood, 2 delen groen en 1 deel geel.

Lichtere kleuren kunnen verkregen worden door te mengen met wit (waarbij `white` aan het eind nog weggelaten mag worden), en donkerder door te mengen met zwart of grijs.

Xcolor heeft ook nog commando's om rijen in tabellen (`tabular`) te kleuren.

CTAN: `/macros/latex/contrib/xcolor/`.

ConTeXt en wiskunde

Voor mensen die serieuze wiskunde in hun LaTeX-documenten willen verwerken is het amsmath package bijna onontbeerlijk. Dit package is voor de AMS (American Mathematical Society) ontwikkeld door de vorig jaar overleden Michael Downes. Het is onderdeel van een suite die ook wel AmS-LaTeX genoemd wordt. Het bevat onder andere voorzieningen voor het uitlijnen van stelsels vergelijkingen.

Er is nu een collectie ConTeXt modules beschikbaar die dezelfde faciliteiten biedt. `T-amsl.tex` komt overeen met AmS-LaTeX, en `t-nath.tex` komt overeen met het LaTeX-package nath. Deze zijn te vinden op CTAN in `macros/context/contrib/maths/`.

Nath is overigens een package voor een 'Natural math notation'. De notatie van de wiskundige formules kan hier min of meer losgekoppeld worden van de presentatie. Bijvoorbeeld het commando `\frac` kan zowel een deling met een horizontale deelstreep als

met een schuin deelteken genereren, en voegt zelf zo nodig haakjes toe.

Op CTAN: `macros/latex/contrib/supported/nath/`.

Documentatie over wiskunde

De CTAN file `info/math/voss/Voss-Mathmode.pdf` bevat een uitgebreide beschrijving over het vormgeven van wiskundige formules met LaTeX. Het is geschreven door Herbert Voss. Het beschrijft de standaard LaTeX-commando's, de AmS-LaTeX uitbreidingen, en in het kort nog een hele serie andere packages die nuttig zijn in het wiskundige werk. Voor iedereen die formules moet gebruiken van harte aanbevolen. Misschien iets om een keer in de Maps te publiceren.

Er is ook een document van dezelfde schrijver over kleurgebruik in wiskundige formules, maar dit is in het Duits, wat waarschijnlijk door minder mensen begrepen wordt dan Engels. CTAN: `info/math/voss/Voss-mathCol.pdf`.

Andere bijdragen

Onderstaande bijdragen zijn LaTeX-packages of -classes, tenzij anders vermeld. De locaties zijn op CTAN. Meestal betreft het updates van bestaande packages.

pdfcrop `Pdfcrop` is een Perl-script om een PDF-file te 'croppen' dat wil zeggen de witruimte eromheen weg te halen. Sommige programma's genereren PDF-files op een A4 of soortgelijke pagina en dat maakt het lastig om ze in een document in te voegen. Met `pdfcrop` kan de overbodige ruimte weggehaald worden. CTAN: `support/pdfcrop/`

subfig Subfig is een nieuw LaTeX-package dat het vroegere subfigure vervangt. De naam is gewijzigd omdat het incompatibel is met het oude package. Er is wel een configuratiefile waarmee het oude grotendeels geëmuleerd kan worden. CTAN: `macros/latex/contrib/subfig`.

subfloat Subnummering voor tables en figures. Dit is anders dan het subfig package dat subfiguren binnen een hoofdfiguur ondersteunt. CTAN: `macros/latex/contrib/supported/subfloat/`.

ccaption Een package voor het manipuleren van captions: Continuation captions (voor als een float een vervolg is van een vorige), ongenummerde captions, captions buiten een float-omgeving, het gebruik van stijlen voor captions en het definiëren van nieuwe floats. CTAN: `macros/latex/contrib/supported/ccaption/`.

epigraph Een package voor het zetten van 'epigraphs', stukjes tekst onder een hoofdstuk titel, vlak voor de eigenlijke tekst van het hoofdstuk. CTAN: `macros/latex/contrib/supported/epigraph/`.

lineno Een package voor het nummeren van regels in de uitvoer van een document. Dit is een onderdeel van de ednotes bundel voor kritische edities. CTAN: `/macros/latex/contrib/supported/ednotes/`.

minitoc Een package om mini-inhoudsopgaves per hoofdstuk, sectie en dergelijke te maken. CTAN: `macros/latex/contrib/minitoc/`.

floatrow Dit package kan gebruikt worden om de layouts van floats te veranderen. Bijvoorbeeld floats naast elkaar, captions naast floats. CTAN `macros/latex/contrib/floatrow/`.

AUCTeX Een Emacs uitbreiding voor het editen van TeX- en LaTeX-files. De nieuwe versie (11.53) bevat vele verbeteringen, o.a. betere ondersteuning van PDF, het editen van `dtx`-files en ConTeXt ondersteuning (weliswaar nog in de kinderschoenen). Voor meer informatie zie `http://www.gnu.org/software/auctex/`.

movie15 Een package voor het invoegen van multimedia files (video en geluid). Dit maakt gebruik van PDF versie 1.5 (vandaar de 15 in de naam). CTAN: `macros/latex/contrib/movie15/`.

CurVe Een package voor het maken van een curriculum vitae. CTAN: `macros/latex/contrib/supported/curve/`.

europecv Nog een package voor het maken van een CV; ditmaal volgens de normen van de Europese Commissie. CTAN: `macros/latex/contrib/europecv/`.

Piet van Oostrum
piet@cs.uu.nl

The State of ConTEXt

Abstract
In this article I will describe the current state of the ConTEXt macro package and the forces that play a role in its evolution. I will also indicate the directions in which we look for further developments.

ConTEXt developments

The public part of the ConTEXt story started around 1995. If we summarize the main developments in this macro package we can roughly identify the following points of focus:

□ a configurable environment where users can define styles, using an interface in a language of choice; the multilingual interface was first needed when the chemical package ppchTEX was adapted to English and made generic
□ support for document collections as we find in educational settings, with a focus on re-usability; multilingual support, selective processing and dedicated modules for chemical formulas and consistent usage of physical units evolved from there
□ features aimed at highly interactive documents, optimized for reading on computer screens; support for one source, multiple output was part of that
□ extensive support for grid snapping combined with advanced multi column typesetting
□ typescripts as a means of building font collections and combining typefaces in many (possibly weird) ways
□ all kind of fuzzy configuration options needed in order to mimic the behavior of desktop publishing applications
□ integrated support for processing XML documents and using XML databases

Extending and improving ConTEXt has never been related to strong versioning or promises for successors. Part of the game is that we try to remain downward compatible. And so, officially we still have ConTEXt version 1. Successive releases are tagged by date.

The most recent change was not so much related to new features but more to the machinery behind the screens. Those who have looked into the source code, probably have noticed that for reasons we will not discuss now, keywords and variable names look rather Dutch, that is, until recently. Around August 2004 we made the move to a low level English interface. Although we had some help from a Perl script that had been written for this purpose years ago, still quite some manual checking had to be done.

This does not mean that ConTEXt is completely clean under the hood. When we started developing the system, TEX's were small, and so we ended up with quite some dirty (not that verbose) code. One can easily recognize the older code, but we hope to weed out the ugly bits in due time.

There is good reason to qualify the current version as ConTEXt version 2. The reason for this is that users who use low level Dutch keyword constants (prefixed by `\v!` and `\c!`) in their non–dutch styles, need to translate these into English. A bonus is that third party extensions will be easier to implement. Such developments will further be stimulated by Taco Hoekwater's ConTEXt API project and Patrick Gundlach's ConTEXt interface description project hosted at `contextgarden.net`. I must admit, that the decision to go low–level–english now and not later, was triggered by their initiatives.

Of course one can legitimately ask whether there is still need for further developments in TEX macro-packages like ConTEXt. At Pragma ADE we deal with documents coded in TEX as well as the more avant–garde XML format. It cannot be denied that XML coding makes documents much less error-prone: it's much simpler to check the syntax of an XML file than of an TEX file. However, it can also not be denied that the loss of typographic detail (or more precisely: the means of authors to improve the look and feel of the final result) is a high price to pay. Of course there are also document types that cannot easily be covered by a manageable set of XML elements. Just think of highly complex math, physics or chemistry, or manuals that use a wide range of visualisations. One easily ends up with something that, although coded without backslashes, looks rather familiar to TEX users.

An even more important observation is that whatever means of going from document source to typeset product we choose, the visualisation problem will not change. No matter how many tools (or macros) one writes, differences in designs (and not seldom inconsistencies in designs) demand unique solutions. Although one can easily become overwhelmed by the possibilities that todays publishing tools provide, there is still a place for the proven TEX technology.

ConTEXt and browsers

Recently we redesigned the Pragma ADE web-site, a site that is mostly dedicated to ConTEXt. The HTML pages are generated from XML sources using xsltproc. Some of the PDF documents are generated from the same XML code. In addition, we generate templates and interfaces for the PDF framework, of which we now run an instance on the web site. This framework is a shell around TEX and friends, and provides features like page imposition and font tests, wrapped in an interface, but very TEXish underneath.

When rebuilding the web site, it was enlightening to find out that standards like css were not always precisely supported, even after being around for many years. FireFox (mozilla gecko engine) does a decent job. But for Internet Explorer, we have to cheat dimensions and use dirty tricks to get the alignment right. Opera was not that bad, but could not handle relative dimensions well. In the end we had to follow yet another approach to make Apple's Safari Browser (based on the kde engine) happy as well. One lesson that I learned here was that even an abundance of implementations (or renderers) and tons of documentation (it's easy to find info in the web on css and HTML) makes defining a simple layout a painful and time consuming process. It's also interesting to see that the amount of XSLT code needed is not on fore-hand smaller than the ConTEXt code needed to generate similar output in PDF. Although the TEX community is under pressure of evolving techniques, it should also realize that its huge repository of tools and macros is not that bad after all.

Interesting is that browsers can handle complex operations, like displaying Arab or Chinese and handling widgets and JavaScript quite well, but setting up a simple geometry based on fractions (percentages of the screen size) goes beyond their capacity. Something similar can be observed with the XML related css cousin XSL-FO: I still have to run into a nicely typeset book done that way with a more than mediocre design. Again the focus seems to be more on the machinery around it, than on the creation of masterpieces. But then, this may well be beyond its purpose. Whatever a TEX user may think of css compared to his or her favorite macro package, its influence is undeniable. The evolution of the Mozilla platform demonstrates this: it provides a user interface builder based on css and xhtml called xul. When PDF came around, I made some documents that could considered to be programs. It looks like in the end typesetting and user interfacing finally meet each other.

Future developments

The majority of documents is a collection of paragraphs of running text, itemized lists, a few graphics here and there, and a couple of tables. TEX and TEX-related packages can handle such documents with ease. However, it seems that even in automated work-flows, where most of the interface can be hidden, TEX is seldom considered to be an option. But, when no other alternative is available, or when other applications failed to perform, this 25 year old program can come to help. It's interesting to observe that the TEX community can still attract new users who don't consider the user interface too much of a problem. So it definitely makes sense to continue development, if only because there is still a large group of documents that demand such tools and typographical detail. As long as TEX can keep up, the ConTEXt story will continue and we will see version 4 (extremely modularized), version 8, 16 and maybe 32 some time in the future. In the end it may be that properly typeset documents where time and effort is put in the look and feel, become a niche, and make place for documents with a minimum of design that can be generated each time they are updated, using the user's preferences.

What is currently happening at the ConTEXt frontier? ConTEXt has been ε-TEX aware for a long time, and the pdfTEX engine is supported quite well. The good news is that pdfTEX is still under active development. For instance large parts of the font handling were redesigned, and paragraph optimization (pdfTEX implements a font expansion algorithm akin to the hz micro–typography algorithm by Prof. Hermann Zapf) as well as protruding (hanging punctuation) have become more user friendly. ConTEXt supports both mechanism quite well.

With the arrival of Aleph, the stable descendant of Omega, support for this extension will become more visible than it was so far. Although UTF is supported, as well as some specialized Chinese encodings, using Aleph will bring unicode support in the broadest sense, given that adequate fonts and hyphenation patterns become available. In many aspects Omega is not as multilingual as advertised, and certainly not by nature. Omega and therefore Aleph provide some mechanisms, but one still needs macros on top of these to tie the directional typesetting to actual languages and layout. Taking fonts as a starting point, the MacOs X specific unicoded TEX variant XeTEX also looks promising. According to one of the ConTEXt MacOs X experts, Adam Lindsay, there are hardly any extensions to ConTEXt are needed in order to get documents typeset in virtually every script.

Other developments that may become of interest are Taco Hoekwater's merge of TEX and MetaPost. There ConTEXt will not only benefit from a speedup due to more efficient inter–process communication, but it may also open new worlds. The average user will probably not use ConTEXt the way we do, for instance to create DTP like output from XML sources, which often means multiple calls to MetaPost per page. Think of documents with 250–500 pages, hundreds of

(possibly run time manipulated) graphics, thousands of calls to MetaPost, with an occasional size of over 500 Megabytes, and you can imagine that any speed improvement counts. Most features that we use in projects end up in the kernel, and so many users may profit from an efficient integration.

I already mentioned XML. In the next couple of years, more ConTEXt subsystems will use this format in one way or another. If you take a closer look at the distribution, you will notice that quite some XML objects are present already, like in the figure database mechanism and other tools. New is foXet, yet another XSL-FO engine. Formatting objects (fo's) are a kind of building blocks that are to be handled by a typesetting engine.

Although foXet ended up on our agenda due to some vague promises made long ago, the actual development of foXet was triggered by the observation that the ConTEXt MathML engine is being used to fill in the gap in commercial engines. Why bother making small bitmaps (or PDF snippets) of formulas while TEX can do the whole thing? It is interesting to notice that most of the documents that this applies to are rather trivial to typeset with either ConTEXt built–in XML features or by using XSLT to generate intermediate TEX code. It is also interesting to observe that there are ConTEXt users who use XML documents with ConTEXt as a backend, thereby hiding TEX completely.

The magic sound of XSL-FO occasionally makes our customers express the wish for an engine that can handle them (even if their designs are not that well suited for it). Somehow the magic obscures the fact that it's a relatively slow process, that it may take longer to implement (as said before: the problem does not change), does not necessarily lead to well typeset documents, et cetera. If one knows that something is possible (and with TEX much is possible) the demands of designers are seldom adapted. When something is not possible at all (and this occurs with XSL-FO) my guess is that the demands will be dropped. Float handling and marginal notes are examples of areas where TEX is hard to beat.

Paragraph building

So what about TEX's superior paragraph builder? Unfortunately most of the documents that we have to typeset professionally are designed by those who use DTP systems with poor quality paragraph builders. This makes that they simply cannot believe that there are programs that can do a decent job. As a result we end up with colorful and abundantly illustrated documents that have rather complex layouts (especially if you take into account that they are typeset automatically) but with poorly typeset paragraphs, and that is what they recognize. It is hard to explain that by setting all TEX's penalties to their maximum, the solution space becomes pretty small. Even the somehow always demanded ragged right justification then looks plain bad. The problem for the TEX community is that alternatives for TEX don't have to provide TEX quality paragraph routines. As long as they can get the layout done, they win the game. ConTEXt users who like to look into the source will have noticed that quite some control was added in order to meet these demands, even to the extent that it may lower the quality.

So what good is it for TEX users? As with many things, it's no bad idea to take the best of all worlds. There is nothing wrong with DTP, and for many applications, an Office Suite does well. And for a certain range of documents XSL-FO is a good choice. Of course it remains puzzling why some of todays publishing on demand work-flows are presented as something new, while in practice it already could have be done that way for decades using SGML and TEX, at far lower costs too. In some sense TEX was simply to far ahead.

One can mix those techniques. Just as one makes a graphic in a drawing program, one can imagine embedding one page documents coded in XSL-FO as graphic in a TEX document. In this way we get a kind of 'placed XML'. And ConTEXt already can happily combine TEX and XML in such ways. Also, it's more convenient to store information in a standardized (XML) format, than to invent some syntax for each situation and develop different tools for each of them. For instance, if we want file information in our documents, we use `xmltools` to generate a directory database (this can be done at document processing time by using a system call) and we then let TEX filter the information from that database. Another example is OpenOffice. Anyone who has taken a closer look at this program will probably have recognized similarities with XSL-FO related developments. Seeing TEX as an alternative back-end for texts edited in that environment is not such a bad idea.

All these worlds can meet each other in ConTEXt. In ConTEXt, TEX and XML come together not only in foXet, but also in what we've called 'The Example Framework'. The PDF logo has the x, m, and l hidden inside, but the actual purpose of this project is to hide TEX from users. On our web site you can play with some of these framework features.

It will be clear that the future of ConTEXt is to some extent related to the advance of XML, although the pure TEX approach will not be neglected. For many documents the TEX syntax (or in our case, the ConTEXt one) is quite well suited and efficient. Although I nowadays code most database related documents in XML (like the PDF showcase document interfaces) I have no plans to abandon TEX. Even thinking of coding a manual like the one about MetaFun in XML already gives reasons for nightmares. And so . . . plenty of ConTEXt ahead.

Hans Hagen

MetaPost Developments

Keywords
MetaPost, development, sarovar, bugs, extensions

The MetaPost system (by John Hobby) implements a picture-drawing language very much like that of MetaFont except that it outputs Encapsulated PostScript files instead of run-length-encoded bitmaps. MetaPost is a powerful language for producing figures for documents to be printed on PostScript printers, either directly or embedded in TEX documents. It includes facilities for directly integrating TEX text and mathematics with the graphics.

The version number of the MetaPost executable is still well below the 1.0 mark (0.641 is current), but not much has happened in recent years. This situation is far from satisfactory, especially since a fairly large number of bugs are known to exist at this date, but John Hobby simply could not find the time to solve these bugs, let alone handle feature requests.

Resulting from a renewed community interest in MetaPost, last summer a small group of people have made a proposal to Hobby for the creation of a special development group that would take care of the development of MetaPost from then on. Luckily, he agreed, on the condition that he will only allow tested code to be inserted into the MetaPost distribution. Among the currently active group are the following people:

- Karl Berry
- Giuseppe Bilotta
- Hans Hagen
- Taco Hoekwater
- Bogusław Jackowski

Karl Berry has created a homepage on the TUG server for MetaPost

- http://www.tug.org/metapost

He also created a mailinglist for discussions and questions. Details can be found at

- http://www.tug.org/mailman/listinfo/metapost

Taco Hoekwater has set up a project at Sarovar that hosts a source repository as well as a bug / feature request tracker

- http://www.sarovar.org/projects/metapost

The MetaPost manuals (mpman and mpgraph) have recently been released under a BSD-ish license. Dylan Thurston at Debian converted the sources to LATEX, and in the future they will become a standard part of the distribution.

As of today, the known errors in the documentation have been removed, and a number of bugs have already been fixed in the repository. More bugs will be fixed in the near future, and the group hopes that a new bugfix release will be available around EuroTEX 2005.

Taco Hoekwater
taco@elvenkind.com

The ℵ (Aleph) project

Keywords
Omega, Aleph, eTeX, eOmega

Abstract
A brief introduction to the ℵ project, a TEX extension providing most of Ω and ε-TEX features.

The path

TEX was created by Donald Ervin Knuth more than 20 years ago. At the time when it was initially developed, it was supposed to serve mainly one purpose (provide a high–quality typesetting workbench for Knuth's books), but was general–purpose and powerful enough to be quickly adopted as a more or less standard environment in the scientific community, thanks to its capability to easily typeset complex formulas.

Usage of TEX outside the scientific/technical domain has always been confined to niche applications, partly due to the absence of high–level formats like LATEX, but more geared towards nontechnical writing, and partly because TEX, in its original design, has very limited support for languages other than English.

Efforts to push the limits of TEX have been made, in at least three different directions, by different teams. This led to the creation of multiple, sometimes incompatible extensions of the original engine; we have for example

- pdfTEX, which gives TEX the capability to produce output directly in PDF form, and introduces micro–typesetting capabilities;
- ε-TEX, based on previous extensions of TEX which added right–to–left typesetting capabilities to TEX, which strives to break some of the structural limitations of TEX while maintaining maximum compatibility;
- Ω (Omega), an effort to bring the TEX world to up–to–date standards and push it towards a multicultural world.

As it was mentioned before, not all these extensions are compatible with each other; for example, while pdfTEX and ε-TEX can be merged in a single program, the changes in Ω are so extensive that they put the program in a rather isolated position.

While hopes and desires for a unified TEX extension have always been present, they have not been pressing because the most common format (LATEX) made no use of ε-TEX extensions, and other formats that did take advantage of those extensions (like ConTEXt) didn't have enough of a market to be of interest to Ω users. Things started to change recently, as the LATEX team found the original TEX more and more restrictive, and ConTEXt started spreading, its power and flexibility appealing to users outside the domain of latin scripts.

Birth of a new branch

The Ω project started with the best of intentions and reached some outstanding results; for example:

- 16–bit registers allow Ω to typeset documents too complex to be handled by TEX (or even ε-TEX);
- Ω introduces the concept of ΩTP (Ω Translation Process), a way to transcode texts, therefore allowing the input of text in any script in any encoding, in a font independent way and without the use of active characters; for example, one can write in Greek or Arabic using a plain English keyboard: ΩTP takes care of translating Latin characters (or sequences thereof) into the appropriate Arabic or Greek characters
- Ω can typeset text in many directions; ε-TEX provided some support for right–to–left typesetting, but Ω brings this capability to any combination of direction (left to right, right to left, top to bottom, bottom to top), easily combining these layouts in the same page.

But Ω presents some characteristics that can make its adoption a difficult choice with an uncertain future. These are the characteristics of an experimental project with a very broad (maybe too broad) final destination, a moving target where stability is only a secondary if not even tertiary target. While this experimental nature of the project is not intrisically negative (on the contrary, it guarantees that the project has enough dynamism to project itself into the future; in fact, it has been what brought Ω to its current status) it does hinder a widespread adoption of the tool for production use.

We therefore have a program, Ω, that has a lot of potential, and that needs to continue to develop; but this potential has to be somewhat harnessed to make the results available in production use contexts. Therefore, we chose to take the Ω code base and use it to start a new project, called ℵ (Aleph),[1] which could provide the power of Ω in an "affordable" manner.
Four main goals were set:

□ it had to be stable;
□ it had to be fast;
□ it had to be powerful;
□ it had to be readily available.

Meeting goals

Stability and speed
When the project was started, in late 2002, there were two publicly available versions of Ω which could be called "current": version 1.15 and version 1.23 (which I would call the "old" and "new" version respectively).

The reason why the old one was not taken off the distributions when the new one was published is that the new one suffered from "excessive bloat" which rendered it essentially unusable: processing even a simple document with the new version could take from five to ten times longer than processing it in the old version, memory consumption during the processing was at least twice as much and the resulting DVI was enormous. These shortcomings of the new version were a result of the introduction of a very powerful enhancement, with interesting potential but that needed to be greatly refined before it could become of common usage.

Because of this, most Ω people kept using the old version, which was almost as fast as TEX (although obviously not as slim). This version, on the other hand, had some extremely serious bugs which caused it to crash whenever overfull boxes were present.

Finding which version to choose to base ℵ on was not easy. Indeed, the first release of ℵ (at the time ε-Ω, Release Candidate 0) which was just a proof of concept that ε-TEX could be merged with Ω, was available both in a 1.15–based version and in a 1.23–based version, although the officially supported one was the former, therefore with an implied preference for speed over stability, on the assumption that fixing bugs would have been easier than solving the speed/bloat problem. This has later proved to be indeed the best choice.

Power
The immediate outcome of the third goal was that ℵ should have provided both ε-TEX and Ω features; since of course Ω already provides some of the extensions provided by ε-TEX, we could limit ourselves to the programming enhancement (extra marks, protected macros, `\scantokens`, etc).

Availability
This was probably the most important goal, since it would have been the one that "made the difference" with, e.g., Ω 2: ℵ had to be available in a usable status as quickly as possible; this led, among other things, to the choice of stripping from the Ω base the code that dealt with SGML and XML, since it conflicted with the code that implemented the `\middle` primitive in ε-TEX. Priorities led to this decision.

History of releases

The first version of ε-Ω was released in December 2002; while the "official" version merged ε-TEX on Ω 1.15, a parallel release based on Ω 1.23 was also made available. That version was no more stable than any of its components, and had an extra few bugs that crept in during the adaptation of the ε-TEX changefiles to the Ω structure. In particular, it had all the bugs present in Ω 1.15, which made it scarcely usable due to the major problem with overfull boxes.

The second step was trying to fix the most outstanding bugs coming from the Ω 1.15 codebase. This led to the first version of ε-Ω testable in production–use environments, Release Candidate 1, around June 2003. This was the version presented at TUG2004.

Subsequent versions finally officially switched to the ℵ name; the last public release (Release Candidate 2) also fixed some other significant bugs and started introducing some minor new features (the most important being the `\boxdir` primitive to retrieve/change the direction of a box, a feature backported from Ω 1.23).

Status

ℵ is actively developed on the TEXlive repository. A mailing list for discussions concerning the present and future of the project, including both the core program itself and any ancillary tool, is available (`aleph@ntg.nl`).

Development, as always, is focused mainly on the discovering (and fixing) of bugs, but discussions on possible future features are welcome. Currently, issues and bugs in ΩTPs and their interaction with some ε-TEX features (namely protected macros and the `\scantokens` primitive) are the most prominent target.

Progress

The main focus of ℵ will always be one of stability. This means, among other things, that each new release is supposed to be at least as stable as the previous one.

A test suite analogous to the TRIP test for TEX is being discussed. Indeed, TRIP proved itself a trusted friend in the discovery and resolution of the most notable bugs coming from the Ω 1.15 code base, namely the ones dealing with overfull boxes and leaders.

Given the stability of ℵ, it is important to remark that this does not imply a static, frozen behavior (à la TEX); on the contrary, ℵ should be considered a foundation on which to build: experimental projects to test the implementation of new features and ideas are welcome, provided they are developed separately; once they reach enough stability to be available for production uses, they might be candidates for the introduction in future versions of ℵ.

Acknowledgements

I wish to thank

- Donald Ervin Knuth, for providing us all with TEX
- John Plaice and Yannis Haralambous, for giving us Ω
- Peter Breitenlohner and the NTS team, for giving us ε-TEX
- Idris S Hamid, Alan Hoenig and Hans Hagen for pushing me into attempting the merge and supporting me all time long
- all the distribution maintainers for their constant feedback, help and support, with a particular thank to Christian Schenk and Fabrice Popineau for their essential help in getting me started with the coding
- everybody in the TEX world for making it the great community it is

Notes

1. The project was originally named ε-Ω, since it provided both ε-TEX and Ω features.

Giuseppe Bilotta
Dipartimento di Matematica e Informatica
Università di Catania
viale A. Doria, 6
95125 Catania
Italy
gip.bilotta@iol.it

Producing graphs with MetaPost

multiple aligned graphs and error bars

Keywords
MetaPost, graphs, error-bars

Abstract
MetaPost is an interesting companion for generating figures for documents written in TEX or one of its derivatives. This article focuses on generating graphs in MetaPost, and more specifically on two problems one can encounter when creating graphs: multi part graphs and error bars.

Introduction

In Maps 29 Karel Wesseling described how several MetaPost graphs can be aligned relative to each other, when including them in a `\startcombination[1*2]` command in ConTEXt. Here I describe a different approach to the same problem: aligning multiple graphs in a single figure.

The method proposed in that article required a solid, *near white* background to be printed behind the graphs. The lightness of this background ensured that it appeared white when printed, but somehow it still feels as a 'dirty' trick. Having the graph and the axes in a fixed location to the bounding box of the figure can be highly desirable for some effects though. One such situation occurs when one wants to align all graphs with the vertical axis on the left margin throughout the document (so that the axis labels stick out in the left margin).

When that is not required, a different solution becomes possible. Note that I use LaTEX instead of ConTEXt, but the same technique should work in other TEX and MetaPost combinations.

Another issue that is often encountered in generating (scientific) graphs is the inclusion of error bars in the graph. The second part of this note provides a possible solution for that.

Multiple graphs in a single figure

This is the annotated code to produce the graph shown in figure 1. The code generates a graph with multiple panes, in a single figure. The alignment is not accomplished in TEX but in MetaPost. We use the graph module written by John Hobby. This example uses LaTEX for its labels, and this temporary document requires some preamble material. This is also where you change the typefaces should you want to do so, and include other packages.

Since this example uses pdfLaTEX instead of TEX, the invocation of `mpost` is slightly more elaborate:

```
mpost -tex=latex file.mp
```

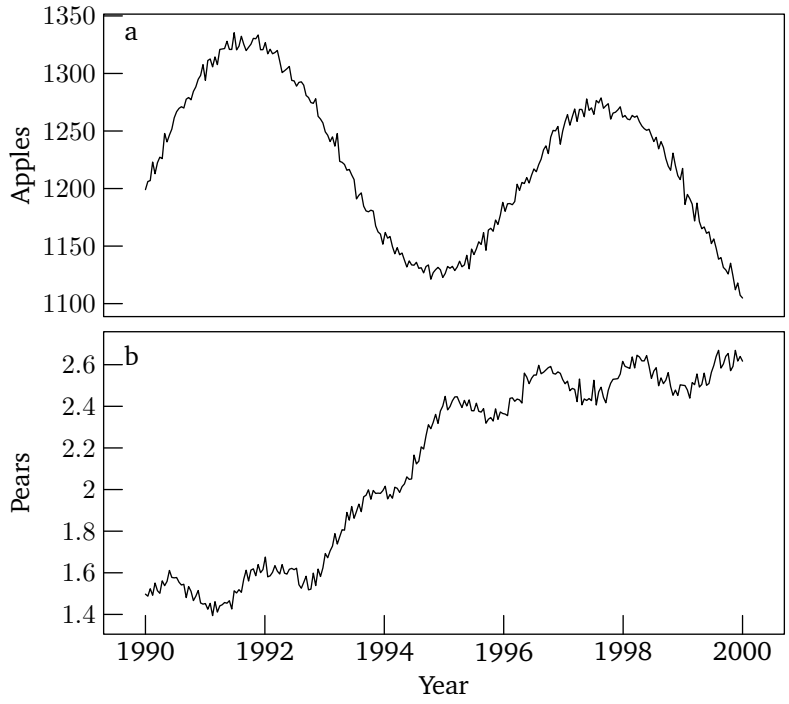


Figure 1. The sample figure with multiple graphs in a single figure. The MetaPost code is shown in the first listing. Note that this not really is about Apples and Pears, the data is just some goniometric functions with some added Gaussian noise.

```
mptopdf file.0
mv file-0.pdf file.pdf
```

This loads the graph package for MetaPost and sets up latex, so that it uses the correct fonts – here we show how to use times, but in the figures themselves, the Maps fonts have been used. After the fonts are changed, the numbers for the graph package have to be initialised again, which is done by the init_numbers function.

```
input graph;

verbatimtex
\documentclass[10pt]{article}
\usepackage{amsmath}
\usepackage{txfonts}
\begin{document}
etex

init_numbers(btex$-$etex, btex$1$etex, btex${\times}10$etex,
  btex${}^-$etex, btex${}^2$etex) ;
```

Start the figure itself. Some variables are declared here: a **pair** corner; to hold the corner of the final image for placing the axis labels, a few numerics to hold the size of the graphs, and an array **picture** thepart[]; to store the parts of the figure. In this example there are only two parts, more parts are left as an exercise for the reader.

```
beginfig(0);
  pair corner;
```

```
numeric width, height, gap; width := 90mm; height := 40mm; gap := 2mm;
picture thepart[];
```

Begin the first part of the figure, for the top of the final figure. A fairly simple graph, all auto scaled and without horizontal markers as they will appear on the bottom graph. More fancy examples for individual graphs can be found in the `mpgraph` manual. The label for this part of the figure is drawn outside the graph routines, so the physical units can be used for positioning the label in the top left corner of the graph.

```
draw begingraph(width,height);
   gdraw "Apples.dat";
   autogrid( ,itick.lft);
endgraph;

label.llft( btex a etex, (2mm, height−1mm) );
```

This is the crucial part: here the current graph is captured into the array, and the currentpicture is wiped to start with the other part of the graph.

```
thepart[1] := currentpicture;
currentpicture := nullpicture;
```

Here we start all over again, drawing the second graph, and storing it in the array. The labels at the bottom are done manually to make sure the full year is used – and not rounded to three digits, as MetaPost might do.

```
draw begingraph(width,height);
   gdraw "Pears.dat";
   autogrid( ,itick.lft);
   itick.bot( btex 1990 etex, 1990); itick.bot( btex 1992 etex, 1992);
   itick.bot( btex 1994 etex, 1994); itick.bot( btex 1996 etex, 1996);
   itick.bot( btex 1998 etex, 1998); itick.bot( btex 2000 etex, 2000);
endgraph;
label.llft( btex b etex, (2mm, height−1mm) );

thepart[2] := currentpicture;
currentpicture := nullpicture;
```

Assemble the graph: draw both parts, with the required gap in between. In this case the last wipe is not needed, as the bottom part will end up in the same location. I would still advise to follow this method, as it avoids many bugs and frustration when you decide later on to add parts.

```
draw thepart[1] shifted (0,height+gap);
draw thepart[2] shifted (0,0);
```

Figure out what the overall lower left corner of the current canvas is.

```
corner := llcorner currentpicture;
```

Draw the axis labels. This is not done using the standard MetaPost graph methods, as they *include* the axis labels in determining the width of the graph. This has the effect of drawing the label off centre. Since we already know the length of the axis, determining the centre is easy. That leaves the offset, which can be determined from the lower left corner of the assembled graph. Note that the labels for the vertical axes are at `height`/2 and 3`height`/2 + `gap`.

The 'Apples' and 'Pears' are smashed, to make sure the baselines are at the same distance from the axis. Yes, I'm aware that this can be done more easily in MetaFun, but I've never managed to get it to work outside of ConTEXt.

```
  label.lft( btex $\smash[b]{\text{Pears}}$ etex rotated 90,
    (xpart corner, height/2) );
  label.lft( btex $\smash[b]{\text{Apples}}$ etex rotated 90,
    (xpart corner, gap + 3*height/2) );
  label.bot( btex Year etex,
    (width/2, ypart corner) );
endfig;
```

Close the LaTEX document used for typesetting the labels.

```
verbatimtex
\end{document}
etex
end
```

I'm sure there are a dozen more solutions, and one may be more suitable than the other, depending on one's needs. This one worked rather well for me.

Error bars in MetaPost graphs

This second part of this note shows how to add error bars to a graph. Many graphs in scientific papers require the use of error indicators. This is the annotated code to produce the graph shown in figure 2. Again, we use the graph module written by John Hobby. Just like the previous example, this one uses LaTEX for its labels.

```
input graph;

verbatimtex
\documentclass[10pt]{article}
\usepackage{amsmath}
\usepackage{txfonts}
\begin{document}
etex

init_numbers(btex$-$etex, btex$1$etex, btex${\times}10$etex,
  btex${}^-$etex, btex${}^2$etex) ;
```

Define a few new macros for producing the error bars themselves. There are three versions: one with vertical error bars (errorbary), one with horizontal error bars (errorbarx) and one with indicators on both directions (errorbarxy). All three work in much the same fashion: define an empty **path** p;, and add a line from $(x, y+\Delta y)$ to $(x, y-\Delta y)$ to this path and draw it. Add the end caps, which are just pictures, and a marker to the data point itself. At the end of the routine, the path is declared

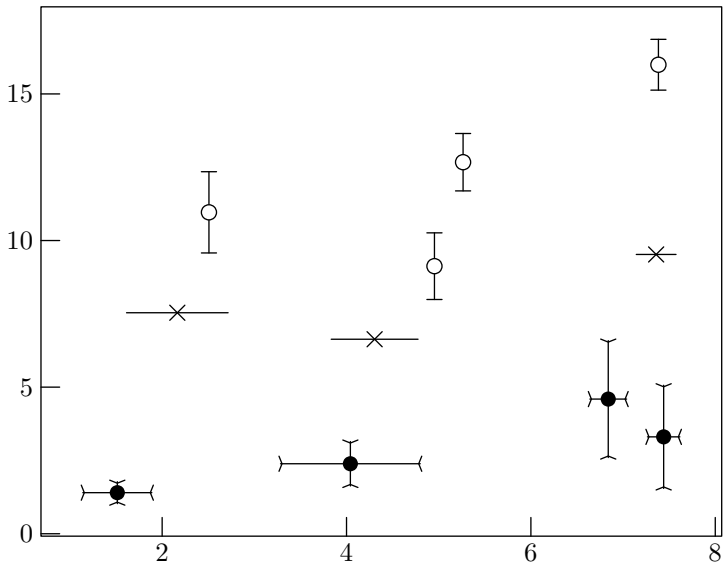


Figure 2. A sample graph with error bars. The MetaPost code is shown in the second listing.

again, which empties the path for the next point. The endcap pictures are rotated, so only a horizontal line is needed in most cases.

The data files are just ASCII text files, with the columns x, y, and Δy in the version with vertical error indicators. The columns are white space separated. The version with horizontal indicators, the columns are x, y, Δx. The final version with indicators on both axes, the order is a bit different: x, Δx, y, and Δy.

```
def errorbary(expr file,marker,endcap) =
  path p;
  gdata( file, s,
   augment.p(scantokens(s1), scantokens(s2)−scantokens(s3));
   augment.p(scantokens(s1), scantokens(s2)+scantokens(s3));
   glabel(endcap, scantokens(s1), scantokens(s2)−scantokens(s3) ) ;
   glabel(endcap rotated 180, scantokens(s1), scantokens(s2)+scantokens(s3));
   gdraw p;
   path p;
   glabel(marker, s1, s2 ) ;
  )
enddef;

def errorbarx(expr file,marker,endcap) =
  path p;
  gdata( file, s,
   augment.p(scantokens(s1)−scantokens(s3), scantokens(s2));
   augment.p(scantokens(s1)+scantokens(s3), scantokens(s2));
   glabel(endcap rotated 270, scantokens(s1)−scantokens(s3), scantokens(s2));
   glabel(endcap rotated 90, scantokens(s1)+scantokens(s3), scantokens(s2));
   gdraw p;
   path p;
   glabel(marker, s1, s2 ) ;
  )
enddef;
```

```
def errorbarxy(expr file,marker,endcap) =
  path p;
  gdata( file, s,
   augment.p(scantokens(s1)−scantokens(s2), scantokens(s3));
   augment.p(scantokens(s1)+scantokens(s2), scantokens(s3));
   glabel(endcap rotated 270, scantokens(s1)−scantokens(s2), scantokens(s3));
   glabel(endcap rotated 90, scantokens(s1)+scantokens(s2), scantokens(s3));
   gdraw p;
   path p;
   augment.p(scantokens(s1), scantokens(s3)−scantokens(s4));
   augment.p(scantokens(s1), scantokens(s3)+scantokens(s4));
   glabel(endcap, scantokens(s1), scantokens(s3)−scantokens(s4) ) ;
   glabel(endcap rotated 180, scantokens(s1), scantokens(s3)+scantokens(s4));
   gdraw p;
   path p;
   glabel(marker, s1, s3 ) ;
  )
enddef;
```

Here we build the graph. We declare some placeholders for the end caps and the marker symbols.

```
beginfig(0);
  numeric width, height;
  width := 90mm; height := 70mm;
  picture marker[], endcap[] ;
  numeric Symbolsize; Symbolsize := 2mm;
```

Create the end caps. Two versions are shown here: a simple line, and a wedge like shape.

```
  draw ((−0.5,0) −− (0.5,0)) scaled Symbolsize ;
  endcap[1] := currentpicture ;
  currentpicture := nullpicture ;

  draw ((−0.5,−0.2) −− origin −− (0.5,−0.2)) scaled Symbolsize ;
  endcap[2] := currentpicture ;
  currentpicture := nullpicture ;
```

Here we create three markers for the three different data sets: an open circle, a closed circle and a diagonal cross.

```
  fill fullcircle scaled Symbolsize withcolor white ;
  draw fullcircle scaled Symbolsize ;
  marker[1] := currentpicture ;
  currentpicture := nullpicture;

  fill fullcircle scaled Symbolsize ;
  marker[2] := currentpicture ;
  currentpicture := nullpicture;

  draw ((−0.5,−0.5) −− (0.5,0.5)) scaled Symbolsize ;
```

```
  draw ((0.5,−0.5) −− (−0.5,0.5)) scaled Symbolsize ;
  marker[3] := currentpicture ;
  currentpicture := nullpicture;
```

Finally: the graph itself. All three modes are shown, with different end caps and markers.

```
  draw begingraph(width,height);
    errorbary("oranges.dat", marker[1], endcap[1])
    errorbarx("oranges.dat", marker[3], nullpicture)
    errorbarxy("oranges.dat", marker[2], endcap[2])

    autogrid(itick.bot,itick.lft);
  endgraph ;
endfig ;
```

Close the LaTeX document used for typesetting the labels.

```
verbatimtex
\end{document}
etex
end
```

(Meta) Post Script

In the article that started this all, Karel Wesseling complains about the manual of the graph package. I can see his point, I'm not a fan of the documentation of metapost in general. There are other manuals, including a nice one written by André Heck: "Learning MetaPost by Doing", a link is provided below. Another manual that contains a lot of general MetaPost advice is the MetaFun manual – although some parts are obviously beyond plain MetaPost.

`http://remote.science.uva.nl/~heck/Courses/mptut.pdf`

Maarten Sneep
Atoom & Laserfysica
Vrije Universiteit
Amsterdam
sneep@nat.vu.nl

Circuit_macros

An application of little languages

Keywords
Electric circuit diagrams, line drawings, LaTeX

Abstract
The evolution of the Circuit_macros package is described, with some of the conventions for drawing circuit elements and some of the lessons learned.

Introduction

Adding diagrams and other artwork to TEX and LATEX documents is a source of guaranteed discussion among users of these tools. There is a large selection of programs capable of creating artwork in .eps form and a somewhat smaller group that invoke the TEX engine as part of the drawing process. The circuit_macros tools are in the second group. This article briefly describes the macros provided in the package, along with some decisions taken during their evolution and some lessons learned.

In 1989 I needed to produce a few simple diagrams, including Figure 1, for a research monograph [1] that I was producing on a basic LATEX installation. From the perspective of the year 2004 it is hard to remember that there were few graphic tools readily available at that time. The picture environment mentioned in the LATEX manual [5] seemed adequate, but the coordinate calculations were tedious, and I automated them by writing an interpreter. I called it dpic, since it implemented a subset of the pic language [4]. The original pic language was developed at a time when a typist would be given hand-written notes with a sketch or two and told to put them "into the computer", so simplicity was its primary feature, together with moderate drawing power. The resulting language is very easy to learn and read.

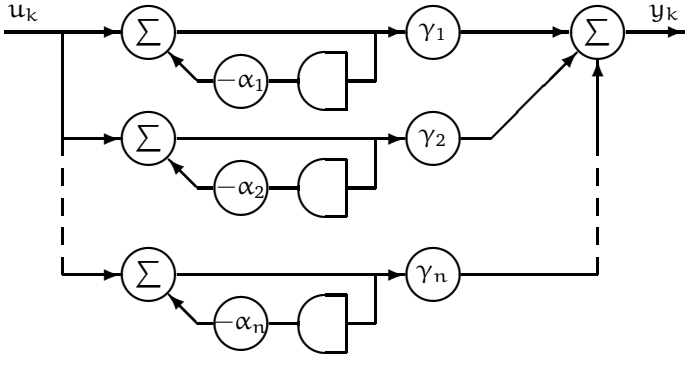


Figure 1. An operational diagram produced by careful use of LATEX picture objects and the epic style.

I also needed to draw basic electric and electronic circuits, but found little support for including them in LaTeX documents. It was a simple step to supplement the basic line drawing of the pic language with circuit-element macros defined in the widely available m4 macro language. However, the LaTeX picture objects produce only limited line slopes and lengths, so alternative processing is required. One solution is the GNU pic processor gpic [6] that generates tpic `\special` commands, but it was also easy to modify dpic to generate output in several formats, of which the PSTricks [7] package seemed to offer the most power and flexibility at the time, at least for producing Postscript files. The current Circuit_macros distribution and the dpic interpreter are the result of those efforts, drawings I have had to produce since, and suggestions received. The macro libraries, example files, and user's manual [2] total approximately 12 000 source lines. The overall philosophy has been to combine the power of proven components implementing "little languages", as encouraged in the Unix computing environment. Thus the workflow has a Unix flavour, but is readily performed on other operating systems, such as a Windows machine with the Cygwin tools installed. A "make" facility, for example, can automate the complete production of a book containing scores of diagrams.

An overview will be given of the macros, together with some of the processing possibilities that the dpic interpreter offers. Element design is emphasized, rather than the details of the pic [6] or m4 [3] languages. For more information on the macros, the user's manual [2], available in any CTAN repository in `graphics/Circuit_macros`, can be consulted.

In the following, keep in mind the pic drawing elements: linear objects (line, arrow, move, spline, arc), and planar objects (box, ellipse, circle). Planar blocks containing arbitrary objects can be defined. The start, centre, and end of linear objects and the compass corners of planar objects can be referenced. Strings to be typeset in a later word-processor step are allowed. Variables beginning with lowercase letters can be assigned values and used in computations. Any drawn element can be given a name beginning with an upper-case letter. The scope of a variable is the block that contains it, but positions can be referenced from outside a block. Pic also has a selection of mathematical functions, as well as looping and other facilities.

A definition in the m4 macro language is of the form `define(‘`*name*`’,` *replacement text*`)`, and the macro is invoked with comma-separated arguments as *name*`(`*arg 1*`,` *arg 2*`, ...)`. Strings are protected from expansion by enclosing them in `‘, ’` quotes, and recursive macro expansion is allowed.

Workflow alternatives

The essentials of the process to be described are shown in Figure 2. The choice of output formats of dpic is provided both for compatibility and necessity.

Dpic without options or with the `-t` option is for producing simple, portable diagrams. The Mfpic output is for compatibility, for creating fonts containing portable diagrams, or as an intermediate step in producing pdf files. The MetaPost output is provided for compatibility and because this is also a good route to producing pdf files on many installations.

The xfig output of dpic allows the definition of detailed circuit elements in m4 and pic form, and their interactive placement using xfig. Xfig can output diagrams in pic format, so an iterative design process is sometimes appropriate.

Some TeX and Metafont programs have fixed-length storage arrays, which can be over-filled by diagrams containing many drawn objects. The .eps format with psfrag strings is an alternative but also occasionally exceeds program capacity. The raw Postscript output of dpic removes any requirement for passing the drawing through LaTeX, but the text labels are not then typeset. Very large or complex diagrams can be produced by creating *two* versions, one containing only graphics converted

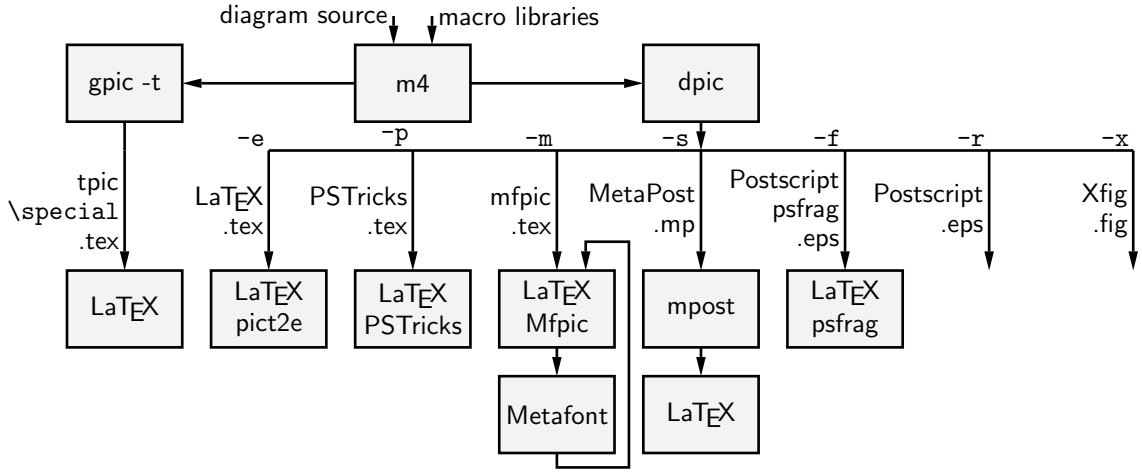


Figure 2. The Circuit_macros workflow. The dpic options and filetypes produced are shown. In some cases, TEX or PDFLatex can be substituted for LaTeX. Strings are not typeset in the raw Postscript or xfig output of dpic.

to .eps form by dpic, and the other containing only labels at appropriate places, overlaying the first diagram in the final document.

For producing Postscript, the PSTricks option has proven both flexible and reliable, and can be assumed by default in the following descriptions. It can be supplemented by `ps4pdf` or sometimes the PDFtricks package.

What is a circuit?

Basic circuits contain two-terminal elements, but multiterminal elements are allowed. In fact, a circuit diagram can be a rather general line diagram together with typeset text. Color, elaborate fills, and visual transformations associated with pictorial graphics are comparatively rare, however. Therefore, almost any line-drawing tool can produce circuits. Regardless of language, effort is required to design high-quality standard elements that are easy to use and modify.

No library of drawing objects can hope to satisfy the needs and taste of everyone, and the potential number of circuit elements is huge. Therefore, the Circuit_macros manual encourages the user to modify and add to the published macros, and some attention has been paid to ease of modification in their design. However, there is always a compromise between macro generality and simplicity.

Two-terminal elements

In the Circuit_macros collection, two-terminal elements are drawn according to a common convention. The first argument of each macro defines an invisible line along which the element is drawn; thus, for example, `inductor(up_ 0.5 from A)` draws an inductor 0.5 units up from predefined position A. All macro arguments are optional, and when the first argument (called a *linespec*) is omitted, a line of default length in the current drawing direction is assumed.

An invisible line that can be named is first drawn as specified by the *linespec*. Then the visible components are added, an invisible block is placed over the element body to facilitate later labeling, and another invisible line is finally drawn to allow post-reference to the element. For example, the command `L1: inductor(,W)` produces the inductor shown in Figure 3, the second argument specifying a “wide” body shape with a default number of loops. The element (the initially drawn invisible line, in fact) has been named L1, so the positions L1.start, L1.center, and L1.end can later be referenced.

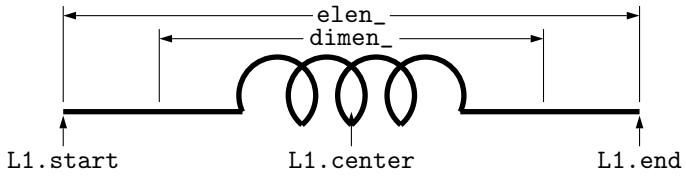


Figure 3. Inductor produced by `L1:inductor(,W)`.

Figure 3 shows the length of the macro `dimen_`, which evaluates to the dpic parameter `linewid`. Element bodies are drawn in units of `dimen_`, so changing its definition or assigning a new value to `linewid` changes the size of elements. The macro `elen_` is the default element length and is initially set to `dimen_*3/2`.

Depending on the element, the arguments may be used to specify element type and other variations. Figure 4 shows the diode variants, for example. In addition to the radiation arrows shown, a variety of arrows and marks is available to signify element variability.

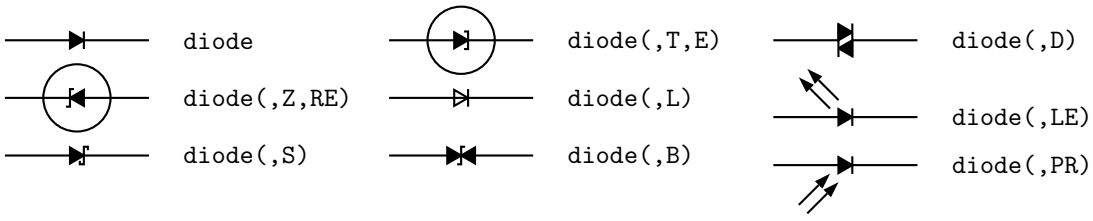


Figure 4. Variants of the macro `diode(`*linespec*`,B|D|L|LE[R]|P[R]|S|T|Z,[R][E])`.

Multiterminal elements

By convention, each multiterminal elements is enclosed in a block. Connection nodes and important sub-elements are named. If a *linespec* is one of the arguments, it defines the reference direction and length of the element but not its position, since the enclosing block is positioned by default or by specifying the location of one of its elements. Thus, for example, `bi_tr with .B at Here` draws a bipolar transistor in the current drawing direction, placed with internal location B (the base connection) at the current drawing location `Here`, which is always defined. Multiterminal elements are often asymmmetric with respect to the drawing direction and typically contain an argument to orient them to the right or left. Thus, Figure 5 shows commands to orient a transistor to left and right, and the resulting diagram.

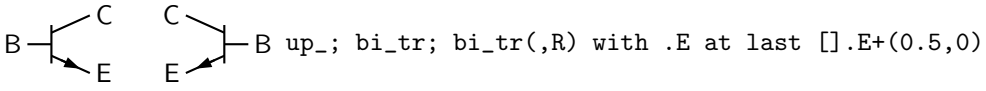
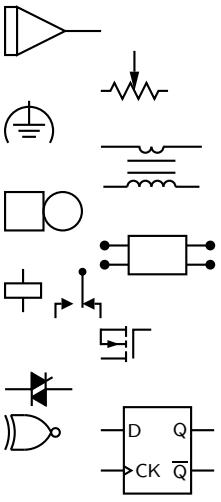


Figure 5. The macro `bi_tr(`*linespec*`,L|R,P,E)` contains internal locations E, B, and C. The figure shows both left and right orientation of the transistor base with respect to the upward drawing direction.

The libraries

In addition to two-terminal elements, the circuit libraries currently contain definitions for the following, with representative samples shown:



- operational amplifiers, delays, and integrators
- potentiometers,
- grounding and protection symbols,
- transformers,
- audio components: speaker, bell, microphone, buzzer, earphone,
- general n-ports,
- contacts and relays,
- bipolar and MOS devices,
- scr devices,
- functional and rectangular logic gates, and
- flip-flops and multiplexors.

The distribution contains examples of flowcharts, binary trees, signal-flow graphs, parallel-line arrows, and integrated circuits. In total, there are approximately 600 macros, a minority of which are meant to be called internally by other macros.

Changing direction

Circuit elements all have a reference direction, and most have an orientation (left or right) with respect to this direction. Most elements can be drawn up, down, left, right or, if required, at any angle. Labels typically should not rotate with the element.

Although pic provides several ways of placing an element, it does not provide arbitrary rotation, which must be performed instead at the macro expansion stage. A transformation matrix similar to the Postscript coordinate transformation matrix is defined each time an element *linespec* or its default is executed, and the element is rotated accordingly. The reference direction can also be set explicitly.

By convention, all macros are designed with respect to a reference direction pointing right (or 0 degrees). Coordinates are specified as `vec_(`*x*`,`*y*`)` and relative coordinates as `rvec_(`*x*`,`*y*`)`. These two macros expand to the required rotated and translated coordinates.

Customizing elements

Recent versions of the libraries have attempted to simplify the customization of defined elements. The diode macro shown in Figure 4 is an example. This macro could simply test its arguments and invoke one of many special diode macros, but this solution was proving cumbersome and hard to maintain. Instead, the diode macro now draws the initial invisible line, sets some internal dimensions, and invokes the internal macro `m4gen_d(`*chars*`)` that draws the details. Its argument is a "DNA-like" character string. This string is tested for a given substring, and if present, the substring is removed, the corresponding element part is drawn, and the process is repeated for other substrings. Thus for example, the argument of `m4gen_d(LCFR)` contains L to draw the left stem of the diode, C to draw the cathode crossbar, F for the filled arrowhead, and R for the right stem. Custom elements are then easy to define by omitting argument substrings or modifying the macros

to test for new substrings and draw the corresponding subcomponents.

Time will tell whether this DNA-like solution is flexible and robust. It provides easy customization but requires the user to understand it.

Interaction with LaTeX and post-processors

Often it is necessary to know the dimensions of typeset text in a figure, but these dimensions are unavailable until after LaTeX is run on the document containing the diagram. This classic forward-referencing problem can be handled just as LaTeX handles other references, by writing definitions to a file and processing twice.

The distribution includes a small style file, `boxdims.sty`, defining a LaTeX macro `\boxdims` that inserts the dimensions of its second argument into the definition of an m4 macro named by the first argument. Such definitions are written to the file *jobname*`.dim`, where *jobname* is the main LaTeX file, or to an explicitly named file. On the second pass, m4 reads in the `.dim` file and, *voilà*, the dimensions of the typeset text are known.

The pic language allows arbitrary strings to be inserted into its output. For example, these strings can command PSTricks, MetaPost, or Postscript to displace, fill, clip, or rotate objects, including their typeset labels.

Comments

The Circuit_macros library has evolved over a decade and a half. Circuits can be (and are) produced in any of several excellent drawing languages now available, although comments I have received testify to the ease of learning pic. Drawing tools are only part of the issue, however; good design requires thought and work no matter what mechanism places the lines on the document.

References

[1] J. D. Aplevich. *Implicit Linear Systems*. Springer Verlag, Heidelberg, 1991. Lecture Notes in Control and Information Sciences, Vol. 152.

[2] J. D. Aplevich. M4 macros for electric circuit diagrams in LaTeX documents, Circuit_macros user's guide, 2004.

[3] B. W. Kernighan and D. M. Richie. The M4 macro processor. Technical report, Bell Laboratories, 1977.

[4] B. W. Kernighan and D. M. Richie. PIC—A graphics language for typesetting, user manual. Technical Report 116, AT&T Bell Laboratories, 1991.

[5] L. Lamport. *LaTeX, A Document Preparation System*. Addison-Wesley, Reading, Mass., 1986.

[6] E. S. Raymond. Making pictures with GNU PIC, 1995. In GNU groff source distribution.

[7] T. Van Zandt. PSTricks user's guide, 1993.

Dwight Aplevich
University of Waterloo Waterloo Canada

Een briefhoofd maken

Abstract
Kort na het succesvol compileren van mijn eerste TEX-document wilde ik graag overstappen op TEX voor zoveel mogelijk documenten. En het idee om het briefhoofd te kunnen typesetten tegelijk met het printklaar maken van de brief zelf leek me erg leuk. Maar dan liefst wel met zo min mogelijk code voor dat briefpapier in de betreffende briefbestanden. Dus met alle nodige commando's in een aparte stylefile. En ook graag met een eenvoudiger briefhoofdje om op vervolgvellen te plaatsen als de brief langer is dan een pagina. Dankzij tips en hulp van Henk de Haan is me dat destijds gelukt en in de loop van de tijd heb ik op dezelfde manier ook anderen hun briefpapier helpen maken.

Phoenix

Als eerste praktijkvoorbeeld beschrijf ik het briefhoofd zoals ik dat zeven jaar geleden voor een vriend in Phoenix maakte. Ik ging uit van de opzet van het briefpapier dat hij tot dan toe gebruikte en van fonts die in elke TEX-distributie voorhanden zijn. Voor dit artikel heb ik de maten van het briefpapier omgezet naar A4-formaat in plaats van het in Amerika gebruikelijke briefpapier dat korter en breder is.

De diverse definities worden bewaard in een stylefile. Voor de paginanummering en voor het zetten van de vervolgvellen maak ik tevens gebruik van de bestaande stylefiles `fancyheadings`, `afterpage` en `letterspacing`.

Bestaande stylefiles zoals deze twee zijn te vinden op CTAN, bijvoorbeeld via de opzoekmachine van http://www.ntg.nl/ctan.html.

`fancyheadings` van Piet van Oostrum staat, zo blijkt daar, op
[CTAN] macros/latex/contrib/fancyhdr/fancyhdr.sty
Waarbij je voor [CTAN] kunt lezen: ftp.dante.de/tex-archive/
`afterpage` van David Carlisle is te vinden op
[CTAN] macros/latex/required/tools/afterpage.dtx
`letterspacing` van Phil Taylor is te vinden op
[CTAN] macros/generic/letterspacing.tex

Letters en cijfers pakken

Om te beginnen heb ik wat lettertjes uitgekozen. Computer Modern (CM) is altijd op voorraad en in diverse boeken staan de aan te roepen fontnamen, bijvoorbeeld in het boek "A Guide to LaTEX" van Helmut Kopka en Patrick W. Daly. De 'gewone' CM schreefletter heet cmr10, de cursieve is cmti10, de kleinkapitaal cmsc10, et cetera. Je kunt naar hartelust pakken wat je wil uit de grote TEX letterbak.

De 10 in de fontnaam slaat op de grootte waarvoor de letter is bedoeld, maar daar trek ik me niet veel van aan: met een simpel commando is de letter precies zoveel op te blazen of te krimpen als ik wil en die specifieke lettergrootte krijgt meteen een eigen naam mee, in dit geval de achternaam van een van de personen die in het briefpapier worden genoemd:

```
\newfont{\reman}{cmcsc10 scaled 1100}
```

Nog een iets kleinere versie van dezelfde letter en een definitie voor het zetten van cijfers:

```
\newfont{\bcert}{cmcsc10 scaled 900}
\newfont{\ciphers}{cmmi10 scaled 900}
```

De mooie 'hangende' cijfertjes die hier worden aangeroepen, zitten min of meer verstopt in de Computer Modern Math Italic. Alsof die cijfers vooral nut zouden hebben binnen cursieve wiskundige reeksen. Vreemd vind ik dat. In elke standaard LaTeX-installatie kun je van alles met fonts, zelfs dingen als `\textsl` voor *halfscheve* tekst wat me nogal overdreven voorkomt, maar de optie voor hangende cijfers binnen gewone tekst is iets exotisch. Wietse Dol stuurde me eens een commando waarmee het wel gewoon werkt:[1]

```
\newcommand{\onum}[1]{\ifmmode\mit{#1}\else$\mit{#1}$\fi}
```

`\onum{1234567890}` geeft dan 1234567890

Het oude heden

Nu ik toch bezig ben met die cijfertjes, kan ik er meteen voor zorgen dat de datum (`\oldtoday`) boven de brief ook wordt gezet zoals het hoort, dus in de nette cijfers:

```
%--------------Old style today----------------------------------%
\def\oldtoday{{\onum{\number\day}} \ifcase\month\or
  januari\or februari\or maart\or april\or mei\or juni\or juli\or
  augustus\or september\or oktober\or november\or december\fi
  \space {\onum{\number\year}}}
%---------------------------------------------------------------%
```

`\oldtoday` wordt nu 4 januari 2005

(Voor het Amerikanse origineel had ik de volgorde anders, eerst de maand en dan de dag.)

En vooruit, dan ook de juiste tijd, waarbij `\now` de tijd in tabelcijfers geeft en `\oldnow` de tijd geeft in oldstyle. Punt van zorg is wel dat er bij mij wat ongevraagde ruimte zit rond de dubbele punt tussen uren en minuten. Geen idee hoe die er tussen komt of hoe ik die weghaal:

```
\newcount\hh \newcount\mm
\hh=\time\divide\hh by 60
\mm=\hh \multiply\mm by 60 \mm=-\mm
\advance\mm by \time
\def\now{\number\hh:\ifnum\mm<10{}0\fi\number\mm}
\def\oldnow{{\onum{\number\hh:\ifnum\mm<10{}0\fi\number\mm}}}
```

`\oldnow` wordt nu 11:23

Picture zonder plaatjes

Nu de letters en de cijfers klaar liggen komen we toe aan het eigenlijke briefhoofd.

Om de letters en cijfers exact daar te zetten waar ik ze wil hebben, gebruik ik de `picture`-omgeving. Om het hele briefhoofd straks zonder ingewikkelde poespas in een kort commando te kunnen plaatsen, noem ik de set opdrachten daartoe `\kop`. En voordat ik die `picture` ga beschrijven, stel ik vast dat we bij het schuiven met de tekens denken in stapjes van een millimeter:

```
\newcommand{\kop}{\setlength{\unitlength}{1mm}
```

Het commando om tekst ergens te plaatsen is put en daarachter wordt tussen haakjes gezet hoeveel stapjes naar rechts, links, omhoog of omlaag; dat moet vanaf een vast nulpunt. Ik begin meestal gewoon met het eerste woord, zet dat ergens neer waar ik al eens wat heb neergezet en daarna voeg ik de rest toe. Het alles op zijn plaats schuiven komt daarna wel. Als het gecentreerd moet staan dan print ik een voorbeeld, ik vouw de pagina verticaal en met een lineaal kijk ik hoever de woord- en cijfergroepjes moeten worden verschoven.

In de onderstaande regel komt de reeks een millimeter meer naar rechts te staan als ik van de 33 een 34 maak, en een millimeter lager als ik de 36 verander in 35. Je kunt ook in kleinere stapjes werken: als je van 33 mm een honderdste meer wil maken dat schrijf je 33.01 – voldoende mogelijkheden voor fijnafstelling dus! De tekst komt te staan in de letter die ik \reman heb genoemd. De afkortingen zijn van die typisch Amerikaanse kwalificaties voor beroepsgroepen.

```
\put(33 , 36){{\reman marcia reman scialli, m.s.w., l.c.s.w.}}
```

Een streepje van 42,5 mm en een half puntje dik gaat zo:

```
\put(47 , 30){\rule[0pt]{42.5mm}{.5pt}}
```

Nu dan de hele \picture – onderaan een regel met de bekende bolletjes waar ook items bij itemize vaak mee beginnen en een mix van cijfers en letters:

```
\newcommand{\kop}{\setlength{\unitlength}{1mm}
   \begin{picture}(0 ,0)
   \put(33 , 36){{\reman marcia reman scialli, m.s.w., l.c.s.w.}}
   \put(47 , 33){{\bcert board certified diplomate}}
   \put(47 , 30){\rule[0pt]{42.5mm}{.5pt}}
  \put(27 , 25){{\reman john v. scialli, m.d., f.a.a.c.a.p., d.f.a.p.a.}}
   \put(55 , 22){{\bcert board certified}}
   \put(55 , 19){{\bcert adult psychiatry}}
   \put(55 , 16){{\bcert child psychiatry}}
   \put(44 ,-232){\rule[0pt]{4.5cm}{.3pt}}
   \put(24 ,-235){{\bcert%
             Individual, Couple, Family Therapy $\bullet$ Consultation}}
   \put(2  ,-240){{\bcert%
             {\ciphers 4647} North {\ciphers 32}nd Street, Suite\ %
             {\ciphers 260}\ %
             $\bullet$\ %
             Phoenix, Arizona\ %
             {\ciphers 85018}-{\ciphers 3347}\ %
             $\bullet$\ %
             ({\ciphers 602})\ %
             {\ciphers 224}-{\ciphers 9888}
             }}
   \end{picture}}
```

Midden onderaan de pagina (center, foot... cfoot) komt het paginanummer in oldstyle. Om dat voor elkaar te krijgen maak ik gebruik van het fancyheadings pakket:

```
\RequirePackage{fancyhdr}
\renewcommand{\headrulewidth}{0pt}
\renewcommand{\footrulewidth}{0pt}
\lhead{\ }          \chead{ }                \rhead{\ }
\lfoot[ ]{ }        \cfoot{{\onum{\thepage}}} \rfoot[ ]{ }
```

Hetzelfde pakket zorgt ervoor dat je op vervolgvellen een beknopte versie van het briefhoofd kwijt kunt. Ook nu weer een `picture`-omgeving maar dan midden bovenin geplaatst: `chead`.

```
\newcommand{\morepages}{\afterpage
   {\chead{
   \setlength{\unitlength}{1mm}
   \begin{picture}(0 ,0)
   \put(-31 , 16){{\reman marcia reman scialli, m.s.w., l.c.s.w}}
   \put(-17 , 13){\rule[0pt]{42.5mm}{.5pt}}
  \put(-37 , 8){{\reman john v. scialli, m.d., f.a.a.c.a.p., d.f.a.p.a.}}
   \end{picture}
  }}}
```

Overzichtelijk

Als de meeste van bovengenoemde commando's en definities in een stylefile worden gezet (`john.sty`, deze plaats ik in zijn geheel onderaan in dit artikel) dan blijft de brief zelf vrij overzichtelijk:

```
\documentclass[11pt]{artikel3}
\usepackage[english]{babel}
\usepackage{a4,fancyhdr,john}
\pagestyle{fancy}
\begin{document}
\kop

Phoenix, \oldtoday

Dear reader,\\

At present the time is \oldnow\ and the quick brown fox does a
great day of jumping over a kennel of lazy dogs.

\morepages
\newpage

Ah, this letter spans more than one page!

Time to call in the help of \verb|morepages|
to typeset the header there as well.

\end{document}
```

Het commando `\morepages` moet worden gegeven na de eerste woorden van het document en voordat de tweede pagina is begonnen. Da's wel wat omslachtig.

Het breed hebben

In mijn eigen briefhoofd heb ik mijn naam iets breder gezet. Nu had ik de letters stuk voor stuk kunnen plaatsen, of ik had zelf tussen elke letter een commando voor een stukje witruimte kunnen zetten. Dat was wellicht beter geweest om zo minder afhankelijk te zijn van weer een andere stylefile, maar ik heb ervoor gekozen om gebruik te maken van het pakket `letterspacing` (ik heb de betreffende commando's in een `lettersp.sty` gezet) van Phil Taylor.

```
\newfont{\topkopletter}{cmr12 scaled 1100}
%[...]
 \newcommand{\kop}{\setlength{\unitlength}{1mm}
   \begin{picture}(0 ,0)
   \put(50, 25){{\letterspace spread 0.2 \naturalwidth {\topkopletter
          Frans Goddijn}}}
%[...]
   \end{picture}}
```

Frans Goddijn wordt dan Frans Goddijn

Omcirkelde letter: logo

Voor het logo in het briefhoofd van een stichting was het nodig een cirkel rond een letter te trekken. Die letter was oorspronkelijk de hoofdletter P uit de palatino maar nadat die letter was 'kwijtgeraakt' bij het overschakelen naar een nieuwe TEX-installatie waarbij de font definition files, map-files en dergelijke zodanig waren verbeterd dat het niet meer werkte, heb ik er maar gewoon een Computer Modern van gemaakt.

Het plaatsen van zo'n logo heeft wat meer voeten in aarde. Gelukkig waren er behulpzame TEX-krachten die me hierbij de weg wezen. Hieronder wordt eerst de letter gekozen en van de naam `PeeHuge` voorzien. Dan wordt er een aparte 'letterdoos' gemaakt met daarin de omschrijving van de omcirkelde letter, ook weer in een `picture`-omgeving. Vervolgens wordt er een opdracht gedefinieerd waarmee de doos wordt geopend en de inhoud tevoorschijn komt en dat commando vindt zijn plaats in de grotere `picture`-omgeving zoals we al eerder hebben gemaakt.

```
\newfont{\PeeHuge}{cmr10 scaled 2400}

\setlength{\unitlength}{1mm}
\newbox\vignet
\setbox\vignet=\hbox{
  \begin{picture}(0 ,0)
    \put(-2.5 ,16){\circle{10}}
    \put(-4.8 ,12.8){\PeeHuge P}% x,y
  \end{picture}}

\newcommand{\pcirkel}{\unhcopy\vignet}

\newcommand{\kop}{%
   \setlength{\unitlength}{1mm}
  \begin{picture}(0 ,0)
   \put(67 ,14){\pcirkel}
%[...]
  \end{picture}}
```

Vensterenvelop

Omdat ik veel gebruik maak van vensterenveloppen wilde ik de mogelijkheid hebben van het commando `briefkop` met daar direct achter naam en adres van de ontvanger:

```
\briefkop{Robert den Hartog\\Stationsweg 46\\6863 EP Arnhem}
```

Het commando `briefkop` opent een `minipage` met daarin een van tevoren door mij gedefinieerde afzender `\postadres` en ruimte voor het argument van `\briefkop`: het adres van de ontvanger. De diverse `hspace` en `vspace` zijn de bekende schuifregelaars om de boel te kunnen aanpassen als er andere enveloppen komen met weer wat andere vensterafmetingen.

```
\newcommand{\briefkop}[1]{\hspace*{-20mm}%
\begin{minipage}[t]{8.2cm}
\vspace*{12mm}
\vspace*{-2.5mm}\hfill{\underline{\postadres}}\newline
\vspace{.6mm}{\footnotesize \textbf{#1}}
\end{minipage}
```

john.sty

```
\newfont{\reman}{cmcsc10 scaled 1100}
\newfont{\bcert}{cmcsc10 scaled 900}
\newfont{\ciphers}{cmmi10 scaled 900}

\RequirePackage{afterpage}

\newcommand{\onum}[1]{\ifmmode\mit{#1}\else$\mit{#1}$\fi}

%--------------Old style today----------------------------------%
%% \oldtoday provides the date with oldstyle numbers
%%
\def\oldtoday{%
  \ifcase\month\or january\or february\or march\or april\or
  may\or june\or july\or august\or september\or october\or
  november\or december\fi \space
  {\onum{\number\day}}
  {\onum{\number\year}}}
%-----------------------------------------------------------------%

% -------------Old style \now-----------------------------------%
%% \oldnow provides time in old style numbers
% \now macro levert met \now de tijd in hh:mm
% \oldnow in oldstyle numbers
\newcount\hh \newcount\mm
\hh=\time\divide\hh by 60
\mm=\hh \multiply\mm by 60 \mm=-\mm
\advance\mm by \time
\def\now{\number\hh:\ifnum\mm<10{}0\fi\number\mm}
\def\oldnow{{\onum{\number\hh:\ifnum\mm<10{}0\fi\number\mm}}}

\newcommand{\kop}{\setlength{\unitlength}{1mm}
   \begin{picture}(0 ,0)
```

```
   \put(33 , 36){{\reman marcia reman scialli, m.s.w., l.c.s.w.}}
   \put(47 , 33){{\bcert board certified diplomate}}
   \put(47 , 30){\rule[0pt]{42.5mm}{.5pt}}
  \put(27 , 25){{\reman john v. scialli, m.d., f.a.a.c.a.p., d.f.a.p.a.}}
   \put(55 , 22){{\bcert board certified}}
   \put(55 , 19){{\bcert adult psychiatry}}
   \put(55 , 16){{\bcert child psychiatry}}
   \put(44 ,-232){\rule[0pt]{4.5cm}{.3pt}}
   \put(24 ,-235){{\bcert%
             Individual, Couple, Family Therapy $\bullet$ Consultation}}
   \put(2  ,-240){{\bcert%
             {\ciphers 4647} North {\ciphers 32}nd Street, Suite\ %
             {\ciphers 260}\ %
             $\bullet$\ %
             Phoenix, Arizona\ %
             {\ciphers 85018}-{\ciphers 3347}\ %
             $\bullet$\ %
             ({\ciphers 602})\ %
             {\ciphers 224}-{\ciphers 9888}
             }}
   \end{picture}}

\RequirePackage{fancyhdr}
\renewcommand{\headrulewidth}{0pt}
\renewcommand{\footrulewidth}{0pt}
\lhead{\ }          \chead{ }                \rhead{\ }
\lfoot[ ]{ }        \cfoot{{\onum{\thepage}}} \rfoot[ ]{ }

\newcommand{\morepages}{\afterpage
   {\chead{
   \setlength{\unitlength}{1mm}
   \begin{picture}(0 ,0)
   \put(-31 , 16){{\reman marcia reman scialli, m.s.w., l.c.s.w}}
   \put(-17 , 13){\rule[0pt]{42.5mm}{.5pt}}
  \put(-37 , 8){{\reman john v. scialli, m.d., f.a.a.c.a.p., d.f.a.p.a.}}
   \end{picture}
  }}}
```

Notes

1. Noot v.d. redactie: dit commando werkte niet binnen de Maps stijl, omdat wiskunde in de Maps niet in Computer Modern maar in Euler fonts wordt gezet. Een alternatieve definitie is gebruikt om het voorbeeld correct te kunnen tonen.

Frans Goddijn
frans@goddijn.com

MetaPlot, MetaContour, and Other Collaborations with MetaPost

Abstract
Most methods of creating plots in MetaPost work by doing all of their calculations in MetaPost, or by doing all of their calculations in a preprocessing program. There are advantages to dividing the work more equitably by doing the mathematical and data-visualization calculations in a preprocessing program and doing the graphical and layout calculations in MetaPost. The MetaPlot package provides a standard, flexible, interface for accomplishing such a collaboration between programs, and includes a general-purpose set of formatting macros that are applicable to a wide range of plot types. Examples are shown of linear plots with idiosyncratic annotation and two-dimensional contour plots with lines and filled contours on a non-cartesian mesh.

Introduction

One of the challenges of scientific writing in TEX (or in LATEX) is producing figures that are of comparable quality to the typesetting. These figures often include plots and graphs that represent mathematically-intense visualization of large data files, implying that some form of specialized program must be used to create them. They also typically contain labels, notes, and other text that should be typeset in a manner consistent the rest of the document, which requires using TEX's typesetting engine.

Traditionally, programs that meet these goals have taken one of two approaches. The first approach, used by programs such as ePiX[1] and Gnuplot[2], is to implement the program in a "traditional" programming language such as C++ or FORTRAN, and produce the complete figure as output in TEX/eepic or MetaPost code, which is then postprocessed. The other approach, taken by MetaPost's `graph` package and m3D[3], is to implement the program directly in MetaPost's macro language.

There are advantages and tradeoffs related to both of these approaches. Programming in MetaPost allows one to work directly with the language features such as declarative equations and ability to measure the size of typeset text, and thus allow the user to specify the figure layout in an intuitive, simple, and flexible manner. However, programming in a traditional language allows one to write mathematically-intensive programs that use floating-point numbers and can be compiled rather than run slowly through an interpreter; in addition, it may allow one to take advantage of existing visualization libraries, or to provide an interactive user interface.

This paper describes an intermediate approach, which combines benefits from MetaPost programs and programs in more traditional languages. The initial data processing is done with a program written in a traditional language, which produces a MetaPost output file containing the processed data in an encapsulated form. This processed data is then fed into a set of MetaPost formatting macros, and the scaling, drawing, and annotation of the plots is all done by user-written commands within MetaPost.

Creating plots in two steps in this manner has several advantages: The initial data visualization can be done in a special-purpose program that uses a programming language and code libraries intended for substantial computations, without the need to implement more than a very simple output routine; the MetaPost macros for formatting plots and arranging them within a figure are largely independent of the details of the plots they are working with, and can be written in a generic manner suitable for widespread distribution; and the layout of any given figure can be done using the same processes as for a native-MetaPost drawing.

A Simple Example

Consider, by way of example, a plot of the shape of a meniscus formed by a liquid surface meeting a solid wall as shown in Figure 1. The surface curve is given by a somewhat complicated expression involving inverse hyperbolic cosines,[1] and is representative of calculations that would be easier to do with a traditional programming language.

The C++ program to produce this curve in a MetaPost format is straightforward. The most complicated part is the function to generate a string containing a

MetaPost representation of a point, which we accomplish using the <*sstream*> standard library.

```
string mpoint(double x, double y) {
  ostringstream pointstring;
  pointstring.setf(ios_base::fixed,
                   ios_base::floatfield);
  pointstring.precision(5);
  pointstring << '(' << x << ',' << y << ')';
  return pointstring.str();
}
```

The **setf** and **precision** commands set the numeric format for the stream (fixed-precision, five decimal places), and then the coordinates are fed into the stringstream with the appropriate punctuation, producing a result like `(0.01556,0.75006)`.

Given this and a *capillary*() function that computes the equation for the surface, creating the MetaPost command for the curve is simply a matter of looping through the points and dumping them to the standard output, with appropriate text before and after the loop to define the picture variable and close the curve into a cyclic path.

```
int main() {
  double theta = pi/4.0;
  double d = 1.0;
  double h = sqrt(2.0 * d*d * (1.0 - sin(theta)));
  double y, z;

  cout << "picture capillary;\n";
  cout << "capillary := nullpicture;\n";
  cout << "addto capillary contour "
       << mpoint(0.0, h);
  for(int i = 99; i > 2; i-- ) {
    z = (i/100.0) * h;
    y = capillary(z,h,d);
    cout << " .. " << mpoint(y, z);
  }
  cout << " -- " << mpoint(y, -0.5);
  cout << " -- " << mpoint(0.0, -0.5);
  cout << " -- cycle;\n";
}
```

This produces the following MetaPost code as output:

```
picture capillary; capillary := nullpicture;
addto capillary contour (0.00000,0.76537)
.. (0.00772,0.75771) .. (0.01556,0.75006)
        % [...and so forth...]
.. (3.39322,0.02296) -- (3.39322,-0.50000)
-- (0.00000,-0.50000) -- cycle;
```

We can then follow this with additional MetaPost commands, which scale it to an appropriate size for printing on the page and draw axes and labels on it, in order to produce the plot shown in Figure 1.

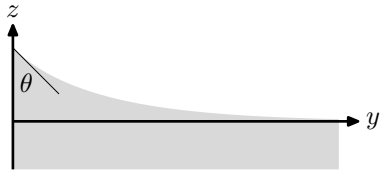


Figure 1. A capillary surface on a liquid touching a solid wall, after Batchelor [4].

```
beginfig(1)
 draw (capillary scaled 0.5in) withcolor
     0.85white;
 linecap := butt;
 pickup pencircle scaled 1pt;
 drawarrow (0,−0.25in) -- (0, 0.5in);
 label.top(btex $z$ etex,(0, 0.5in));
 x1 := (xpart(lrcorner capillary) * 0.5in, 0)
       + (0.1in, 0);
 drawarrow (0,0) -- x1;
 label.rt(btex $y$ etex, x1);
 pickup pencircle scaled 0.25pt;
 x2 := ulcorner capillary scaled 0.5in;
 draw ((0,0) -- (0.24in, −0.24in)) shifted x2;
 label(btex $\theta$ etex,
   x2 + (0.07in, −0.18in)));
endfig;
end
```

Although this example produces a perfectly serviceable result, it has some noteworthy drawbacks. The scale factor of 0.5in does not have a clear relationship to the size of the plot, and producing a plot of a particular size would require measurement of the *capillary* picture and explicit computation of the scale factor. The locations of the annotations are likewise determined by explicit measurement, or by being typed in directly. If we were to change one of the parameters in the C++ program and re-run it, many of the values in the MetaPost code would need to be changed as well.

A more general example: the MetaPlot package

The MetaPlot package is designed to address many of the shortcomings of the example given in Section 2. It provides a consistent way of transferring the plot commands and associated metadata from the generating program into MetaPost, and direct handles for manipulating the plots within MetaPost using its normal idiom of declarative equations rather than procedural assignments.

To accomplish this in a general manner, we define two types of MetaPost data structures: *plot objects* and *plot instances*. A plot object is a plot "in the abstract", containing paths, filled contours, and metadata that make up the plot (or a set of related plots), represen-

ted in a manner that is independent of the details of how the plot is positioned. By contrast, a plot instance is a plot "on the page," containing parameters for the scaling and positioning of a given plot, and a reference to a parent plot object that gives the actual pictures to be drawn.

A typical preamble for a figure using MetaPlot will consist of an **input** `metaplot` command to load the MetaPlot macros, an **input** command to load the MetaPost file that contains the plot objects (typically an output file from the preprocessing program), and calls to the MetaPlot macros to generate plot instances from the plot objects.

The concept of a "plot-object"

Suffix arguments and multi-token variable names in MetaPost allow us to define data structures that approximate structures or objects in more traditional programming. The correspondence is not exact; in particular, there is no data type associated with the overall object. MetaPost is simply passing around a fragment of a variable name and constructing complete variable names from it, so any arbitrary element can be added to the class without changing its type. Thus, the MetaPlot macros can deal with arbitrary types of plots in a generic manner, so long as they meet a few minimal requirements that allow them to be scaled and positioned.

The paths and contours that make up a plot object are not defined in terms of the native data coordinates, but are rescaled to fit within a unit box (that is, extending from 0 to 1 in both coordinate directions), which is treated as the bounding box of the plot for purposes of scaling and positioning. As a result, the possibility of coordinates too small or too large for MetaPost's fixed-point number representation is avoided; in addition, positioning the plot on the page is a simple matter of scaling by the final width and height and shifting by the final position of the lower-left corner. The original data scales are stored in four numeric components that record the values corresponding to the extents of the bounding box.[2]

The remaining details of the format can be shown by rearranging the example from Section 2 into a plot object, as follows. For purposes of later examples, we will presume that this has been saved as `capillary.mp`.

```
% Definition of capillary plot−object
% Picture components
picture capillary.fplot;
  capillary.fplot := nullpicture;
  addto capillary.fplot contour (0.00000,1.00000)
  .. (0.00227,0.99395) .. (0.00459,0.98790)
        % [...and so forth...]
  .. (1.00000,0.41329) -- (1.00000, 0.00000)
  -- (0.00000, 0.00000) -- cycle;
picture capillary.lplot;
  capillary.lplot := nullpicture;
  addto capillary.lplot doublepath
  (0.00000,1.00000) .. (0.00227,0.99395)
        % [...and so forth...]
  .. (1.00000,0.41329);

% Required metadata
numeric capillary.xleft; capillary.xleft = 0.0;
numeric capillary.xright;
  capillary.xright = 3.39322;
numeric capillary.ybot; capillary.ybot = −0.5;
numeric capillary.ytop; capillary.ytop = 0.76537;

% Other metadata
pair capillary.contactpoint;
  capillary.contactpoint = (0.0, 1.0);
numeric capillary.contactangle;
  capillary.contactangle = 45.0;
```

In this case, I have also added an additional component: this version of *capillary* contains a path for the liquid surface line (*capillary.lplot*), as well as the original filled contour (now *capillary.fplot*); the decision about which of them to draw can be made later. A plot object can contain any number of these pictures (even zero), with arbitrary names.

The four required scale variables are *capillary. xleft*, *.xright*, *.ybot*, and *.ytop*; these, for purposes of the MetaPlot macros, must be named thus.

Finally, there are two metadata variables, *capillary.contactpoint* and *capillary.contactangle*, which will be useful in drawing the annotations on this particular plot. These, again can be present in any number, and have arbitrary names. Of note is that *.contactpoint* is given in the same unit-box coordinate system that the paths and contours are in, allowing it to be positioned by the same macros that scale and position the picture components.

Creation of a plot instance

The next step after creating plot objects is manipulating them on the page by means of plot instances. A plot instance thus needs to contain three sets of components: coordinates and dimensions of the plot as shown on the page, a representation of the plot's internal scale for use in alignment and producing axes, and a means of accessing picture components from its parent plot object. These are created by the *plot_instantiate*() macro, which is part of MetaPlot; the version below is simplified somewhat.

```
% Args: inst is the new plot instance.
% plot_object is the parent plot object.
def plot_instantiate(suffix inst)
  (suffix plot_object) =
```

```
% Define (unknown) parameters for plot—instance
% location on page
  numeric inst.pagewidth, inst.pageheight;
  numeric inst.pageleft, inst.pageright,
          inst.pagetop, inst.pagebottom;
  inst.pageleft + inst.pagewidth = inst.pageright;
  inst.pagebottom + inst.pageheight = inst.pagetop;

% Define (known) parameters for plot's scaling
  numeric inst.scaleleft, inst.scaleright,
          inst.scaletop, inst.scalebottom;
  inst.scaleleft := plot_object.xleft;
  inst.scaleright := plot_object.xright;
  inst.scalebottom := plot_object.ybottom;
  inst.scaletop := plot_object.ytop;

% Pointer—function to plot_object's plots, scaled
% and positioned.
  vardef inst.plot(suffix name) =
    plot_object.name xscaled inst.pagewidth
      yscaled inst.pageheight
      shifted (inst.pageleft, inst.pagebottom)
  enddef;
enddef;
```

Note that, immediately after a plot instance is created, the page information is unknown and the scale information is known.

We can now start putting plot objects on the page in a limited fashion, by assigning known values to the unknown page information, and then drawing the scaled picture elements.

```
input metaplot     % MetaPlot macros
input capillary    % capillary plot object

plot_instantiate(plotA, capillary)
plotA.pageleft = 0.0;
plotA.pagebottom = 0.0;
plotA.pagewidth = 2.0in;
plotA.pageheight = 0.75in;
beginfig(2)
  draw plotA.plot(fplot) withcolor 0.85white;
  draw plotA.plot(lplot)
    withpen pencircle scaled 1pt;
endfig;
end
```

The result of this is shown in Figure 2. Note that the color of the filled plot and the line size for the line plot are specified in the draw command, rather than in the plot object.

Manipulation of plot-objects

The bare plot instances are of little use without a set of macros for manipulating them. We start with a macro to set the x-axis and y-axis scales to equal values:

```
def plot_setequalaxes(suffix inst) =
  inst.pagewidth = inst.pageheight
    * ((inst.scaleright − inst.scaleleft)
        / (inst.scaletop − inst.scalebottom));
enddef;
```

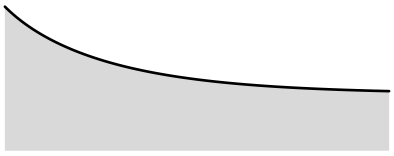


Figure 2. The capillary surface, in its unadorned form as plot object elements scaled to 2.0in by 0.75in.

This is written so that the page-related variables do not appear in the denominator of fractions, because either one (or both) of them may be unknown when the macro is called, and MetaPost can only solve linear equations.

There are also a set of macros for converting between locations expressed in the plot's coordinates and locations on the page. For example,

```
def plot_xpageloc(suffix inst)(expr scalex) =
  inst.pageleft + (scalex − inst.scaleleft)
    * (inst.pagewidth
        / (inst.scaleright − inst.scaleleft));
enddef;
```

The additional macros in this series are *ypageloc*, *zpageloc* (which takes an x and a y coordinate as input, and returns a point), and *xscaleloc* and *yscaleloc* for the reverse direction of converting from a page location to a plot coordinate.

With these, we have most of what we need to manipulate plots in an intuitive way. For instance, consider the figure from Section 2, which can now (with some small changes) be written in a much more general way as

```
input metaplot     % MetaPlot macros
input capillary    % capillary plot object

plot_instantiate(plotB, capillary)
plot_setequalaxes(plotB);
plotB.pageleft = 0.0;
plotB.pagebottom = 0.0;
plotB.pageheight = 0.75in;
beginfig(3)
  draw plotB.plot(fplot) withcolor 0.85white;
  linecap := butt;
  pickup pencircle scaled 1pt;
  % z—axis (vertical)
  z1 = (plotB.pageleft, plotB.pagebottom);
  z2 = (plotB.pageleft, plotB.pagetop + 0.1in);
  % y—axis (horizontal)
  z3 = (plotB.pageleft, plot_ypageloc(plotB,0.0));
  z4 = (plotB.pageright + 0.1in,
        plot_ypageloc(plotB,0.0));
```

```
  drawarrow z1 -- z2;
  label.top(btex $z$ etex, z2);
  drawarrow z3 -- z4;
  label.rt(btex $y$ etex, z4);
  pickup pencircle scaled 0.25pt;
  % Label for contact angle
  z5 = plotB.plot(contactpoint);
  z6 = z5 + 0.24in
       * dir(-90 + capillary.contactangle);
  z7 = z5 + 0.18in
       * dir(-90 + 0.5*capillary.contactangle);
  draw z5 -- z6;
  label(btex $\theta$ etex, z7);
endfig;
end
```

The result of this is shown in Figure 3. We can demonstrate that this is flexible by adjusting the value of θ to $\pi/6$ rather than $\pi/4$, and recreating the figure using exactly the same files; the result is shown in Figure 4. Note that changing the contact angle raises the contact point, making the plot taller in scale coordinates; thus, it is drawn at a smaller scale to maintain the 0.75-inch page height.

Having two figures in this way is not the clearest way to compare the two plots, particularly with the differences in scale. A better approach is to overlay them at the same scale, making use of the existence of the filled plot from one plot object and the line plot from the other to provide a visually clear result. A simple way of placing both plots on the same coordinate axes is to require that their (0,0) and (1,1) points coincide on the page, which we do by means of the *plot_zpageloc* command; the remainder of the file is as much in the previous plots, although there is a little additional code in making certain that the axis-arrows cover both plots.

```
input metaplot      % MetaPlot macros
input capillary     % capillary plot object
input capillary2    % capillaryb plot object

plot_instantiate(plotB, capillary)
plot_setequalaxes(plotB);
plotB.pageleft = 0.0;
plotB.pagebottom = 0.0;
plotB.pageheight = 0.75in;

plot_instantiate(plotC, capillaryb)
plot_zpageloc(plotB, 0.0, 0.0)
  = plot_zpageloc(plotC, 0.0, 0.0);
plot_zpageloc(plotB, 1.0, 1.0)
  = plot_zpageloc(plotC, 1.0, 1.0);

beginfig(5)
  linecap := butt;
  pickup pencircle scaled 1pt;
  draw plotB.plot(fplot) withcolor 0.85white;
```

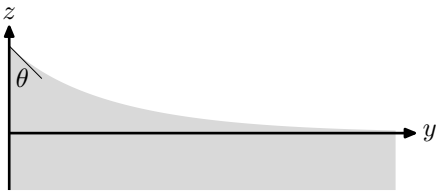


Figure 3. The capillary surface, with equal y and z scales, a page height of 0.75in, and appropriate annotations.

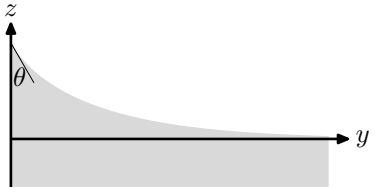


Figure 4. The capillary surface with parameters and page height as in Figure 3, but with $\theta = \pi/6$.

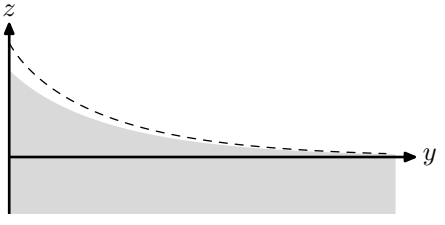


Figure 5. Two capillary surfaces, as in Figure 3 and Figure 4, showing the difference in the curves as a result of varying θ.

```
  draw plotC.plot(lplot) dashed evenly
    withpen pencircle scaled 0.5pt;
  % z-axis (vertical)
  z1 = (plotB.pageleft, plotB.pagebottom);
  x2 = plotB.pageleft;
  y2 = max(plotB.pagetop, plotC.pagetop) + 0.1in;
  % y-axis (horizontal)
  z3 = (plotB.pageleft, plot_ypageloc(plotB,0.0));
  x4 = max(plotB.pageright, plotC.pageright) + 0.1in;
  y4 = plot_ypageloc(plotB,0.0);
  drawarrow z1 -- z2;
  label.top(btex $z$ etex, z2);
  drawarrow z3 -- z4;
  label.rt(btex $y$ etex, z4);
endfig;
end
```

The result of this is shown in Figure 5.

Creation of axes

Any quantitative graph is meaningless without grid-labels for the coordinate axes, and so MetaPlot includes macros to create them. Unlike MetaPost's `graph.mp` package, MetaPlot's axis-drawing function-

ality requires that the user specify most of the details of the formatting, with the benefit of having a much more flexible implementation.[3]

The core of the axis-drawing functionality is a set of macros for creating generic tickmarks, labeled tickmarks, rows of tickmarks, and so forth, which are included with MetaPlot in a `axes.mp` file (and thus, for consistency, are prefaced with *axes_* rather than *plot_*). These are interfaced to the plot object coordinates by the *plot_xtickscale* and *plot_ytickscale* macros.

```
def plot_xtickscale (suffix inst)
  (expr startpoint, endpoint, ticklength, tickspace,
   tickdir, tickzero, tickstep, ticklabelformat) =

 axes_tickscale (
   startpoint, % First endpoint of the tickrow
   endpoint, % Second endpoint of the tickrow
   ticklength, % Length of tickmarks
   tickspace, % Space between tickmark and label
   tickdir, % Tickmark direction
   plot_xscaleloc (inst)(xpart (startpoint)),
                % Coordinate value at first endpoint
   plot_xscaleloc (inst)(xpart (endpoint)),
                % Coordinate value at second endpoint
   tickzero, % Coordinate value for a known
                %  tick location
   tickstep, % Coordinate space between ticks
   ticklabelformat
                % Format for tick labels
                %  (syntax from format.mp package)
                %  (use "" for no tick labels)
   )
enddef;
```

The *plot_ytickscale* definition is nearly identical. Note that these macros do not actually draw the tickmarks; they return a picture object, which can then be explicitly drawn or otherwise manipulated.

A simple way of adding grid labels to the previous example would be the following:

```
beginfig(6)
  % [...repeat of definitions of fig(4)...]
  x5 = plotB.pageleft;
  x6 = x4;
  y5 = y6 = plotB.pagebottom;
  draw plot_xtickscale(plotB)(z5, z6,
    0.08in, 0.06in, down, 0.0, 1.0, "%3f")
    withpen pencircle scaled 0.5pt;
  y7 = plotB.pagebottom;
  y8 = y2;
  x7 = x8 = plotB.pageleft;
  draw plot_ytickscale(plotB)(z7, z8,
    0.08in, 0.06in, left, 0.0, 0.5, "%3f")
    withpen pencircle scaled 0.5pt;
endfig;
```

The results of this are shown in Figure 6. As can be seen with the x-axis, the *tickscale* macros do not include the axis-lines themselves, thus allowing the user to draw them with a different line style than that used for the ticks, or to leave them off entirely.

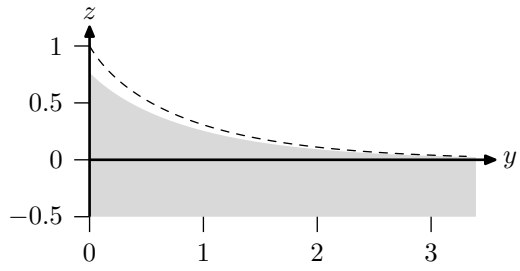


Figure 6. Fig. 5 repeated, with simple grid labels added.

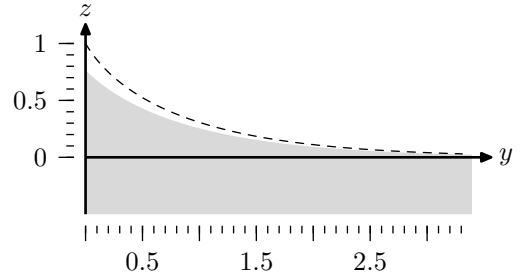


Figure 7. Fig. 5 again, with more advanced grid labels.

For a more polished look, we can move the grid ticks a small distance away from the plot, limit the y-axis range to the region that has meaningful significance, and add intermediate ticks without labels. In addition, this example illustrates the use of the *tickzero* parameter to start the labeled x-axis ticks at .5 rather than 0.

```
beginfig(7)
  % [...repeat of definitions of fig(4)...]
  x5 = plotB.pageleft;
  x6 = x4 − 0.1in;
  y5 = y6 = plotB.pagebottom − 0.06in;
  draw plot_xtickscale(plotB)(z5, z6,
    0.08in, 0.06in, down, 0.5, 1.0, "%3f")
    withpen pencircle scaled 0.5pt;
  draw plot_xtickscale(plotB)(z5, z6,
    0.08in, 0.06in, down, 0.0, 1.0, "")
    withpen pencircle scaled 0.5pt;
  draw plot_xtickscale(plotB)(z5, z6,
     0.04in, 0.06in, down, 0.0, 0.1, "")
    withpen pencircle scaled 0.5pt;
  y7 = y4;
  y8 = y2 − 0.1in;
  x7 = x8 = plotB.pageleft − 0.06in;
  draw plot_ytickscale(plotB)(z7, z8,
     0.08in, 0.06in, left, 0.0, 0.5, "%3f")
    withpen pencircle scaled 0.5pt;
  draw plot_ytickscale(plotB)(z7, z8,
    0.04in, 0.06in, left, 0.0, 0.1, "")
    withpen pencircle scaled 0.5pt;
endfig;
```

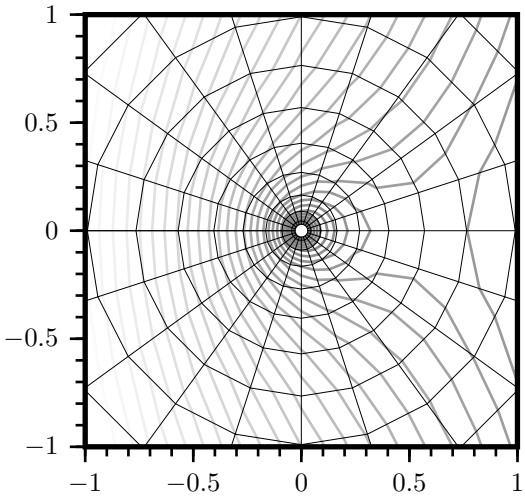


Figure 8. Sample graph created by MetaContour and MetaPlot, showing potential lines for a combination of a linear gradient and a point source, plotted on a polar grid.

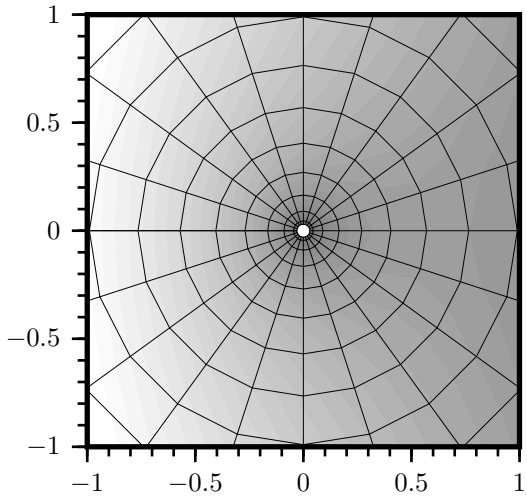


Figure 9. Another sample graph created by MetaContour and MetaPlot, illustrating a filled contour-plot style rather than using contour lines.

MetaContour: a C++ program for contour plots

Now that the MetaPost side of the collaboration has been described in some detail, we return to the matter of programs that generate plot objects as output. One of the particular reasons for developing MetaPlot was to have a way of producing contour plots, and so the MetaPlot package comes with a C++ program, MetaContour, for creating them.

The internals of MetaContour are beyond the scope of this paper, but it does make use of one additional capability of plot objects that is worth noting—the ability to include color information. The plot object is defined with commands like the following, with color directives.

```
picture contplotA.LinePlot;
contplotA.LinePlot := nullpicture;
addto contplotA.LinePlot doublepath
  (0.48075,0.50000)-- (0.48163,0.50597)
  withcolor contourcolor27;
addto contplotA.LinePlot doublepath
  (0.48420,0.50000)-- (0.48492,0.50490)
  withcolor contourcolor28;
addto contplotA.LinePlot doublepath
  (0.45994,0.50000)-- (0.46169,0.51245)
  withcolor contourcolor23;
    % [...and so forth...]
```

Then, before the plot object file is read into the main MetaPost file, the *contourcolor* array is defined as desired.

```
% Contour colors for grayscale scheme
color contourcolor[ ];
contourcolor0 = 1white;
contourcolor1 = 0.98white;
    % [...and so forth...]
contourcolor30 = 0.4white;
```

Thus, each line of the contour plot is associated with a color, and it will be drawn in that color unless it is overridden by another color directive; for instance, if we wanted to plot the contour lines all in black, we could do so simply by specifying:

```
draw continstA.plot(LinePlot) withcolor black;
```

Aside from the color contour-line plot just described, the MetaContour output contains a filled contour plot, and an image of the mesh of data points. Some examples of the these are shown in Figure 8 and Figure 9; although these are much more complex than the examples from preceding sections, the MetaPlot commands used to generate them are nearly identical.

Conclusion

The examples that have been shown illustrate only a small sampling of the capabilities of MetaPlot. In using MetaPost to generate the figures, it provides an easily extensible layout capability that is not limited by the imagination of the package author. The standardized plot-object interface simplifies the process of writing plot-generation programs, as they can leave the details of layout and annotation to the MetaPlot postprocessing.

At the time if this publication, MetaPlot and MetaContour should be available from CTAN in the `/graphics/metaplot` directory. They are still very much works in progress; I look forward to suggestions

and improvements, and hope that others will find them to be useful tools.

Notes

1. For those who are curious, the equation (from [4]) is

$$\frac{y}{d} = \cosh^{-1}\frac{2d}{z} - \cosh^{-1}\frac{2d}{h} + \left(4 - \frac{h^2}{d^2}\right)^{\frac{1}{2}} - \left(4 - \frac{z^2}{d^2}\right)^{\frac{1}{2}},$$

where $h^2 = 2d^2(1 - \sin\theta)$ is the height of the meniscus, θ is the contact angle, and d is a scaling parameter related to the surface tension and liquid density.
2. Although these variables are represented here as numerics and thus are still vulnerable to under- or overflow, it would be a simple matter to replace them with string-represented numbers from the `sarith` package.
3. There is, of course, no need for flexible implementations and simple interfaces to be mutually exclusive, and functions for more automated axes may be included in MetaPlot as it continues to be developed.

References

[1] Hwang, A., ePiX, `http://mathcs.holycross.edu/~ahwang/current/ePiX.html`.
[2] Gnuplot, `http://www.gnuplot.info`.
[3] Phan, A., m3D, `http://www-math.univ-poitiers.fr/~phan/m3Dplain.html`.
[4] Batchelor, G. K., *An Introduction to Fluid Dynamics*, Cambridge University Press, 1967.

Brooks Moses
Mechanical Engineering,
Stanford University,
Building 520,
Stanford, CA 94305
U.S.A.
bmoses@stanford.edu

Support for typesetting greek in ConTEXt

(cb-greek fonts)

Keywords
Maps, ConTeXt, module, greek, cb-greek, font

Abstract
There are situations where one needs to typeset pieces of text in greek. Until recently there was no direct support to do this in ConTEXt. With the integration of the module greek this has changed. The basics where built by Giuseppe Bilotta (Italy). The module uses a subset of the cb-greek fonts. The article describes the module and the way greek text is coded. A couple of short examples as well as a longer example of greek text are given.

Introduction

Some time ago there were questions concerning the support of the greek language in ConTEXt. It appeared that Giuseppe Bilotta from Italy had already done some work on this. He prepared typescripts and a sample file which shows the use of the cb-greek fonts. After some cleanup, these typescripts were added to the alpha distribution of ConTEXt and tested by users.

Description of the subset of the cb-greek font package

The cb-greek font package is a very large collection of font files available as MetaFont – and also as type1 versions. In the archive of the type1 files (some 70 MB) there is a huge number of typefaces and fontsizes included that one may never need. In cooperation with the TEXlive team a medium subset hereof (some 14MB) was prepared. This version is also included on the TEXlive media 2004 including the corresponding map file cbgreek.map.

The chosen subset of the cb-greek fonts comprises a rather complete series of fonts including serif, sans serif, roman, bold, italic, slanted, small capitals and monospaced variants. The included design-sizes are 8pt, 10pt and 12pt (see the annex for a sample text of each font).

Installation

ConTEXt comes with the module greek included in the distribution. In order to make the cb-greek fonts available one has to install the cb-greek fonts from the TEXlive media. The font-subset can be found on the DVD under /texlive2004/texmf-dist/fonts/tfm,pfb/public/cb. The necessary map file resides under /texlive2004 /texmf-dist/fonts/map/dvips/cbgreek.map. Place the tmf- and pfb-files into your TDS at the appropriate place. Copy the map file to a place where TEX can find it.

After running mktexlsr one should be ready for the first tests.

Use of the greek module

Invoking the greek module is done at the beginning of the document by saying: `\usemodule[greek]`

After this two commands are available by which the greek texts are to be surrounded:

`\startgreek ...\stopgreek`

Alternatively you can also typeset greek inline with `\greek{...}`

The module does two things:

— It does a fontswitch with `\switchtobodyfont[cbgreek]`
— Switches to the local language `\language{greek}`

When starting a piece of greek text the typeface that will be used is according to the bodyfont in the main text e.g. it will be rm at 10pt or ss at 12pt or whatever has been set for the maintext's bodyfont.

Possibly one would like to have the greek fragment typeset in sans serif or monospaced and with another size. This can easily be achieved with

```
\startgreek
\tt\tfd
p'anta <re~i.
\stopgreek
```

πάντα ῥεῖ.

Besides this, one can add commands like `\bf`, `\it`, `\em` and `\sc`.

The greek alphabet

For illustration we type first the latin alphabet and show then how the different positions come out in the greek alphabet:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
α	β	ς	δ	ε	φ	γ	η	ι	ϑ	ϰ	λ	μ	ν	ο	π	χ	ρ	ς	τ	υ		ω	ξ	ψ	ζ

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Α	Β	ℭ	Δ	Ε	Φ	Γ	Η	Ι	Θ	Κ	Λ	Μ	Ν	Ο	Π	Χ	Ρ	Σ	Τ	ϒ	℧	Ω	Ξ	Ψ	Ζ

There are some peculiar outcomes specially in the uppercase characters. Furthermore you see that j gives ϑ and c gives the end-of-the-word sigma ς. It is noteworthy that s gives either the normal or the end-of-the-word sigma, depending on the position of occurrence:

`\greek{basile'ios}` βασιλείος

All diacritical signs are available to typeset polytonic/classical greek. Accents, breathings and subscribed iotas can be created as displayed in the following table:

Greek	Coding	Name
ά	`\greek{'a}`	acute
ὰ	`\greek{`a}`	grave
ᾶ	`\greek{~a}`	circumflex
ῥ	`\greek{<r}`	rough breathing
ὠ	`\greek{>w}`	smooth breathing
α᾽	`\greek{a"}`	apostrophe
ᾳ	`\greek{a|}`	subscript iota
ῃ	`\greek{h|}`	
ῳ	`\greek{w|}`	

Note that you will have to type a double-quote after the last character to obtain an apostrophe:

```
\greek{pant"}  παντ’
```

When using uppercase characters, diacritical signs get the correct position automatically:

```
\greek{‘A, ’A, ~A, <A, >A, A|}  Ἀ, Ἁ, Α͂, Ἄ, Ἂ, Ἇ
```

When using small capitals no accents or breathings at all are typeset.

```
\greek{\sc <oi >Ajhna’ioi}  ΟΙ ΑΘΗΝΑΙΟΙ
```

Examples

Now let us see what we can achieve with this module!

We start with presenting the input coding of the first 10 verses of the first book of the Odyssee by Homeros. Following that, the text is typeset in greek. – Because I believe that there are not too many among us who can read greek, different translations of the same verses are shown next . . .

As a second example, the text is shown which was presented in the article on the book revival of Johan Polak’s ”Bloei der decadence” (Maps#31).

Greek text Odyssee, 1, 1–10

From: http://www.vox-graeca-gottingensis.de/Texte/Homer/Odyssee/od1.1-10.htm

```
\startgreek
\startitemize[n,packed]
\bf
\item >’Andra moi >’ennepe, Mo~usa, pol’utropon, <’os m’ala p’olla
\item pl’agqjh, >epe‘i Tro’ihs <ier‘on ptol’iejron >’eperse;
\item poll~wn d" >anjr’wpwn >’iden >’astea ka‘i n’oon >’egnw,
\item poll‘a d" <’o g" >en p’ontw| p’ajen >’algea <‘on kat‘a jum’on,
\item >arn’umenos <’hn te yuq‘hn ka‘i n’oston <eta’irwn.
\item >all" o>ud" <‘ws <et’arous >err’usato <i’emen’os per
\item aut~wn g‘ar sfet’erh|sin >atasjal’ih|sin >’olonto,
\item n’hpioi, <o‘i kat‘a bo~us <Uper’ionos >Hel’ioio
\item >’hsjion a>ut‘ar <o to~isin >afe’ileto n’ostimon <~hmar.
\item t~wn >am’ojen ge, je’a, j’ugater Di’os, e>ip‘e ka‘i <hm~in.
\stopitemize
\stopgreek
```

1. Ἄνδρα μοι ἔννεπε, Μοῦσα, πολύτροπον, ὅς μάλα πόλλα
2. πλάγχθη, ἐπεὶ Τροίης ἱερὸν πτολίεθρον ἔπερσε·
3. πολλῶν δ’ ἀνθρώπων ἴδεν ἄστεα καὶ νόον ἔγνω,
4. πολλὰ δ’ ὅ γ’ ἐν πόντῳ πάθεν ἄλγεα ὃν κατὰ θυμόν,
5. ἀρνύμενος ἥν τε ψυχὴν καὶ νόστον ἑταίρων.
6. ἀλλ’ οὐδ’ ὣς ἑτάρους ἐρρύσατο ἱέμενός περ
7. αὐτῶν γὰρ σφετέρῃσιν ἀτασθαλίῃσιν ὄλοντο,
8. νήπιοι, οἱ κατὰ βοῦς Ὑπερίονος Ἠελίοιο
9. ἤσθιον αὐτὰρ ὁ τοῖσιν ἀφείλετο νόστιμον ἦμαρ.
10. τῶν ἀμόθεν γε, θεά, θύγατερ Διός, εἰπὲ καὶ ἡμῖν.

English translation by Samuel Butler

From: http://www.perseus.tufts.edu/cgi-bin/ptext?
doc=Perseus:text:1999.01.0218:book=1:line=1

1. Tell me, O Muse, of that many-sided hero who traveled far and wide
2. after he had sacked the famous town of Troy.
3. Many cities did he visit, and many were the people
4. with whose customs and thinking he was acquainted;
5. many things he suffered at sea while seeking to save his own life and to achieve the safe homecoming of his companions;
6. but do what he might he could not save his men, for they perished through their own
7. sheer recklessness in eating the cattle of the Sun-god Helios;
8. so the god prevented them from ever reaching home.
9. Tell me, as you have told those who came before me,
10. about all these things, O daughter of Zeus, starting from whatsoever point you choose.

Dutch translation by Van 's Gravenweert

From: Jan van 's Gravenweert. De Odysséa van Homerus, naar het Grieksch, in Nederduitsche verzen gevolgd. Johannes van der Hey en Zoon, Amsterdam. 1823.
http://www.koxkollum.nl/Homerus/odyssee01sGravenweert.htm

1. Spreek, Zangster! van den held, in krijg en list volleerd,
2. Die, toen hij Trojes wal tot puin had omgekeerd,
3. Moest zwerven; die op zee rampzalig heeft geleden,
4. Een aantal volkren zag en onbekende zeden,
5. Wanneer hij voor zich zelv' en 't leger redding zocht;
6. Maar, hoe trouwhartig ook, geen benden helpen mogt,
7. Die woest en onbedacht zich zelve in 't onheil bragten,
8. En 't rundervee der Zon ten gruwbren maaltijd slagtten, -
9. Der Zon, wier gramschap haar geen' dag van keeren gaf.
10. Spreek, Zangster! van hunn' togt, hun lijden en hun straf.

German translation by Johann Voß, 1781

From: Project Gutenberg Germany: http://gutenberg.spiegel.de/homer/odyssee/odyss011.htm

1. Sage mir, Muse, die Taten des vielgewanderten Mannes,
2. Welcher so weit geirrt nach der heiligen Troja Zerstörung,
3. Vieler Menschen Städte gesehn und Sitte gelernt hat
4. Und auf dem Meere so viel unnennbare Leiden erduldet,
5. Seine Seele zu retten und seiner Freunde Zukunft.
6. Aber die Freunde rettet' er nicht, wie eifrig er strebte;
7. Denn sie bereiteten selbst durch Missetat ihr Verderben:
8. Tore! Welche die Rinder des hohen Sonnenbeherrschers
9. Schlachteten; siehe, der Gott nahm ihnen den Tag der Zukunft,
10. Sage hievon auch uns ein weniges, Tochter des Kronions.

Swiss german translation

Here follows for curiosity a special translation in swiss dialect. It is written in the spoken language of the county of Bern.

From: A. Meyer. Homer Bärndütsch, Odyssee. Edition Francke im Cosmos Verlag, Bern. 4. Auflage 1988. ISBN 3305000694. (With permission from the publishing house.)

1. Göttlechi Tochter, o sing mer das Lied doch vom gwaglete Ma itz!
2. Isch es gwüss ihn ja, wo Troja, di herrlichi Veschti het proche.
3. Het drufaba dert mängerlei Lüten Örter glehrt bchenne.
4. Allerlei Bittersch u Leids ufem Meer uss erläbt u erlitte.
5. Toll um sys Läben ou grunge, für d'Heifahrt vo all syne Gspane.
6. Aber doch alles für nüt und ekeine het er errettet.
7. Zgrund si si ggange dür eigeti Schuld u verwägene Frävel.
8. Töde di Tröpf nid am Sunnegott syner heilige Stiere!
9. Gwaltig ertöibt wäge däm laht dä Gott sälb Fahrt la vergrate.
10. Öppis vo däm doch sing is u pricht is, du herrlichi Göttin!

The next example shows the footnote in Johan Polak's "Bloei der decadence" (Maps#31), now produced with the greek module:

```
\unknown\greek{peri g‘ar qaq‘a p’antojen >’esth.}
```

...περι γὰρ χαχὰ πάντοθεν ἔστη.

This is the second part of the verse 270 in the 14th book of the Odyssee. It can be translated as:

English : ...for evil surrounded us on every side
German : ...denn ringsum drohte Verderben
Dutch : ...want het gevaar kwam van alle kanten

Conclusion

ConTEXt provides a module which can be used to typeset pieces of greek text in documents based on other languages than greek. The module is based on a subset of the cb-greek fonts, allowing to typeset greek fragments in serif, sans serif, roman, bold italic and slanted, typewriter in roman, italic and slanted including small capitals for serif, sans and typewriter. The cb-greek fonts are already available from the CTAN archives. A medium subset is included in TEXlive media 2004 together with the suitable map-file. Due to the fact that a broad range of typefaces is available in the subset, there should always be a way to make such a choice that the combination of the maintext's bodyfont with the cb-greek font gives an appealing result.

Post script

When the writer was at school he learned the following saying:

Τῶν πονῶν πολούσιν πάντα θ’ ἀγαθὰ ὁὶ Θεοί

We will be glad to hear your personal translation. The best and most hilarious solutions will be presented in the next edition of the Maps! You are invited to send your solution to: ntg-secretaris@ntg.nl.

Willi Egger
Hans Hagen
w.egger@boede.nl, h.hagen@pragma.com

Annex: cb-greek fonts subset in TEXlive 2004

Serif, roman, bold-roman

Τῶν πονῶν πολούσιν πάντα ϑ' ἀγαθὰ ὀὶ Θεοί
Τῶν πονῶν πολούσιν πάντα ϑ' ἀγαθὰ ὀὶ Θεοί

Serif, italic, slanted

Τῶν πονῶν πολούσιν πάντα ϑ' ἀγαθὰ ὀὶ Θϵοί
Τῶν πονῶν πολούσιν πάντα ϑ' ἀγαθὰ ὀὶ Θεοί

Sans, roman

Τῶν πονῶν πολούσιν πάντα θ' ἀγαθὰ ὀὶ Θεοί

Sans, italic, slanted

Τῶν πονῶν πολούσιν πάντα θ' ἀγαθὰ ὀὶ Θεοί
Τῶν πονῶν πολούσιν πάντα θ' ἀγαθὰ ὀὶ Θεοί

Small Caps, serif, sans, mono

ΤΩΝ ΠΟΝΩΝ ΠΟΛΟΥΣΙΝ ΠΑΝΤΑ Θ' ΑΓΑΘΑ ΟΙ ΘΕΟΙ
ΤΩΝ ΠΟΝΩΝ ΠΟΛΟΥΣΙΝ ΠΑΝΤΑ Θ' ΑΓΑΘΑ ΟΙ ΘΕΟΙ
ΤΩΝ ΠΟΝΩΝ ΠΟΛΟΥΣΙΝ ΠΑΝΤΑ Θ' ΑΓΑΘΑ ΟΙ ΘΕΟΙ

Typewriter, roman, italic, slanted

Τῶν πονῶν πολούσιν πάντα θ' ἀγαθὰ ὀὶ Θεοί
Τῶν πονῶν πολούσιν πάντα θ' ἀγαθὰ ὀὶ Θϵοί
Τῶν πονῶν πολούσιν πάντα θ' ἀγαθὰ ὀὶ Θεοί

A Simple Book Design in ConTEXt

Abstract
I show how a simple book design can be implemented in ConTEXt

Motivation

Whenever I learn a new TEX system, I try to implement a design for a significant number of pages. Many years ago, when I was learning LaTEX (2.09), I wrote a `rawls.sty` to mimic Harvard Press's design for John Rawls's *A Theory of Justice* [3], a design that featured a number of interesting features: not only were there parts, chapters, and sections, but the sections were numbered consecutively throughout the book, orthogonally to the chapters. So the first section of Chapter 3, say, might be numbered section 16. Page headers had rules under the header texts. I mention all this because in those pre-LaTEX2$_\varepsilon$ days, it was far from trivial to make substantive changes to the default styles. I remember studying Don Knuth's (plain) code for his Computer Journal article on Literate Programming and thinking what a nightmare it would be to implement in LaTEX; but that was 1988. Matters have certainly improved since then.

When I first encountered ConTEXt I was immediately impressed by the `setups` mechanism of key/value pairs approach to a design interface. I began using ConTEXt for typesetting internal documentation here at Duke Press (coded in DocBook XML and processed using Simon Pepping's *DocbookInConTeXt* [2]). But I had in mind all along trying out ConTEXt in a larger project. I wanted to see how easy it would be to render a book design compared to LaTEX. I suspected it would be much easier; I was right.

The Text

When I discovered that a very strange book I first read as a youth, *A Voyage to Arcturus* by David Lindsay [1], had been deposited in Project Gutenberg, I knew I had my text. In the event, the OCR text was quite corrupt, and it took a while to make the necessary edits to bring it to an acceptable standard.

The design I had in mind for the book was based on a mathematics text I read in college. The unifying theme was a vertical rule separating visual elements of the chapter headings and page headers.

Fonts

I decided to use a Bembo clone (called Bergamo) for the text and an Optima clone (called Opus) for the chapter headings and header texts. Both are from the FontSite 500 collection. To use these fonts with ConTEXt, I write some typescripts.

```
\starttypescript [serif] [bergamo] [ec]
 \definefontsynonym [Bergamo-Roman] [5borjx8t] [encoding=ec]
 \definefontsynonym [Bergamo-Bold]  [5bobjx8t] [encoding=ec]
```

```
 \definefontsynonym [Bergamo-Italic][5borix8t][encoding=ec]
 \definefontsynonym [Bergamo-Bold-Italic][5bobix8t][encoding=ec]
 \definefontsynonym [Bergamo-Caps]  [5borcj8t] [encoding=ec]
 \definefontsynonym [Bergamo-Bold-Caps] [5bobcj8t] [encoding=ec]
\stoptypescript
```

Observant readers who know the Berry naming conventions will see that Bergamo contains both full 'f' ligatures and old-style numerals.

Note here that I declare that maths be in scaled Palatino (even though in this project there are no maths). I find that Palatino for maths blends well with Bergamo, and I wanted to go ahead and set this up for future projects.

```
\starttypescript [Bergamo]
 \definetypeface [Bergamo] [rm] [serif] [bergamo] [default]
 [encoding=ec]
 \definetypeface [Bergamo] [ss] [sans]  [opus]    [default]
 [encoding=ec]
 \definetypeface [Bergamo] [mm] [math] [palatino] [default]
 [rscale=.90,encoding=ec]
 \definetypeface [Bergamo] [tt] [mono] [modern] [default]
\stoptypescript
```

The code for Opus is similar. I store these typescripts in `type-fontsite.tex` and invoke them. Note that I use hanging punctuation and open up the lines to improve readability.

```
% Set up hanging punctuation, pure style;
% Declare Berry naming conventions, ec encoding
\usetypescript[serif] [hanging] [pure]
\usetypescript[berry][ec]

% Load Bergamo and Opus fonts, declare sizes and leading.
% Looks better if I open up the lines a bit.
\usetypescriptfile[type-fontsite]
\usetypescript[Bergamo]
\setupbodyfont[Bergamo,10pt]
\setupinterlinespace[line=1.35em]

\setupalign[hanging]
```

Chapter Heads, Page Headers and Footers

I set up the heads with these options

```
\setuphead[chapter]
          [page=yes,
           before={\null\blank[4*line]},
           after={\blank[4*line]},
           command=\mychap]
```

Note the `command` option. This allows me to design my own chapter head appearance. `\mychap` looks like this: `#1` refers to the chapter number, and `#2` refers to the chapter title

```
\def\mychap#1#2%
{\hbox to \hsize\bgroup
\hfill % the % after {#1} suppresses a space
\setupframed[offset=0.5em,frame=off]
\tbox{\framed[width=2cm,align=left]{\ss #1}}%
```

```
% now instructions for #2
\tbox{\framed[width=.5\textwidth,align=right,leftframe=on]
{\raggedright
\hyphenpenalty 10000 \ss #2}} \egroup}
%anything but rag right with
%no hyphenation looks bad
```

I want dropped caps for my chapter openers, and small caps afterwards for a certain number of words that I choose. The dropped cap will be in Opus, be 3\baselineskips tall, be dropped one line, and have 2 points of padding.

```
\def\Drop{\DroppedCaps
  {} {Sans} {3\baselineskip} {2pt} {1\baselineskip} {2}}
\def\chap#1/#2/{\Drop #1{\sc#2}}
```

so I can say

```
\chapter{The S\'eance}

\chap O/n a march evening/, at eight o'clock, Backhouse, the
```

You can see the result in Figure 1.

To unify the design, I make the headlines mirror the chapter openers, with a vertical rule separating verso the page number and book title and recto the chapter title and page number, all in Opus. First I declare doublesided pages and turn off auto page-number placement. Then I specify a different scheme for chapter opening pages.

```
% Remove auto page numbering placement; I'll do it manually
\setuppagenumbering[alternative=doublesided,location=]
% Set up header texts, recto and verso
\setupheadertexts[][{\getmarking[chapter][current]}
                     \quad\vrule\quad\pagenumber]
        [\pagenumber\quad\vrule\quad A Voyage to Arcturus][]
\definetext[chapterstart][footer][pagenumber]
% Define heads for chapter opening pages
 \setuphead[chapter][header=empty,footer=chapterstart]
\setupheader[style=\ss]
```

The result can be seen in Figure 2.

Now I specify the Table of Contents:

```
% Set up table of contents format. Move whole operation to the right
% to better center the TOC, and make sure chap numbers
% align properly (flushright) in their own box
\definelist[chapter]
\setuplist[chapter]
          [alternative=a,
          margin=.2\textwidth,
          numbercommand=\NumCom]
\def\NumCom#1{\hbox to 2em{\hfill #1}}
```

Setting Up the Pages

Last (actually first) I set up the pages and a switch for page imposition. Pay attention to the commented lines for crop marks, etc.

```
% text size
\definepapersize[arc][height=220mm,width=145mm]
\setuppapersize[arc][letter]
```

```
% Set up arrangements for printing as booklet. Toggle as needed.
%\setuparranging[2UP,rotated,doublesided]

\setuplayout[margin=0pt,width=middle]
\setuplayout[topspace=2\baselineskip,height=middle]
%\showsetups
% Layout modifications to headers, etc
\setuplayout[header=2\baselineskip,footer=2\baselineskip]
\setuplayout[location=middle]
% \setuplayout[marking=on] % crop marks
\setupindenting[medium]
\frenchspacing % I guess Bush would call this 'freedomspacing'
```

Finally, for output targeted for a computer screen instead of print, I can say

```
\setuppapersize[S6][S6]
\setupinteraction[state=start]
```

I can't argue strenuously enough for this approach to books and articles destined for a computer screen. The advantages to making one's way through the text by just the touch of the space bar are, to me, self–evident.

Future Work

Clearly, implementing a simple design in ConTEXt is quite straightforward. In fact, the advantages of using ConTEXt become more obvious the more complicated the document design. I hope that this article might motivate others to give ConTEXt a try for their own typesetting projects.

Eventually I plan to code the book in XML along with supporting files for browser display and direct typesetting with ConTEXt. For the moment, I will post the screen version at `http://www.duke.edu/~grath/arcS6.pdf`, after a friend designs a suitable cover page for it. Other versions will follow when ready.

But be warned—many have found Lindsay's philosophy detestable (a worship of suffering is one characteristic of it). The English writer C.S. Lewis certainly found it so, even if the book did influence his wonderful space novels.

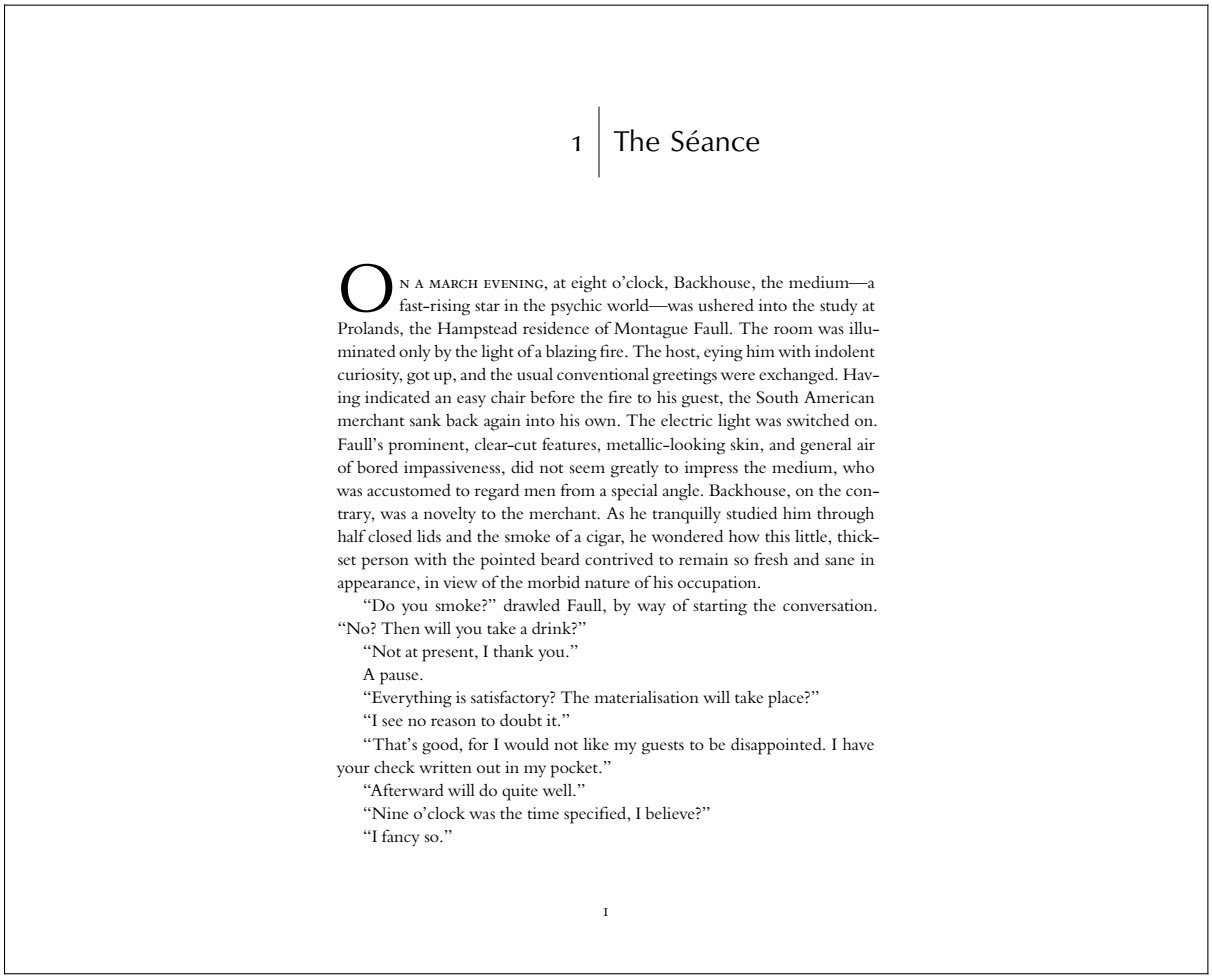
References

[1] Lindsay, David, *A Voyage to Arcturus*, London, Methuen, 1920. Text available at Project Gutenberg: `http://www.gutenberg.net/etext/1329`. Other, corrupt editions can be found on `amazon.com`.

[2] Pepping, Simon, *DocbookInConTeXt*, available at `http://www.leverkruid.nl/context/index.html`.

[3] Rawls, John, *A Theory of Justice*, Revised edition, Belknap Press, Cambridge, MA, 1999.

Steve Grathwohl
Duke University Press
Durham, NC USA



1 | The Séance

On a march evening, at eight o'clock, Backhouse, the medium—a fast-rising star in the psychic world—was ushered into the study at Prolands, the Hampstead residence of Montague Faull. The room was illuminated only by the light of a blazing fire. The host, eying him with indolent curiosity, got up, and the usual conventional greetings were exchanged. Having indicated an easy chair before the fire to his guest, the South American merchant sank back again into his own. The electric light was switched on. Faull's prominent, clear-cut features, metallic-looking skin, and general air of bored impassiveness, did not seem greatly to impress the medium, who was accustomed to regard men from a special angle. Backhouse, on the contrary, was a novelty to the merchant. As he tranquilly studied him through half closed lids and the smoke of a cigar, he wondered how this little, thick-set person with the pointed beard contrived to remain so fresh and sane in appearance, in view of the morbid nature of his occupation.

"Do you smoke?" drawled Faull, by way of starting the conversation. "No? Then will you take a drink?"

"Not at present, I thank you."

A pause.

"Everything is satisfactory? The materialisation will take place?"

"I see no reason to doubt it."

"That's good, for I would not like my guests to be disappointed. I have your check written out in my pocket."

"Afterward will do quite well."

"Nine o'clock was the time specified, I believe?"

"I fancy so."

1

Figure 1 A chapter opening page

The conversation continued to flag. Faull sprawled in his chair, and remained apathetic.

"Would you care to hear what arrangements I have made?"

"I am unaware that any are necessary, beyond chairs for your guests."

"I mean the decoration of the séance room, the music, and so forth."

Backhouse stared at his host. "But this is not a theatrical performance."

"That's correct. Perhaps I ought to explain. There will be ladies present, and ladies, you know, are aesthetically inclined."

"In that case I have no objection. I only hope they will enjoy the performance to the end."

He spoke rather dryly.

"Well, that's all right, then," said Faull. Flicking his cigar into the fire, he got up and helped himself to whisky.

"Will you come and see the room?"

"Thank you, no. I prefer to have nothing to do with it till the time arrives."

"Then let's go to see my sister, Mrs. Jameson, who is in the drawing room. She sometimes does me the kindness to act as my hostess, as I am unmarried."

"I will be delighted," said Backhouse coldly.

They found the lady alone, sitting by the open pianoforte in a pensive attitude. She had been playing Scriabin and was overcome. The medium took in her small, tight, patrician features and porcelain-like hands, and wondered how Faull came by such a sister. She received him bravely, with just a shade of quiet emotion. He was used to such receptions at the hands of the sex, and knew well how to respond to them.

"What amazes me," she half whispered, after ten minutes of graceful, hollow conversation, "is, if you must know it, not so much the manifestation itself—though that will surely be wonderful—as your assurance that it will take place. Tell me the grounds of your confidence."

"I dream with open eyes," he answered, looking around at the door, "and others see my dreams. That is all."

"But that's beautiful," responded Mrs. Jameson. She smiled rather absently, for the first guest had just entered.

It was Kent-Smith, the ex-magistrate, celebrated for his shrewd judicial humour, which, however, he had the good sense not to attempt to carry into

Figure 2 A page with headers

OpenType in ConTEXt

Multi–dimensional font support and advanced font features

Keywords
ConTeXt fonts OpenType

Abstract
This is a summary of issues encountered and solutions implemented in order to support some advanced OpenType features in ConTEXt. This article describes an accompanying set of support files that address installation (using TEXFONT), accommodating extended opticals families, and some "pro" font features. The extended character set afforded by pro fonts enables support for comprehensive small caps and old–style figures. Although the typescripts and commands are described together, certain features (like variant encodings for TEXFONTand optical typescripts) can be used independently of the other features described.

Introduction

This article, originally produced as a ConTEXt "My Way" article, introduces some issues in installing, using, and integrating OpenType fonts in ConTEXt. OpenType has been discussed elsewhere in detail, but it should suffice to say that it is a modern melding of POSTSCRIPT and TrueType, enabling advanced typography features and Unicode support.

RomanCaps and
ItalicCaps and
BoldCaps and
BoldItalicCaps
Figure 1 Small caps in several variants

With some of the most complex OpenType fonts, one also sees integration with multiple design sizes. This is not necessarily unique to OpenType fonts, but is a co–occurring feature seen with "premium"–quality fonts. Some of the advanced typographical features seen in such fonts are integrated glyphs for small caps (across every font variant), old style and tabulation figures, and greek glyphs, all within the same font. This article introduces some strategies for taking advantage of small caps and old style figures, and for rudimentary math fonts for fonts that include greek language support.

It should be noted that this package owes its existence to Adobe's "Type Classics for Learning,"[1] an education–only package of very high quality OpenType fonts, for 99 USD. Many thanks to Bruce D'Arcus for pointing out the existence of this package, and his enthusiasm and cunning for encouraging me to do something with it. He also pointed me to the work of Achim Blumensath, whose efforts in the LATEX domain sparked my initial ideas about font installation. Otared Kavian provided an extended mathematics example used in earlier drafts. George Williams and Eddie Kohler provided the actual font wizardry. Obvious thanks to Hans Hagen and Don Knuth for providing such a rich foundation for this modest addition to the TEX and ConTEXt canon.

The working assumption throughout this article is that users use ConTEXt with PDFTEX, which can handle the converted forms of each of these fonts. The support files (typescripts and sample encodings) are available at the author's web site.[2]

Font installation using FontForge

The first issue to deal with is getting the fonts to be installed into the TeX tree. Early on, I decided that I should try to use TEXFONT, because it was clearly a ConTEXt–friendly tool, it had excellent globbing support (e.g., for converting and installing an entire directory of fonts at once), and, frankly, because I was slightly more familiar with it than other tools. It was pointed out that FontForge[3], by George Williams, was a powerful, freely–available, open–source, cross–platform tool that handled OpenType fonts as well as any other. This first font installation method therefore depends on a working installation of FontForge.
I extended TEXFONT to include an optional pre-processing step that converts an OpenType font to a `.pfb` and `.afm` pair. From there, TEXFONT could work its usual magic.

The first option of interest to someone installing OpenType fonts is `-preproc`. This command–line switch triggers a run of FontForge for each `otf` found in the current directory.

For example, if you wanted to install the Cronos Pro OpenType fonts, go into the directory containing the fonts, and issue the command:

```
texfont --makepath --install --enco=texnansi --preproc --vend=adobe --coll=CronosPro
```

If you prefer to work with batch files in TEXFONT, the following line should be a good place to start:

```
--en=? --ve=adobe --co=CronosPro --so=auto --pre
```

In order to support at least some of the many extended characters in a “Pro” font, another command–line option was added to TEXFONT, `-variant=blah` (abbreviated as `-va=`), which allows one `.enc` file to masquerade as another. For example, if I am working with the `texnansi` encoding, I can create a *variant* encoding that substitutes small caps for the lowercase letters. I name that encoding `texnansiSC.enc`, put it in a place where it can be found (like fonts/enc/dvips/context under TDS 1.1), and run:

```
texfont --en=texnansi --va=SC --pre --ve=adobe --co=CronosPro
```

The `-variant` option appends the variant's name (`SC`, here) to the end of the name of the encoding (`texnansi`), and looks for that particular `.enc` file on the path. One variant, `texnansiOSFSC.enc`, is included in the support files as a starter. It was the only variant I personally needed for initial support of my own fonts, but you're welcome to create (and share!) your own.

It is worth noting that there is nothing about the `-variant` option that is intrinsic to OpenType fonts. If you are working with any sort of font with extended glyphs (such as Swash Caps), you can create a font that accesses those extended glyphs (while masquerading as a known encoding) by using this method.

Font installation using LCDF Typetools

For those who don't have the patience to hand–assemble custom encoding variants, a more palatable option will be to use the LCDF Typetools[4] from Eddie Kohler. These tools have advanced, specialised features with respect to OpenType. The `otftotfm` tool is very capable of installing fonts by itself, but integration with TEXFONT gives further automation (globbing again) and better integration with ConTEXt.
In order to use the LCDF Typetools within TEXFONT, you should add the command–line option `-lcdf`. This option alone will install the raw `.otf` files, which PDFTEX, as of this writing, can use and embed into PDF files, but cannot subset. A more likely option for most users will be to combine `-lcdf` with `-preproc`, which converts the `.otf` files into normal Type1 `.pfb` files, a more commonly usable format for modern TEX systems.

Eddie's tools' real strength lies in the ability to activate OpenType features and use all the glyphs in the font to their full potential. These features include non–standard ligatures, contextual swashes, small caps, number cases and spacing, and historical alternates. The tools accomplish this by creating a custom encoding and by embedding ligature information in the `.tfm` files it creates.

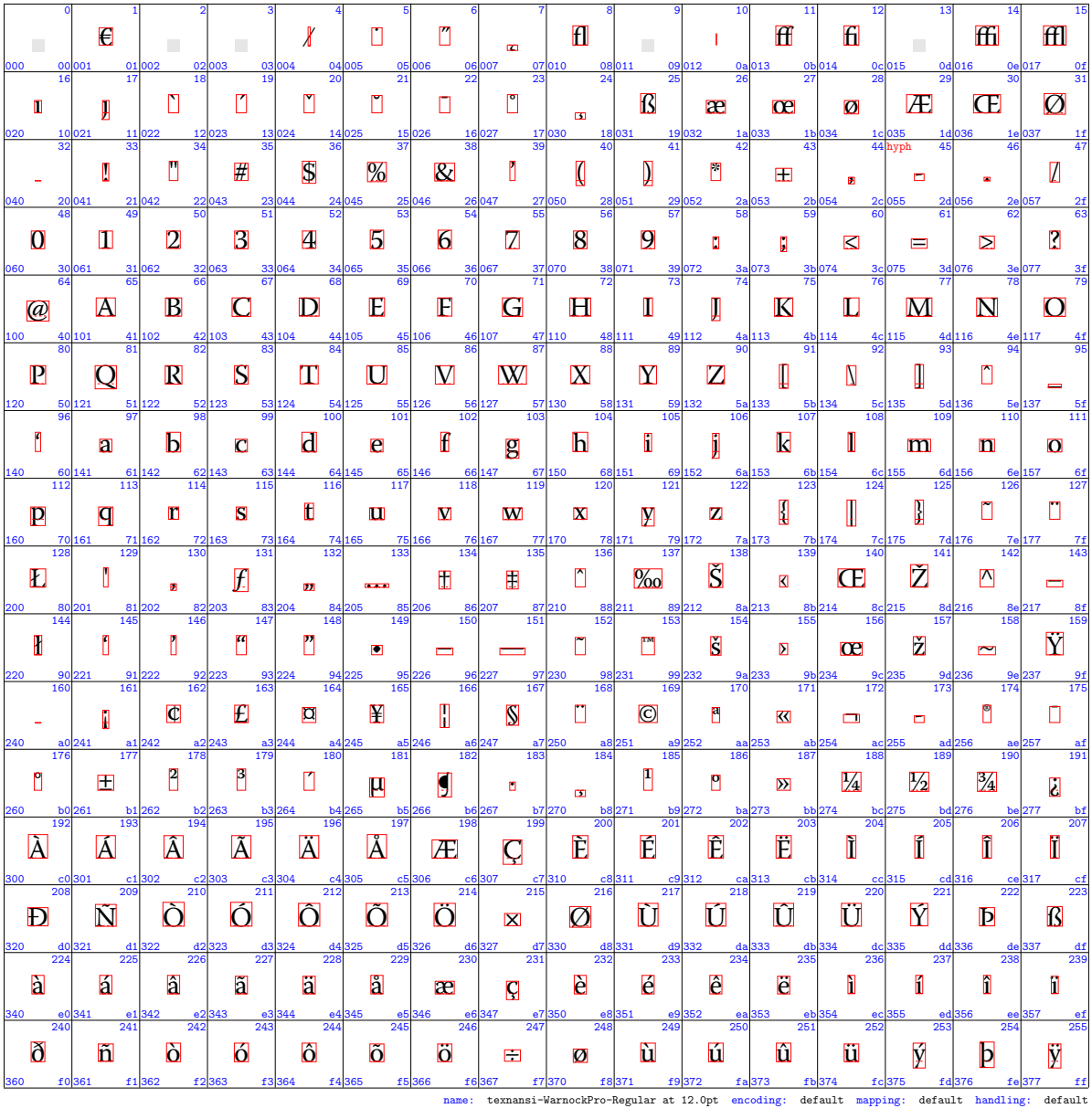


Figure 2 WarnockPro-Regular in the normal tex'n'ansi encoding

A TEXFONT user needs to be aware of two main things when using TEXFONT with LCDF Typetools. The `-variant` command–line option is re-interpreted as a means to pass four–letter OpenType features. For example, the command–line option of `-va=liga,kern,onum,pnum` passes the options of *ligatures, kerning, old–style numerals,* and *proportionally–spaced numerals* to the LCDF Typetools. The resultant `.tfm` files will be named: baseencoding-LIGA-KERN-ONUM-PNUM-fontname.tfm,

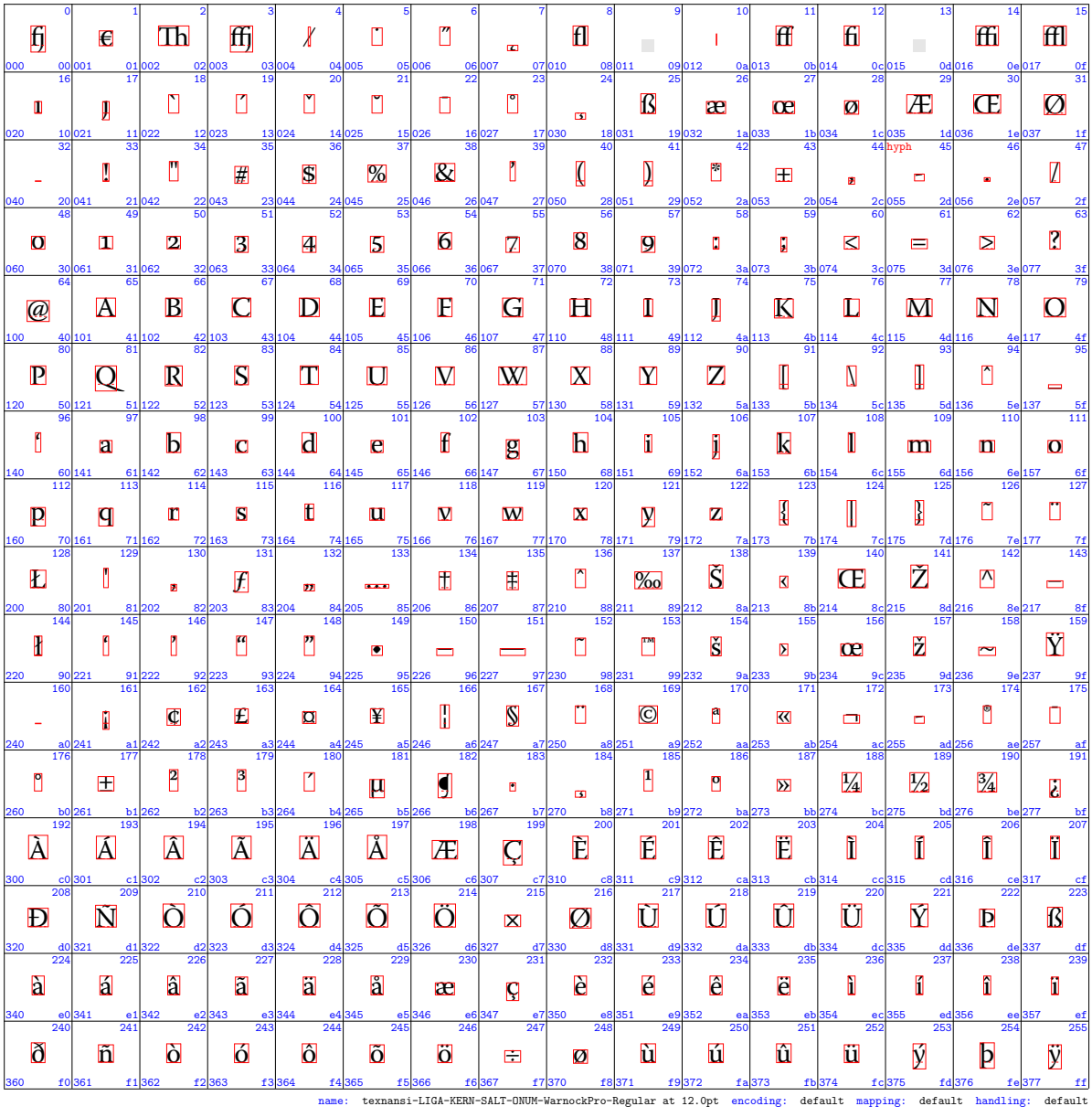


Figure 3 WarnockPro-Regular in the tex'n'ansi encoding, modified with the LIGAture, KERNing, Stylistic ALTernates, and Oldstyle NUMerals features. Note that not only are ligatures inserted into blank slots in the encoding, but the numerals and some letters (like the 'k', 'v', and 'y') are substituted with alternate forms.

and the produced `.map` file will be named: baseencoding-LIGA-KERN-ONUM-PNUM-vendor-collection.map.

The second thing to be aware of is TEX's limit of 256 glyphs in any font encoding. Many features (including ligatures and especially contextual swashes) require open slots in the base encoding. Although LCDF Typetools is clever in deciding which glyphs should be discarded from an over–full encoding, a user should keep this limitation in mind.

Opticals

Most fonts that TEX users are familiar with include only two design axes, namely weight (e.g., light, regular, or bold) and shape (e.g., italic, slanted, or roman)[5]. Back in the days of metal type, each size of a given font had different design characteristics, because the eye is sensitive to different features at different scales. "Optical" fonts essentially add another design axis. This design axis was well–developed in the days of multiple master fonts, but since that technology appears to be dying, premium fonts are now being issued at discrete points along that axis. The font package that was used in the development of these macros and typescripts typically included four optical font sizes for each font: caption, regular, sub–head, and display. Their differences are shown in figure 4.

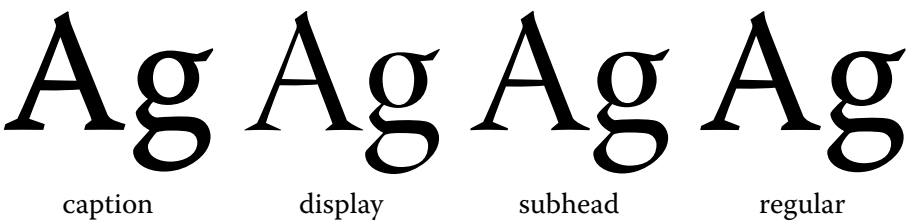


Figure 4 The four design sizes of Warnock Pro Opticals, shown at the same point size

At small design sizes (caption), there is typically lower contrast, a heavier stroke, a slightly larger x-height, and generally courser features. At large design sizes (display), strokes, tapers, serifs, and other details are much more refined.

This package supports these various design sizes with a series of extensive typescripts. It's essentially a brute–force method that defines font synonyms for each design size for each variant. That is, support for opticals is achieved by using a typescript that has small, regular, large, and extra–large fonts named for each of roman, italic, bold, and bold italic font variants, and adapting the `\tfa`–`\tfd` switching for each body font size.

For example, there are font synonyms declared for each of the following: SerifCaption, Serif, SerifSubhead, SerifDisplay, SerifItalicCaption, SerifItalic, SerifItalicSubhead, SerifItalicDisplay, and so on. . . . Each of these symbolic names is tied to font commands through the definition of a very large "Opticals" typescript, which generically associates a type variation and a type size with a design size. An extract follows:

```
\starttypescript [serif] [Opticals] [size]
\definebodyfont
  [12pt,11pt][rm]
  [tf=Serif sa 1,
  tfa=SerifSubhead sa \magfactor1,
  tfb=SerifSubhead sa \magfactor2,
  tfc=SerifSubhead sa \magfactor3,
  tfd=SerifDisplay sa \magfactor4,
   it=SerifItalic sa 1,
  ita=SerifItalicSubhead sa \magfactor1,
  itb=SerifItalicSubhead sa \magfactor2,
  itc=SerifItalicSubhead sa \magfactor3,
  itd=SerifItalicDisplay sa \magfactor4,
  ...]
\stoptypescript
```

As the bodyfont size changes (`12pt,11pt`, above), the relationships between the opticals change. This is handled in a huge typescript. In order to use this typescript yourself, you should define each of the SerifCaption . . . SerifItalicDisplay synonyms in your own typescript, and include that with the Opticals typescript.

For example, I defined the various opticals for the Warnock font in a typescript called `WarnockProSiz`. I first created a typescript that associated the optical sizes with the actual font names as installed by TEXFONT:

```
\starttypescript [serif] [WarnockProSiz] [texnansi]
 \definefontsynonym[SerifCaption][texnansi-WarnockPro-Capt]
                    [encoding=texnansi,handling=pure]
 \definefontsynonym[SerifText][texnansi-WarnockPro-Regular]
                    [encoding=texnansi,handling=pure]
 \definefontsynonym[SerifSubhead][texnansi-WarnockPro-Subh]
                    [encoding=texnansi,handling=pure]
 \definefontsynonym[SerifDisplay][texnansi-WarnockPro-Disp]
                    [encoding=texnansi,handling=pure]
 ...
\stoptypescript
```

I then created a Warnock typeface to tie my size synonyms with the Opticals typescript:

```
\starttypescript[Warn]
\definetypeface[war][rm][serif][WarnockProSiz][Opticals][encoding=texnansi]
\definetypeface[war][rc][romancaps][WarnockProSiz][Opticals][encoding=texnansi]
\stoptypescript
```

If SerifDisplay
you can SerifSubhead
read all of this SerifSubhead
without squinting at all, SerifSubhead
you have better eyesight than Serif
I do. In fact, these fonts can get quite Serif
small without losing all that much legibility! SerifCaption

Figure 5 The `\tfd...\tfxx` series with a base size of 12pt.

Small Caps

Current support for small caps, both inside and outside TEX, is generally very primitive. Most fonts – if they do offer it – only offer small caps support as a variation on the plain, roman font, and not for any italic/slanted fonts or bold fonts. This means that the small caps shape is of limited use with non–normal font alternatives (what ConTEXt calls `\it` and `\bf`). With a full complement of small caps shapes for each font alternative, small caps can be used more extensively.

The approach chosen for this package was to create another serif font style to exist inside the serif family, alongside the familiar roman (`rm`) style. The new font family, roman caps (`rc`) defines parallel font alternatives, using small caps variants. This means more work in terms of defining font synonyms, but it enables the small caps shape as a full design axis. The definitions begin as follows:

```
\definebodyfont
  [12pt,11pt][rc]
  [tf=SerifCaps sa 1,
  tfa=SerifCapsSubhead sa \magfactor1,
  ...]
```

. . . and proceeds as the other definitions, above. There are sans serif equivalents defined, as well. The sans small caps family (`cs`, caps sans) is treated the same way. If you have installed a small caps type variant for a sans serif font, you should define and use this parallel family.

THIS

Figure 6 `\rc\bf this`

In order to use these font families, you may call them directly with macros like `{\rc\bf this}`. This is inconvenient, and requires you to recall the font alternative as well as the roman caps font style. To alleviate the inconvenience, the support files for this article define a new font command, which switches from the normal style to the caps style while keeping the current alternative. The command is `\SmCap`, and it is used grouped, like other font commands. The following text in the margin is achieved with the code:

```
{\it text {\SmCap text \em text} text \fontstyle\ \fontalternative}
{\bf text {\SmCap text \em text} text \fontstyle\ \fontalternative}
{\rc text {\SmCap text \em text} text \fontstyle\ \fontalternative}
```

text TEXT TEXT *text rm it*
text TEXT *TEXT* text rm bf
TEXT TEXT *TEXT* TEXT RC TF

There is an identical command `\OldStyle`, which assumes that there are old style figures defined in the small caps family. It works in the same way as the above:

```
{\tf 0123456789 \OldStyle 0123456789}\crlf
{\bi 0123456789 \OldStyle 0123456789}\crlf
{\itx 0123456789 \OldStyle 0123456789}\crlf
```

0123456789 0123456789
0123456789 0123456789
0123456789 0123456789

Known issues

The sheer number of fonts defined means that these typescripts are extremely memory–intensive. A modern computer should not struggle, as long as the font memory allocated to ConTEXt is sufficient.

The default compound hyphen that is used in composed words is a hybrid character that doesn't work well in some pro fonts, including Warnock. I think it is best to replace the default compound character with a stretched hyphen with the command:

well–hung
hang–dog

Figure 7 ConTEXt's default compound hyphen vs. a stretched hyphen.

```
\def\compoundhyphen{\scale[sx=1.5]{-}
```

Math support is preliminary and still considered a work–in–progress, but it certainly seems possible to fashion a math font from sufficiently complete (e.g., containing Greek language support) fonts.

Footnotes

1. see http://www.adobe.com/education/ed_products/typeclassics.html
2. see http://homepage.mac.com/atl/tex/
3. née PfaEdit, see http://fontforge.sourceforge.net/
4. Available from http://www.lcdf.org/type/
5. Yes, it's true that the ubiquitous ComputerModern has many design sizes.

Adam T. Lindsay
Lancaster University
UK
atl@alum.mit.edu

Fontgebruik
Een uitdaging voor de lezer

Abstract
Hans Hagen presenteert een zeer overdadige titel- en achterpagina van een handleiding uit 1899. De redactie van de Maps looft een prijs uit voor de beste en elegantste TEX-benadering van deze layout.

Er zijn ongeschreven regels met betrekking tot het aantal verschillende fonts dat op een bladzijde redelijkerwijs mag worden gebruikt. Het aardige van een op TEX gebaseerd systeem is dat er in de regel systeem zit in de samenhang tussen de gebruikte fonts, al is het alleen al in de afmetingen. Iets dergelijks geldt voor kleurgebruik.

Mijn dagelijkse praktijk heeft me inmiddels geleerd dat veel vormgevers het daarmee niet zo nauw nemen. Een vertaling van een in een DTP-systeem gemaakt ontwerp naar een TEX-stijl levert dan ook vaak een rommelig pakket op van font- en kleurdefinities.

Dat uitbundig fontgebruik niet louter van onze tijd is, toont de meer dan honderd jaar oude voorpagina in figuur 1 op de volgende pagina.

De achterpagina maakt het nog wat bonter (zie figuur 2).

De verschijningsdatum van het boekje doet vermoeden dat het de productie van scharreleieren betreft. Echter, de hedendaagse lezer kan zich niet aan de indruk onttrekken dat het verhaal betrekking heeft op een sterke structurering van het voederproces en derhalve mechanisatie. Hoe dan ook, de buitenkant van het drukwerk lijkt nog duidelijk handwerk en roept het beeld op van een zelfgemaakte uitnodiging voor een feest van iemand die zichzelf het tekstverwerken aan het eigen maken is. Maar . . . is zoiets wel na te maken met het ons zo dierbare TEX? Met de Computer Modern Fonts moet men een heel eind komen, afgezien van de onderstrepingen, die in geval van TEX niet tot de fontkarakteristieken horen maar ‘met de hand’ moeten worden toegevoegd. Overigens, oordeel niet te hard over die onderstrepingen, tegenwoordig doen we dat met kleur.

De redactie van de Maps stelt een boekenbon van 50 euro beschikbaar voor de (ten eerste) beste en (ten tweede) elegantste TEX-benadering.

De deadline is 31 maart 2005, zodat de resultaten in de volgende Maps gepresenteerd kunnen worden.

Hans Hagen
pragma@wxs.nl
Maps Redactie (voor inzendingen)
maps@ntg.nl


De Eier- en Vleeschproductie

van het Hoen.

(Zijn nut en voordeel.)

PRACTISCHE HANDLEIDING

VOOR

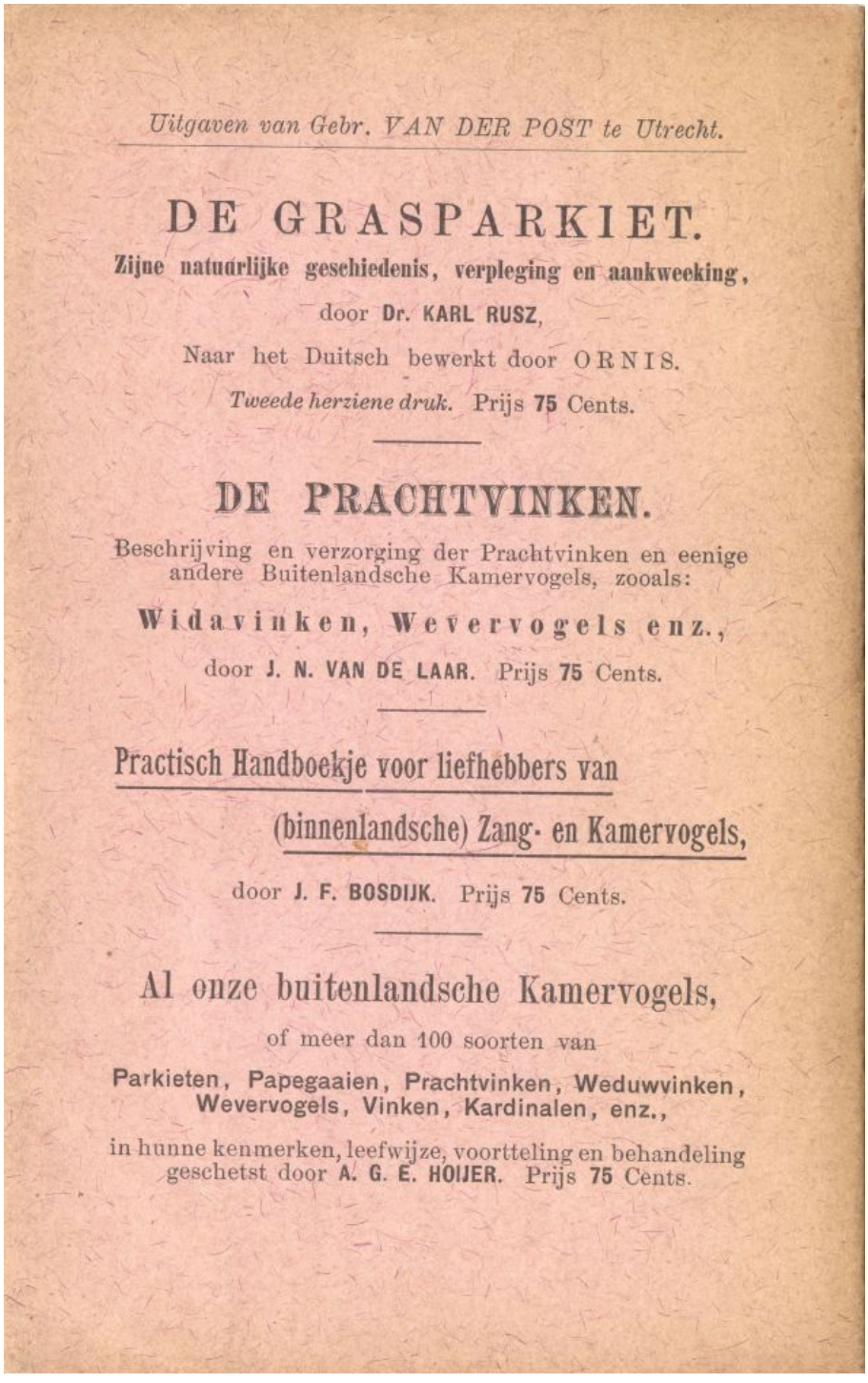
Fokkers en Liefhebbers van Hoenders.

DOOR

F. VAN ESTA TJALLINGII.

UTRECHT. — GEBR. VAN DER POST. — 1899.

Figuur 1 Een voorpagina



Uitgaven van Gebr. VAN DER POST te Utrecht.

DE GRASPARKIET.

Zijne natuurlijke geschiedenis, verpleging en aankweeking,

door **Dr. KARL RUSZ,**

Naar het Duitsch bewerkt door ORNIS.

Tweede herziene druk. Prijs **75** Cents.

DE PRACHTVINKEN.

Beschrijving en verzorging der Prachtvinken en eenige andere Buitenlandsche Kamervogels, zooals:

Widavinken, Wevervogels enz.,

door **J. N. VAN DE LAAR.** Prijs **75** Cents.

Practisch Handboekje voor liefhebbers van

(binnenlandsche) Zang- en Kamervogels,

door **J. F. BOSDIJK.** Prijs **75** Cents.

Al onze buitenlandsche Kamervogels,

of meer dan 100 soorten van

Parkieten, Papegaaien, Prachtvinken, Weduwvinken, Wevervogels, Vinken, Kardinalen, enz.,

in hunne kenmerken, leefwijze, voortteling en behandeling geschetst door **A. G. E. HOIJER.** Prijs **75** Cents.

Figuur 2 Een achterpagina

Conversies

Een terugblik op 12 jaar met een softwarepakket dat op één of andere manier met vriendschappen te maken heeft

TEX is geen pakket dat je in de winkel gaat halen. Je vindt het via een vriend, een studiegenoot of collega en vanaf het moment dat je je met enige hulp door de steile leercurve heen hebt geworsteld hoor je bij het clubje.

Het houdt echter niet op bij die eerste drempel. De ontreddering die de radicaal andere manier van "teksten verwerken" teweeg brengt en de verrassing als je het voor het eerst door begint te krijgen, die komen steeds terug. In mijn geval tenminste wel. Bij het installeren van één van de vele honderden plug-ins, zoals style files en fonts. En bij het ontdekken van weer andere totaalpakketten binnen TEX, zoals ConTEXt. Of bij het upgraden naar een nieuwe versie van een bestaand pakket.

Een paar maanden geleden ging ik over op de Apple en naast alle verrassende nieuwe en aangename dingen als de vormgeving en vooral de geruisloze werking van de Powerbook was de grootste opluchting dat ik 'gewoon' door kon gaan met TEX.

Toen ik begon aan dit artikel, bedoeld als leidraad voor wie ook naar de mac over wil, merkte ik echter dat ik al niet meer precies weet hoe het nu feitelijk is geïnstalleerd, wat waar staat, wat waar zou moeten staan, welke `psfonts.inf` en `.map`-file nu wordt gebruikt… Dat is toch wonderlijk. Ben ik nou zo dom of is de rest nou zo slim?

Werken met TEX komt voor mij neer op een steeds terugkerende conversie. Altijd is er wat, het buskruit moet telkens opnieuw uitgevonden en steeds weer moet ik nagaan waar de huidige Abraham de typografische mosterd ook alweer had gehaald.

Begin 1992 dook TEX in mijn leven op. Ik was met mijn maat Johan Polak in Aken en ik had een afspraak met vriend Martin Pauly die me die dag over LATEX vertelde. Johan en ik schreven columns een krant en twee tijdschriften. De ontdekking van LATEX kwam ter sprake in twee stukjes.

NTG en vrienden

De floppies die ik van Martin kreeg, deden wel wat op mijn PC maar voor een Nederlandse versie vond ik gelukkig de NTG. Ik belde bestuurslid Jos Winnink thuis en tot mijn verrassing was hij erg enthousiast over de nieuwe gebruiker die zich kwam melden voor de Nederlandse set van vijftien floppies. Nadat die een voor een waren uitgepakt stond er een werkend TEX-systeem op mijn PC. De stapel floppies ging weer terug met de post.

Zou die eerste set van een TEX distributie nog ergens op een boekenplank liggen?

Internet was nog niet voor de thuiscomputeraar beschikbaar en via een ander netwerk, het FIDO-net dat wereldwijd duizenden computers via modems en een handig postsysteem met elkaar laat communiceren, leerde ik Henk de Haan kennen. Hij was als natuurkundige al een poos bezig met zijn proefschrift over het verwerken van meetgegevens bij proefnemingen met kernfusie die door muonen werd gekatalyseerd. Op de middelbare school was hij vanwege zijn rustige voorkomen aanvankelijk wat te laag ingeschat, gaandeweg door zijn hoge cijfers opgevallen en uiteindelijk met hoogste score geslaagd voor het gymnasium. Henk had tijdens het (talmen met) werk aan zijn dissertatie tijd te over om me een en ander per mail uit te leggen en vaak pakte ik de telefoon om hem te vragen hoe een bepaalde stylefile moet worden geïnstalleerd. Henk, voor wie er geen 'moeilijke' dingen bestaan, vond het leuk om ze aan een wat minder begaafde makker uit te leggen en we besloten meer samen te gaan doen. Henk deed het denk- en programmeerwerk voor ons bulletin board systeem en enkele jaren lang was mijn PC een dag en nacht open staande gratis bibliotheek van allerlei TEX-bestanden die via modem konden worden opgehaald. Tot het internet voor thuisgebruikers doorbrak beleefde het systeem, deels gesubsidieerd door de NTG, drukke tijden. Henk had het er druk mee, het plezier daarom gaf hem de energie om ook wat harder te werken aan zijn proefschrift, hij promoveerde en kreeg een baan bij een bedrijf dat toepassingen ontwikkelt voor *smart cards*.

Van Henk heb ik de discipline geleerd om een bestand dat bij compileren fouten oplevert, terug te brengen tot de versie die nog wel werkt en van daaraf verder te gaan tot de fout optreedt, dan een bestandje te maken dat zo klein mogelijk is en precies die fout oplevert. Vaak is de fout dan al duidelijk geworden

COLUMN Het hotel in Aken heeft sinds kort geen portier meer. De laatste die deze taak van verwelkomer, uitzwaaier en vertrouwensman vervulde, is met pensioen gegaan. Hij bleef kalm, wat er ook gebeurde in de tijd die hij zag verstrijken, zelfs toen een in hagelwit geklede stoet woestijnprinsen arriveerde, elk met een jachtvalk op de schouder.

Eenmaal binnengekomen, vinden wij de receptie, 'bemand' door een jonge vrouw achter een kubusvormig beeldscherm met een toetsenbord zo dik dat het een bomaanslag zou kunnen doorstaan. We ontvangen elk een kamersleutel, groot en zwaar alsof wij er de stadspoort mee moeten openen. Mijn kamer is eenvoudig en zwelgend in ruimte: de garderobe is een kamer op zichzelf, de slaapkamer biedt een bed om in te slapen, een om in te ontwaken. De badkamer doet een kuur-zwembad concurrentie aan, maar nodigt door zijn koelte en metershoge vensterraam nauwelijks tot ontspanning. Langs een wand strekt zich een spiegel. In onbarmhartig scherp licht oogt de gast alsof zijn laatste dag is aangebroken. Johans kamer is kleiner en knusser ingericht, met twee televisietoestellen waarop weinig valt te zien. We eten in het restaurant, waar ons verrukkelijke gerechten worden voorgezet. Elke gang komt onder zilveren stolpen, die door serveersters met enig elan voor ons worden weggenomen. Wanneer het toetje is verorberd, is ook de avond voorbij.

De volgende dag ontbijten we samen met Martin Pauly, een vriend, eerstejaars student aan de technische universiteit. Martin publiceerde als gymnasiast reeds gespecialiseerde computerboeken en kan dankzij royalties vrij goed rondkomen – op zijn kamer bevindt zich meer hardware dan menig automatiseerder op kantoor heeft staan. Terwijl Johan in het hotel werkt aan een lang artikel voor een geleerd tijdschrift, breng ik een dag door met Martin. Hij laat mij een inwijding ondergaan in de fantastische mogelijkheden van LaTeX, oude reus onder de tekstverwerkers, en hij demonstreert een vliegsimulator. Computerscherm wordt cockpit, uit luidsprekers rondom ons klinken straalmotoren en het zoemen van een inklappend landingsgestel terwijl we 'opstijgen' onder goedkeurend gemompel van de verkeersleiding.

We lunchen in de mensa en nemen elk een piepschuimen dienblad met kuiltjes waarin door keukenjuffrouwen vlug onze gerechten zijn geworpen. We moeten onze dienbladen snel wegpakken omdat er anders nieuwe op worden gelegd. Het valt mij op hoe oud de studenten zijn – gemiddeld vijfentwintig jaar – en tegelijk hoe kinderlijk. Er wordt snel gegeten, nauwelijks gepraat. Folders met goedkope computeraanbiedingen worden gretig gelezen of als servet gebruikt. Wie klaar is, stapt op en ontdoet zich van zijn eetgerei. Drie manshoge vuilnisbakken staan hiertoe gereed en een plexiglazen valluik in de wand. De een gooit alles – dienblad, etensresten, bestek – in de vuilnisbak, de ander werpt de resten in de emmer en laat het plastic achter het luik zakken. Wij keilen alles achter het plexiglas, waar een kokend hete waterval zich erover ontfermt en het meeneemt in een diep, traag stromend huishoudkanaal. Niemand weet waar dat heenvoert en niemand die het iets uitmaakt.

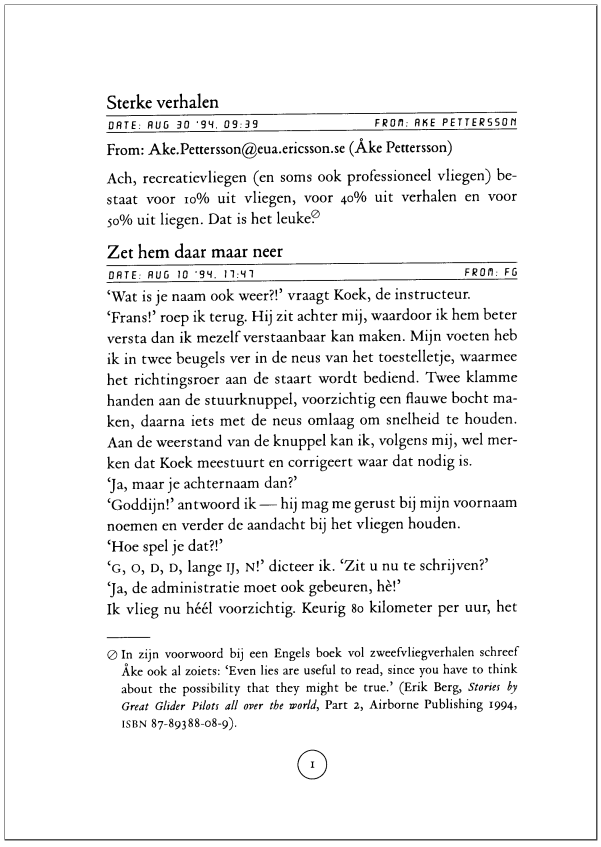
en anders is die voor een ander via TEX-NL snel te herkennen. Henk zijn sterke kant als TeX-gebruiker is zijn enorme geheugen: een bespreking van een stylefile die hij eenmaal heeft gelezen, onthoudt hij feilloos en hij maakt perfect gebruik van al het werk dat zoveel programmeurs in het schrijven van die vele honderden toeters en bellen hebben geïnvesteerd.

Via mijn oudste dochter leerde ik intussen Herman Haverkort kennen. Hij studeerde informatica, had weleens van TeX gehoord maar hij had zich er nog niet in verdiept. Hij kwam kijken hoe ik mijn brieven en andere documenten produceerde. Ik legde hem in een kwartiertje uit hoe dat zo werkt met TeX en dat waren ook de laatste minuten dat ik er meer van af wist dan hij. "Mag ik even?" vroeg hij zodra ik was uitgepraat en hij zette zich aan mijn toetsenbord. Zijn vingers roffelden, macro's rolden over het scherm, directories werden bekeken en Herman was enthousiast. Hij ging naar huis met het TeXbook van Knuth onder de arm, las het snel waarbij hij zich concentreerde op die paragrafen waar een "gevaarlijke bocht" symbool naast staat en hij werd een nieuwe TeX-makker.

Herman ging op eigen wijze om met de TeX-problemen en vragen die hij tegenkwam: teneinde eventuele compatibiliteitsproblemen tussen stylefiles van derden te voorkomen schreef hij bijna alles zelf. Er kwam een zwik stylefiles uit zijn handen waarvan de namen beginnen met HH, zijn initialen. Stylefiles die met elkaar samenwerken als multifunctionele legosteentjes. Stylefiles voor fonts, voor bladindeling, voor bijzondere symbolen. Dankzij Hermans hulp kon ik voor een kleine Arnhemse uitgeverij een schoolboek typesetten met lastige opgaven: een quiz, een kruiswoordpuzzel, aparte tabellen.

En voor een ander boekje bedacht Herman alternatieve paginanummersymbolen: dobbelsteentjes en streepjes waarmee je van 1 tot vijf kunt "turven."

Herman is nu gepromoveerd informaticus en hij stuurt zijn vrienden mails vol anekdotes uit Japan, Duitsland, Scandinavië of Amerika waar hij in aange-



Sterke verhalen

DATE: AUG 30 '94, 09:39 FROM: AKE PETTERSSON

From: Ake.Pettersson@eua.ericsson.se (Åke Pettersson)

Ach, recreatievliegen (en soms ook professioneel vliegen) bestaat voor 10% uit vliegen, voor 40% uit verhalen en voor 50% uit liegen. Dat is het leuke.⊘

Zet hem daar maar neer

DATE: AUG 10 '94, 17:47 FROM: FG

'Wat is je naam ook weer?!' vraagt Koek, de instructeur.
'Frans!' roep ik terug. Hij zit achter mij, waardoor ik hem beter versta dan ik mezelf verstaanbaar kan maken. Mijn voeten heb ik in twee beugels ver in de neus van het toestelletje, waarmee het richtingsroer aan de staart wordt bediend. Twee klamme handen aan de stuurknuppel, voorzichtig een flauwe bocht maken, daarna iets met de neus omlaag om snelheid te houden. Aan de weerstand van de knuppel kan ik, volgens mij, wel merken dat Koek meestuurt en corrigeert waar dat nodig is.
'Ja, maar je achternaam dan?'
'Goddijn!' antwoord ik — hij mag me gerust bij mijn voornaam noemen en verder de aandacht bij het vliegen houden.
'Hoe spel je dat?!'
'G, O, D, D, lange IJ, N!' dicteer ik. 'Zit u nu te schrijven?'
'Ja, de administratie moet ook gebeuren, hè!'
Ik vlieg nu héél voorzichtig. Keurig 80 kilometer per uur, het

⊘ In zijn voorwoord bij een Engels boek vol zweefvliegverhalen schreef Åke ook al zoiets: 'Even lies are useful to read, since you have to think about the possibility that they might be true.' (Erik Berg, *Stories by Great Glider Pilots all over the world*, Part 2, Airborne Publishing 1994, ISBN 87-89388-08-9).

1

naam gezelschap tussen andere vrolijke bollebozen nadenkt over algoritmes die op den duur zouden kunnen leiden tot het eenvoudig uitrekenen van dingen die nu nog niet te becijferen zijn.

Wat ik vooral van Henk en Herman had willen leren maar me niet eigen heb kunnen maken is hun houding ten opzichte van TEX-penarie. De kalme nieuwsgierigheid waarmee zij kijken naar iets dat het niet doet… nooit heb ik die onder de knie gekregen. De doolhoven en onverwachte resultaten van TEX blijven me nog steeds dwars zitten. Ik word niet nieuwsgierig van niet-werkende TEX documenten, ik word er naar van!

Intussen was 4TEX verschenen, een volledig werkend TEX installatiepakket voor de PC. Nog voor de Windows-tijd! Vandaag de dag misschien verouderd, maar tot heden is er geen pakket dat in vergelijkbare mate volledigheid en gebruiksgemak combineert. Een simpel configuratiebestandje, makkelijk te vinden, klaar! Wietse Dol die samen met Erik Frambach de 4TEX CD onder de paraplu van de NTG op de wereld heeft gezet, had een drukke baan bij het Ministerie voor Landbouw en Visserij waar hij een slimme database had aangelegd om vangstquota voor vissers in de gaten te houden. Maar voor een mede-TEX-gebruiker was hij altijd te storen. Ook bij hem die opgewekte energieke aanpak van "ha, een probleempje!!" in contrast met mijn "oh shit... wat nu?"

Na 4TEX was er een terugval in gemak voor de gebruiker. Op de TEXLive CD's is wel veel meer te vinden, maar hoe dat allemaal is te bereiken… Ook werd het na 4TEX opnieuw moeilijker om instellingen te veranderen. De meest gebruikte instellingen worden bewaard in een cryptisch bestand als `texmf.cnf` dat ook nog eens meters diep in de directorystructuur is weggestopt. Waarom toch? Zoekpaden worden er voor de expert aangegeven met uitroeptekens en andere symbolische afkortingen. Dan zijn er voor lettertypes weer andere bestanden elders weggezet, onder andere `psfonts.inf` en `psfonts.map` files. En van deze essentiële bestanden hangen er dan nog verschillende kopieën aan bijna identieke boomstructuren naast elkaar, waardoor het weer wat minder duidelijk wordt welke nu moet worden bekeken of aangepast. Vervolgens wordt die structuur jaarlijks aangepast, herzien, omgegooid.

Lang heb ik het gevoel gekregen dat de superslimmen die de diverse distributies samenstellen heimelijk plezier hebben om de minder begaafden als ik, die pas na enige tijd door hebben hoe het werkt. En hun lol is dan me telkens weer te frapperen door de boel om te gooien zodra ik begin door te krijgen hoe het zit.

Nu denk ik dat het anders ligt. Als niets moeilijk voor je is, heb je ook niet de neiging het simpeler te maken. En als een stel beheerders er met elkaar niet uitkomt, dan is het voor henzelf eenvoudiger om er gewoon een hele directorystructuur bij te zetten waar de ander zijn gang kan gaan. Misschien is het dat. En zo hebben we gigabytes vol mappen en structuren en duizenden verstrooide bestandjes die ooit van belang waren of misschien nog zijn. Voor de insiders geen enkel probleem, voor eenvoudige gebruikers is het alsof er een "big bang" wordt nagespeeld, een razendsnel uitdijend universum aan TEX-bestanden waar alleen sterrenkundigen de weg weten.

Toen de LaTEX-wereld overstapte naar versie 2e begreep ik eerst niet waarom iemand als Gerard van Nes de boel liever bij het oude liet op het systeem dat hij beheerde bij ECN in Petten. Al die aanpassingen door het LaTEX team zouden toch ergens toe dienen? Aan energie heeft het Gerard nooit ontbroken want juist hij zorgde er lange tijd voor dat de halfjaarlijkse NTG-bijeenkomsten vloeiend verliepen en hij was als hoofdeindredacteur van de Maps de actieve spil in de tijd dat de Maps steeds dikker en mooier werd. Toen ik een boekje met verhalen over zweefvliegen had samengesteld, zorgde Gerard ervoor dat er met wat PERL

scripts een web-versie van kwam. Geen moeite teveel voor een collega TEX-gebruiker! Waarom dan niet mee met de vooruitgang en het ECN-systeem optuigen met de nieuwste LaTEX? Ik kan d'r nu wel inkomen. Het systeem werkte, het zou uit zichzelf niet stoppen met functioneren en de bestaande artikelen zouden zo over jaren nog steeds kunnen worden gecompileerd. Door alles vast te zetten, maakte hij van het systeem bij ECN wel een beetje een eiland in een blauwe oceaan, ver van de grote vaarroutes. Maar het leven op dat eiland bleef wel overzichtelijk.

Ik ben telkens gezwicht voor de veelbelovende vergezichten en waar anderen op hun speedboat voorgingen, trok ik er op mijn vlotje achteraan. En er was altijd wel een TEX-makker die me een poosje op sleeptouw nam. Dat blijft een van de mooie aspecten aan de TEX-gemeenschap. Misschien heet de Amerikaanse Maps daarom ook wel TUGboat.

Dankzij zulke voortrekkers kon ik al snel na de eerste installatie van TEX een zelfgekocht font inpassen. Ik koos voor het Van Dijck lettertype omdat die in de *expert set* niet slechts één stel 'hangende cijfers' meenam, maar een hele zwik! Wel vijf verschillende hangende vijfjes! Wietse Dol was direct bereid de benodigde files te maken die het volledige gamma aan letters, cijfers en kleinkapitalen binnen 4TEX bruikbaar maakten. Ik had het bestaan van zulke cijfertjes nog maar net ontdekt en het leek me prachtig ermee te spelen in teksten, kopjes, voetnoten… Later begreep ik wel dat dit de leesbaarheid van de tekst niet bijzonder verbetert. Typografie slaat op hol zodra de letters en cijfers zelf aandacht gaan vragen van de lezers, in plaats van simpelweg te vertellen wat ze in woorden en getallen te zeggen hebben. Ik zou nog weleens naar dat font willen kijken, maar de *font handling* in TEX is dermate veranderd dat ik er zonder hulp nooit uit zo komen. Dus slaapt de uitgebreide Van Dijck fontfamilie ergens op mijn hard disk.

Als alternatief kocht ik de *expert set* bij het Times font dat met OS/2 was meegeleverd. Herman zorgde voor de stylefile en de regels voor postscript en bitmap fontinformatiebestanden. Bij elke nieuwe installatie is het weer even kijken waar die bestanden ook alweer staan en waar ze heen moeten zodat TEX ze vinden kan.

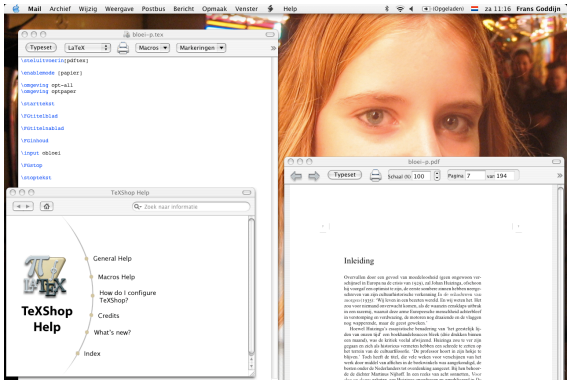
Een volgende grote conversie was de gedeeltelijke overstap naar ConTEXt. Ik had al vaak met begerige ogen gekeken naar de mogelijkheden die dit macropakket biedt. Zo veel toeters en bellen, en alles in een keer beschikbaar zonder ooit nog aparte stylefiles te hoeven installeren. En een verbluffend simpele bediening, met overzichtelijke commando's die de aansturing van ingewikkelder zaken onder de motorkap voor hun rekening nemen. Hans Hagen is onvermoeibaar aan de telefoon als ik dingen niet begrijp en vaak legt hij zelfs meer uit dan ik hem vraag. Voor de aardigheid liet hij me zien hoe ik, in de kop van mijn bestaande briefhoofd, een logo kan laten zetten. Het bijzondere aan dit logo is dat het elke keer net iets anders wordt. Tevoren kan in een percentage worden aangegeven hoeveel het logo mag afwijken van de bron. Zo krijgt elke brief iets unieks dat toch herkenbaar blijft!

Binnen een reeds werkende ConTEXt-omgeving kan ik intussen best een andere tekst zetten, maar dat is geen verrassing. Veel meer dan dat heb ik alleen nog niet onder ConTEXt verwezenlijkt. Als ik zelf aan pagina-maten ga frunniken, dan loopt bijvoorbeeld de plek van het paginanummer weg van waar ik 'm wil. En als ik de macro's wil doorgronden dan duizelt het me van de hoeveelheid mogelijkheden van variabelen en aanwijzingen tussen vierkante haken en accolades. Handleidingen zijn er te kust en te keur, in de vorm van prachtige PDF bestanden, gratis te halen, maar ik heb zo graag een grote dikke handleiding vast, waar dan liefst heel simpel in staat wat je moet invoeren voor een bepaalde uitvoer. ConTEXt *for dummies* dus.

Het aan de praat krijgen van mijn Times font in deze nieuwste TEX-omgeving was echter ook voor Hans wat hoog gegrepen. Hij is al blij dat hij zijn eigen fonts in het gareel houdt want onder zijn steeds veranderende ConTEXt blijft vooral ook de TEXLive installatie telkens wijzigen. Voor Taco is het goochelen met fonts meer routine en hij bouwde een set bestanden voor me zodat een eerste boek dat ik in scherm- en papierversie met behulp van ConTEXt ging typesetten in Times kon worden gezet. Dit werk is door Willi Egger voltooid. Doordat het zo bewerkelijk is om met fonts te slepen, kan ik echter niet zonder meer zijn versie compileren en hij kan die in mijn font niet runnen.

Toen ik aan een nieuwe laptop toe was, adviseerde Siep Kroonenberg me naar de Apple te kijken. Weer eens wat heel anders dan het Windows systeem en niet zo gecompliceerd als de Linux installatie op een PC waaraan ik me weleens had gewaagd. Mooi om te zien en… er is uitstekend op te TEX-en!

De installatie van TEX op mijn nieuwe Apple Powerbook G4 liet ik over aan de TEXshop i-Installer op de site (http://www.rna.nl/ii.html) van Gerben Wierda. Tot mijn verbazing werkten ook mijn eigen stylefiles voor briefopmaak en mijn Times font, nadat ik zo simpel mogelijk via een externe USB-harddisk de "mytexlive" directorystructuur van mijn PC had gekopieerd naar mijn Powerbook bureaublad en vandaar naar de gebruikersmap frans/texmf et cetera. Het werkt. Tot mijn dagelijkse verbazing. Voor een aantal documenten grijp ik wel terug op mijn Windows XP systeem dat is verhuisd naar een klein achterkamertje. De laserjet die daar via het oude type printerkabel aan hangt is net wat scherper dan de USB-inktspuit aan de mac. Ik mis wel de vertrouwde kracht van de WinEdt editor. De gratis editor van TeXShop is niet slecht maar het biedt niet zo heel veel mogelijkheden. Een alternatief



als BBedit is weer behoorlijk aan de prijs. En vanuit TeXShop wordt een eigen PDF previewer gebruikt terwijl de gratis Reader van Adobe veel beter is. Vooral als er een bitmap font is gebruikt. Dan biedt de eigen previewer een alarmerend beeld terwijl de Adobe Reader een scherp beeld toont van dezelfde PDF. Het komt op mij wat vreemd over dat ik op zo'n mooi systeem nog steeds "mktexlsr" op een command line zou moeten doen als er wat is aangepast. Wanneer ik dingen als "sudo" moet tikken om 'de baas' te zijn op mijn eigen laptopje, dan vind ik dat vreemd. Eigenaardig dat ik bestanden in de tetex directory niet kan wijzigen. 't Is toch zeker mijn eigen apparaat? De expert zal zeggen dat ik ook niets te zoeken heb op die directories, dat ik mijn eigen map heb voor mijn aanpassingen. De installer brengt de noodzakelijke aanpassingen aan in bijvoorbeeld fontmap-files waar ik dat liefst zelf zou doen zoals in het verleden mogelijk was.

Maar de laatste tijd denk ik dat ik niet meer moet uitbreiden, niet steeds weer bijleren of nog eens herontdekken hoe het ook alweer zit met installatie van toeters en bellen. Dat was, is en blijft een gebed zonder eind. Als ik de komende weken nu eens tijd besteed aan het *weghalen* van 'mijn' Times font uit mijn setup en ik haal ook zoveel mogelijk style files weg die ik in de loop van de tijd heb opgehaald, uitgepakt en ingebouwd. Dan worden mijn brieven misschien wat minder mooi, maar wat zou dat? Bovendien, misschien ontdek ik dat met het weglaten en loslaten van allerlei extra's het resultaat niet eens veel minder wordt. En als uiteindelijk mijn files zo simpel worden dat ze op om het even welke huidige internetinstallatie of CD/DVD-distributie kunnen worden gecompileerd, dan ben ik voor altijd verlost van de vraag hoe het ook alweer werkte, werken moet of wat er inmiddels is veranderd waardoor het weer anders werkt.

Herman Haverkort werkt nog met TEX, maar ook anders dan voorheen, schrijft hij me in een e-mail.

Hij stoeit veel minder met de toverdoos aan mogelijkheden. "Inderdaad, je kunt van alles uitprogrammeren, maar om dat de hele tijd werkend te houden als style-files waarvan je nog nooit had gehoord worden geupdate, of juist niet, terwijl de rest wel wordt geupdate, of om alles over te zetten naar een nieuw systeem met een nieuwe directory-structuur… Ik heb het ook al lang opgegeven. Ik gebruik alleen nog de MikTEX standaard-setup, zonder zelf geïnstalleerde fonts, en met alleen een paar niet-al-te-diepgravende style-files (hhpage, dat ik nooit heb gepubliceerd, vind ik nog steeds erg handig om automatisch marges voor niet-voorgedefinieerde papierformaten in te stellen). Meer durf ik niet meer. Ten eerste omdat ik aan veel dingen een tijdje werk, dan een half jaar niet, dan weer even wel: het ritme van conferentie-inzendingen, wachten op de beoordeling, definitieve versie schrijven, inzending naar een tijdschrift, wachten op de beoordeling enzovoorts brengt dat met zich mee. Als er in zo'n half jaar stilte iets verandert (bijv. nieuwe computer of verandering van werkgever waardoor ik moet werken met een nieuwe/andere TEX-installatie), betekent dat al gauw dat dingen niet meer compileren als je al te 'slimme' dingen heb gedaan. Ten tweede: sommige uitgevers verlangen TEX-files die ze kunnen compileren en ik wil ze gewoon een file kunnen sturen waarvan ik zeker weet dat eventuele problemen daarbij niets met mijn eigen gekke macropakketjes te maken hebben. Het is leuk dat je aan TEX alles kunt instellen zoals je wil, maar het gevolg is dat de portability (te) ernstig te lijden heeft…"

Natuurlijk bewaar ik wel ergens achteraf mijn oude directorystructuur met al die files die in de afgelopen 12 jaar zijn verzameld, aangepast of speciaal geschreven door mijn TEX-makkers. De bestanden zijn me dierbaar geworden. Mocht Siep eens langskomen, dan kunnen we opnieuw lekker ingewikkeld gaan doen!

Frans Goddijn
frans@goddijn.com

Exact layout with LaTeX

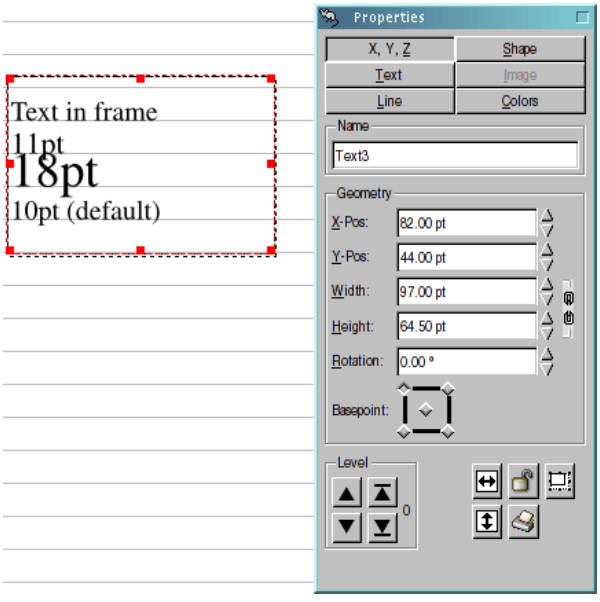
Implementing a letterhead

Keywords
letterhead, picture environment, boxes, PostScript, docstrip

Abstract
This article describes several techniques useful for implementing a professionally-designed layout such as a letterhead.

The Economics Department of Groningen University recently got new logos and an updated house style, including a new letter template.

The LaTeX implementation of the old version was done by Erik Frambach. Although I didn't have to change the basic structure very much, it still involved more than just changing some parameters. This article highlights some of the tricks, both new and inherited, that were used.



The page layout perspective

The previous time, the designer kept pointing out how much simpler and more natural things were with QuarkXPress, and he did have a point.

In case you don't realize what you are missing: the screenshot on this page shows how you can place text in a page layout application[1].

The dialog box contains entries X-Pos and Y-Pos for exact page coordinates of the box on the left. This is very different from TeX and wordprocessors, where positioning is usually relative to the text cursor, and positioning relative to the page requires frightening contortions.

Grid typesetting

More advanced page layout applications support typesetting on a grid of evenly spaced horizontal lines. If things don't fit within their allotted space then either that fact is ignored or the text jumps to the next gridline. The Scribus screenshot illustrates the former: the '18pt' line only occupies the default 12-point line height, and tangles with the line above.

Our letterhead also uses a 12-point vertical grid, although the grid is enforced only for the letterhead itself; TeX and LaTeX haven't been designed with grid typesetting in mind, and I know of no simple, reliable way to enforce grid typesetting in general with LaTeX.

Units

Most of the graphics industry works with 'big points', of which there are exactly 72 to the inch. The specified line height was 12bp. It seemed best to lift all fontsize definitions from size10.clo and modify them for big points.

I found TeXCalc[2], a utility from Taco Hoekwater for unit conversions, very useful.

Below, I'll mostly use big points for vertical measurements.

Using the picture environment

If you need to place items at specified positions, then the LaTeX picture environment comes to mind. This is how you use it:

```
\unitlength=1bp
\begin{picture}(..,..)(..,..)
picture commands
\end{picture}
```

The first pair of numbers is the size of the picture, in unit lengths; the second pair is optional and indicates

the coordinates of the lower left corner, which need not be (0,0).

Actually, the size is not necessarily the real size; it is the size that LaTeX is going to reserve for the picture. Therefore you can put in some extra whitespace by telling LaTeX that the picture is larger than it really is. You can also do the opposite: you can place picture objects at coordinates far outside the declared picture dimensions and LaTeX will typeset the rest of the page as if those objects aren't there.

There were two problems with the picture environment which had to be worked around:

□ The makebox command of the picture environment doesn't bother about baselines. In the picture below, we would like the descender of 'p' to extend below the baseline, but instead it sits on it:

```
\unitlength=1bp
\begin{picture}(12,15)(0,-5)
  \put(0,0){\line(1,0){12}}
  \put(0,0){\makebox(0,0)[bl]{p}}
\end{picture}
```

produces: p

□ A lot of syntax can't be used inside the picture environment. As a workaround, you can prepare material in advance.

Plain TeX boxes

LaTeX itself has various box commands, such as `\mbox`, `\parbox` and the `minipage` environment. However, working with boxes can become incredibly verbose and roundabout if you insist on using the LaTeX box commands. So here comes a quick introduction to plain TeX boxes, at least of what I am going to use of them[3].

TeX uses box registers, which have to be allocated, *e.g.*:

```
\newbox{\pbox}
```

You can fill and place this box:

```
\setbox\pbox=... % define box contents
\copy\pbox %place the box contents
\box\pbox % place and clear the box
```

There are hboxes, vboxes and vtops. They are different in what you can put into them, not in how you can use them. hboxes are like LaTeX mboxes, but you use different syntax to fill them. `\\` does nothing inside an hbox. vboxes and vtops are different: when you start filling such a box you are in internal vertical mode, and `\\` does work. I am not going into the intricacies of modes in boxes; if you want to put anything complicated inside, then the LaTeX alternatives may be more appropriate.

Each box has a reference point, which is normally on a baseline at the left edge. For an hbox there is only one baseline. Vtop and vbox boxes can have more baselines. The reference point for a vtop it is the top baseline and for a vbox the bottom one.

From what we saw in the above example, for placement purposes it seems smart to pretend that there is nothing below the reference point. To this end, we use the `\smash` macro, which pretends that depth is zero, and width and height too.

The following example shows how this works out inside a picture environment.

```
% first, turn off parindent
\newlength\parsave
\parsave=\parindent
\parindent=0bp

\setbox\pbox=\hbox{p\\q}%
\setbox\qbox=\vbox{p\\q}%
\setbox\rbox=\vtop{p\\q}%

% boxes ready, can turn parindent back on
\parindent=\parsave

\noindent
\unitlength=1bp
\begin{picture}(40,40)(0,-15)
  \put(0,0){\line(1,0){40}}
  \put(0,0){\makebox(0,0)[bl]%
    {\smash{\box\pbox}}}
  \put(20,0){\makebox(0,0)[bl]%
    {\smash{\box\qbox}}}
  \put(30,0){\makebox(0,0)[bl]%
    {\smash{\box\rbox}}}
\end{picture}
```

gives you correct alignment with the baseline:

p
pq q p
q

Since we can place objects freely anyway inside a picture environment, the choice between `\vtop` and `\vbox` is often a matter of taste. But if you do want a text block to start at a given point and the number of lines may vary, then a `\vtop` box is the way to go. The relevant part of the classfile might look as follows:

```
\def\@toname{}
\def\toname#1{\def\@toname{#1}}

\long\def\@toaddress{}
\long\def\toaddress#1{%
  \long\def\@toaddress{#1}}
```

```
\def\@frominfo{%
  This Isme \\
  MyStreet 99 \\
  9999 ZZ MyCity \\
  Phone 0123456789}

\newbox\frombox
\newbox\logobox
\newbox\tobox
\def\makeletterhead{%
  \setbox\frombox=\vtop{\@frominfo}%
  \setbox\logobox=\hbox{%
    \includegraphics[width=1in]{logo}}%
  \setbox\tobox=\vtop{\@toname \\\@toaddress}%
  \unitlength=1bp
  \begin{picture}(300,126)
    \put(280,192){\makebox(0,0)[bl]%
      {\smash{\box\logobox}}}
    \put(280,180){\makebox(0,0)[bl]%
      {\smash{\box\frombox}}}
    \put(0,180){\makebox(0,0)[bl]%
      {\smash{\box\tobox}}}
  \end{picture}}
```

You can use this classfile as follows:

```
\pdfoptionpdfminorversion=3
\documentclass{letterdemo}
\begin{document}
\toname{Some Body}
\toaddress{%
  Business Deparment \\
  Room 000 \\
  HisStreet 111\\AA 0000 HisTown \\
  Some Country}
\makeletterhead

MyCity, \today

Dear Some,

This is a sample letter. Hope you are well.

Regards,
\end{document}
```

For a full listing of the classfile, see the end of the article.

Printing a grid

In order to check placement visually, you can print a grid as part of the page header; see Figure 1. The file grid.eps is hand-written PostScript:

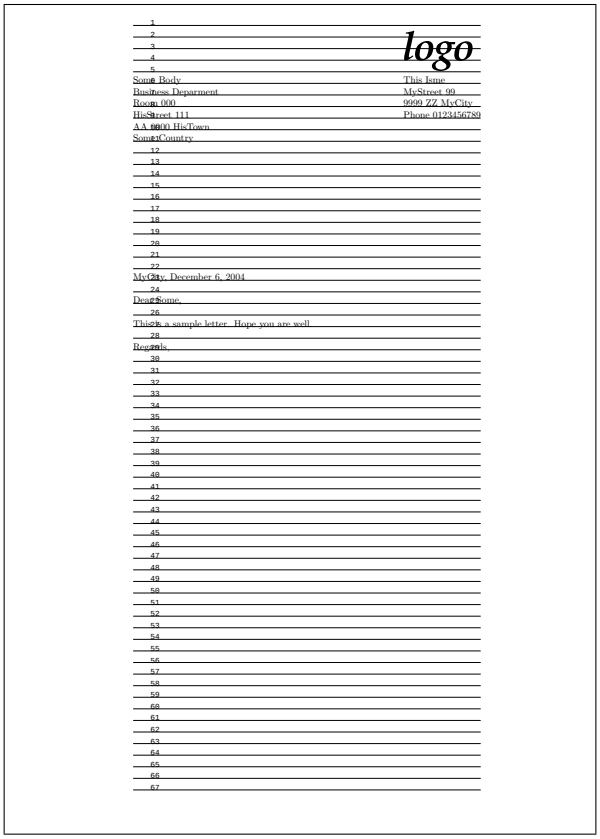


Figure 1. The letter printed with a grid

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 360 806
<< /PageSize [360 806] >> setpagedevice
0.4 setlinewidth
/Courier findfont 8 scalefont setfont
1 12 793 {
 newpath
 dup 0 exch moveto
 dup 360 exch lineto stroke
 dup 18 exch moveto
 dup 12 idiv 67 exch sub 2 string cvs show
} for
```

The nice thing of PostScript is that you can draw lines simply with commands `moveto`, `lineto` and `stroke`, using absolute page coordinates. This basic simplicity gets somewhat obfuscated by PostScript's lack of syntactic niceties, necessitating some juggling with `dup` and `exch`. So you may prefer to do the same with TeX macros.

Ghostscript will convert this to grid.pdf with the right boundingbox, thanks to the page definition on

the third line. Without this line, you need a script such as epstopdf for conversion.

You can place it on the page as part of the page header code:

```
\def\ps@debug{%
  \def\@oddhead{%
    \smash{\raisebox{-705bp}%
      {\includegraphics{grid}}}}%
    \let\@oddfoot\@empty}
\pagestyle{debug}
```

If you can't sort out the exact value of the first raisebox parameter then just use trial and error.

Removing debug code with docstrip

In this simplified example, there is only one piece of code that needs removing. In more complex cases, you can mark the debug code

```
%<*debug>
...
%</debug>
```

and use a docscript 'batchfile' `myletter.ins` to remove them:

```
\input docstrip
\generate{\file{myletter.cls}%
  {\from{letterdemo.cls}{!debug}}}
\endbatchfile
```

Run this batchfile as follows

```
\latex myletter.ins
```

to get a version `myletter.cls` of your classfile without debug code.

The full listing

```
\LoadClass[a4paper]{article}
\usepackage{graphicx}

% duplicate definition of normalsize from size10.clo, but
% use big points for normal line spacing
\renewcommand\normalsize{%
   \@setfontsize\normalsize{10pt}{12bp}
   \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
   \abovedisplayshortskip \z@ \@plus3\p@
   \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
   \belowdisplayskip \abovedisplayskip
   \let\@listi\@listI}
\normalsize

\textwidth=360bp

\parindent=0bp
\parskip=12bp

\pagestyle{empty}
%<*debug>
\def\ps@debug{%
  \def\@oddhead{%
    \smash{\raisebox{-705bp}{\includegraphics{grid}}}}%
    \let\@oddfoot\@empty}
\pagestyle{debug}
%</debug>

\def\@toname{}
\def\toname#1{\def\@toname{#1}}

\long\def\@toaddress{}
\long\def\toaddress#1{\long\def\@toaddress{#1}}

\def\@frominfo{%
  This Isme \\
  MyStreet 99 \\
  9999 ZZ MyCity \\
  Phone 0123456789}

\newbox\frombox
\newbox\logobox
\newbox\tobox
\def\makeletterhead{%
  \setbox\frombox=\vtop{\@frominfo}%
  \setbox\logobox=\hbox{\includegraphics[width=1in]{logo}}%
  \setbox\tobox=\vtop{\@toname \\\@toaddress}%
  \unitlength=1bp
  \begin{picture}(300,127)
    \put(280,192){\makebox(0,0)[bl]{\smash{\box\logobox}}}
    \put(280,180){\makebox(0,0)[bl]{\smash{\box\frombox}}}
    \put(0,180){\makebox(0,0)[bl]{\smash{\box\tobox}}}
  \end{picture}}
```

Notes

1. Scribus, to be precise, Linux' answer to QuarkXPress and InDesign. Url: `http://ahnews.music.salford.ac.uk/scribus/`
2. Available from `http://tex.aanhet.net/utils/`. It is written in Perl/Tk.
3. A more complete discussion can be found in Victor Eijkhout's *TEX by topic* book, which can be downloaded for free from `http://www.eijkhout.net/tbt/`

Siep Kroonenberg
siepo@cybercomm.nl

Boekdrukken en valkuilen

Keywords
drukken, kopiëren, papierrichting, papierdikte, raster, drukkwaliteit

Abstract
Een boek zetten, dat is nog tot daaraan toe, maar het dan ook nog netjes gedrukt krijgen... ik neem u graag mee langs de valkuilen, tot lering en leedvermaak.

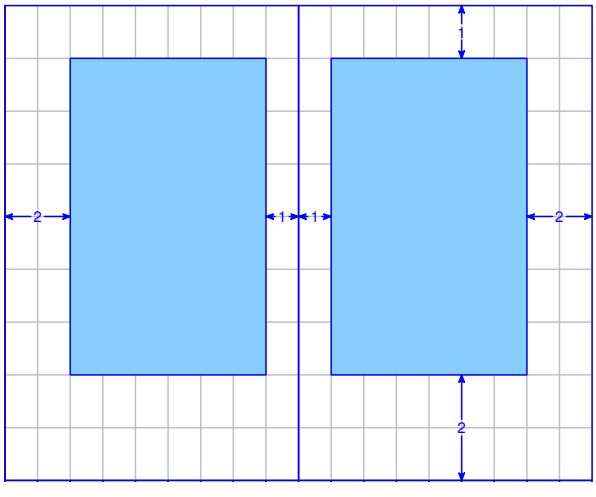
Het schrijven en zetten

Mijn echtgenote besloot twee jaar geleden een biografie te schrijven over haar grootmoeder. Het lag voor de hand dat ik het resultaat in LaTeX zou zetten. Ik leerde haar de meest elementaire commando's van EMACS en LaTeX en zelf maakte ik de opzet voor het boek en vulde ik de hoofdstukken, wanneer ze gereed kwamen, aan met de foto's die ze ervoor had uitgekozen. Het werd een boek van 225 pagina's met 80 zwart-wit foto's.

Opmaak

Na lezing van Willi Eggers' verhaal over de bladspiegel in de vorige Maps kon ik natuurlijk weinig anders dan de gulden snede een kans geven en de binnen- en buitenmarges, zowel als de onder- en bovenmarges een verhouding 1:2 geven (zie figuur 1). Ik gebruikte

Figuur 1. opmaak



Figuur 2. zetspiegel met snijlijnen op A4

daarvoor de MEMOIR class. Ik stelde de papiergrootte op A4 in en voorzag de pdf-uitvoer met behulp van de MEMOIR optie `showtrims` van snijlijnen op een formaat van 160x259 millimeter (figuur 2).

Daarbij heb ik voor 5 millimeter extra rugwit gezorgd, die bij het binden in de rug mag verdwijnen zonder dat daarna in het opengeslagen boek het middenwit smaller wordt dan het snijwit.

Foto's

De foto's werden ingescand op 1200 dpi, genormaliseerd en geconverteerd naar pgm (portable gray map) format en tenslotte interactief uitgesneden, gammagecorrigeerd en opgeslagen in jpeg-formaat. Dit resulteerde in 265 megabyte aan foto's en daardoor tot trage compilatie en weergave. Daarom werden ook lage-resolutie-foto's gemaakt en kreeg de LaTeX brontekst een schakelaar om tussen hoge- en lage-resolutiefoto's te kiezen.

Omslag

Tenslotte ontwierp ik een omslag, rekening houdend met een rugdikte, geschat met behulp van vergelijkbare boeken in mijn boekenkast, van 14 millimeter en een snijoverschot van 5 millimeter (figuur 3).

Zoals u de omslag hier ziet is de kleur ongeveer zoals we die wilden hebben, maar aangezien de bedoeling een twee-kleurendruk was ging in werkelijkheid

Figuur 3. omslagontwerp

Figuur 4. de letter s, gedrukt (links) en op de kleurenlaserprinter(rechts)

een versie in magenta en zwart naar de drukker: het schijnt gemakkelijker te zijn basiskleuren te vervangen dan mengkleuren. Voor de juiste kleur ging een stukje papier mee dat we uit de omslag van een van de boeken uit de boekenkast geknipt hadden. De omslag werd verder zowel met als zonder snij- en vouwlijnen aangeleverd.

Drukken of kopiëren?

Het maken van boeken in kleine oplagen door amateur-schrijvers is tegenwoordig, vooral door de beschikbaarheid van tekstverwerkers, erg in. Men hoort dan ook veel van bedrijven en bedrijfjes die zich bezighouden met het printen en binden van die kleine oplagen, soms in combinatie met advies en begeleiding bij het schrijven en opmaken.

Na het lezen van een enthousiaste reportage in de NRC en positieve geluiden uit de kennissenkring kwamen we daardoor uit bij een bedrijf dat bijzonder klantvriendelijk en attent overkwam. De cdrom met alle gegevens werd opgestuurd, begeleid door een brief die wees op de aangebrachte snijlijnen en de reactie daarop was: geen enkel probleem, dat kunnen wij maken. En de aangeleverde PDF's, konden ze daarmee overweg, ook voor de omslag? Ja hoor, dat ziet er allemaal prima uit.

De prijs, voor 100 exemplaren zou €2000 bedragen, inclusief begeleiding, hoewel we die niet nodig meenden te hebben. Wel duur, maar na twee jaar schrijven en ploeteren kijk je niet op een dubbeltje…

Maar het leek ons verstandig toch even een bezoekje af te leggen om details als kleur, bindmethode, papier en dergelijke te bespreken en voorbeelden te zien van wat men eerder geproduceerd had.

Na twee uur treinen en bussen en een plezierige ontvangst kwam al snel de aap uit de mouw: er was nog één probleempje, het formaat van het boek, 259x160 millimeter, was een beetje onhandig, of ik dat even tot A5 wilde terugbrengen? Men drukte nu eenmaal alles op A5… Wij waren dus snel uitgepraat — het was een beetje jammer van al dat reizen, maar gelukkig was er in die stad een interessant museum.

LES 1: vraag altijd expliciet of het gebruikte boekformaat geen probleem is.

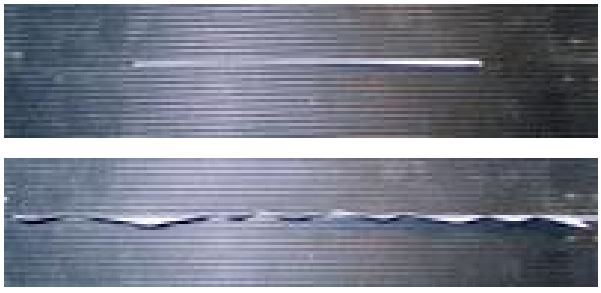
Daarop besloten wij ons eerst maar eens tot een echte drukker te wenden en vroegen per e-mail een prijsschatting bij de Nederlandse drukker van een bekend TEX-tijdschrift waarvan ons de uitvoering, afgezien van het formaat, bijzonder geschikt voor het boek leek en bij de Duitse drukker van een ander TEX-werkje dat ook netjes gedrukt was.

Beide drukkers antwoordden niet. Een telefoontje naar de Nederlandse drukker leerde: "oh, dat e-mailadres op ons briefpapier? Dat wordt nooit gelezen", alsof het de normaalste zaak van de wereld was.

LES 2: laat een e-mail aan een drukker volgen door een telefoontje om te vertellen dat er een e-mail klaarstaat.

De Nederlandse drukker, die toch bekend staat als drukker van kleine oplagen, meldde vervolgens per e-mail:

> Bij het door ons toegepaste drukproces spelen bij kleine oplagen de beginkosten/opstartkosten een



Figuur 5. zijaanzicht van de natte kop- (boven) en rugzijde (onder) van het papier van het eerste proefexemplaar

Figuur 6. front zoals ontworpen (links) en na drukken (rechts)

> dermate grote rol dat wij bij oplagen beneden de plusminus 250 exemplaren geen concurrerende prijzen meer kunnen afgeven. Wij adviseren u om de aanvraag te richten aan ons zusterbedrijf…

Nu laat men zich een financieel voordeel natuurlijk bij voorkeur niet ontgaan, dus wij namen inderdaad contact op met dat zusterbedrijf, dat ons, zoals hierna zal blijken, maandenlang bezig zou gaan houden. Dit bedrijf bleek een kopieerinrichting te zijn, die de gevraagde 100 exemplaren voor €675 wilde produceren.

LES 3: als geld niet erg belangrijk is, laat u dan niet te snel door een drukker naar de kopieerder verwijzen, maar eis een offerte.

Wat er mis ging

De kopieerinrichting stuurde ons, als kennismaking, een daar gedrukt exemplaar op van een tijdschrift, dat er qua papier, foto's en binding prima uitzag, maar de tekst was aanzienlijk minder geslaagd. Deze bleek, net als figuren en foto's, geproduceerd met een kleurenlaserprinter, met als gevolg dat serifs en dunne lijntjes slecht ingevuld waren. Dit is duidelijk te zien in twee foto's die ik met een digitale camera en door een microscoop maakte van een gedrukte letter s (figuur 4, links) en van een dito letter s in het met de laserprinter gemaakte tijdschrift (zelfde figuur, rechts. Na een aanmerking daarop werd ons verzekerd dat het boek niet op de kleurenlaserprinter gedrukt zou worden, maar op een zwart-wit-printer. Waarmee dit voorbeeld-exemplaar dus minder zinvol werd.

Na verloop van tijd ontvingen we een niet-gebonden eerste exemplaar als proefdruk, met een losse, ongevouwen omslag daarbij. Inderdaad zag de tekst er zeer fraai uit en was nauwelijks van echt drukwerk te onderscheiden. Ook de foto's kwamen veel beter over dan gehoopt, op één foto na, die een interferentiepatroon vertoonde als gevolg van het feit dat die foto oorspronkelijk gerasterd was en nu, met een andere rasterfrequentie, opnieuw gerasterd was. Maar het effect was zo klein dat we dat maar zo lieten.

Er waren nog twee problemen die we opgelost wilden hebben:

1. Het olijfgroen van de omslag was te donker uitgevallen; ik stuurde een e-mail met een plaatje van twee kleurvlakken daarin, de een met de kleur die ik zag en de ander met de kleur die ik wilde hebben. Ik verzocht de kleur ongeveer zo aan te passen, wat me werd toegezegd.

2. Het papier lag in de verkeerde richting: papier hoort zo bedrukt te worden, dat het, wanneer het vochtig wordt, aan de kopzijde gaat vervormen, terwijl de rugzijde recht hoort te blijven. Zo voorkomt men dat het gemakkelijk uit de binding losraakt. Hier was dat juist andersom. Men kan dat eenvoudig constateren door met een vochtige spons een paar centimeter van de kant van het papier aan beide zijden nat te maken. Figuur 5 laat het resultaat van dit experiment zien. Ook correctie van dit probleem werd door de kopieerinrichting toegezegd.

Na de gedane toezeggingen besloten wij toestemming te geven om tot drukken over te gaan zonder eerst nog een nieuw proefexemplaar te zien.

Eerste druk

De eerste druk van honderd exemplaren werd bezorgd… en bleek een grote teleurstelling:

□ Het olijfgroen van de omslag bleek ongewijzigd donker: door gebrekkige interne communicatie was men vergeten die aanpassing te maken.

Figuur 7. titel voor (boven) en na expansie

□ De papierrichting was onveranderd fout. Een verklaring daarvoor werd niet verkregen.

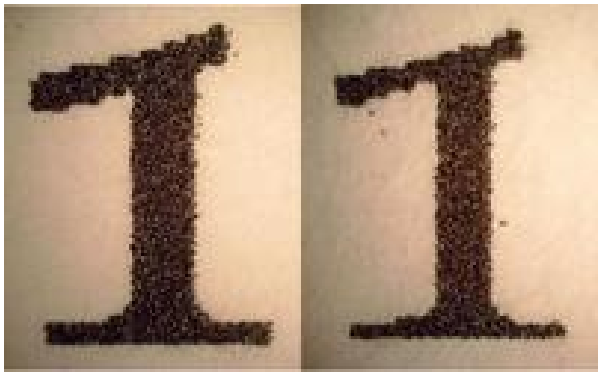
□ Erger was dat men nog eens naar het omslag van het proefexemplaar had gekeken en had bedacht: toch wel jammer dat die tekst langs de rug op de voorzijde van het boek zo dicht tegen de rugzijde ligt, dat de pers-ril (op 5 millimeter van de rug, die in het ongebonden proefexemplaar natuurlijk niet te zien was) daar dwars doorheen loopt. De oplossing daarvoor had men gevonden door die tekst tot 75% te verkleinen. In figuur 6 zijn het ontwerp van het front (links) en het gedrukte resultaat vergeleken.
Men had het wit tussen de woorden van de titel nog een beetje aangepast, omdat die anders wel erg kort zou zijn geworden. Alleen: doordat het handwerk was, was dat nogal onregelmatig gebeurd, zoals is te zien in figuur 7, waar het ontwerp (boven) en het resultaat (onder) op gelijke breedten zijn geschaald.

□ De klap op de vuurpijl was, dat men ook nog eens goed naar de zetspiegel van het proefexemplaar had gekeken en toen zag dat de tekst niet netjes gecentreerd op de pagina stond — die was verschoven naar de rug en de kop. Maar daar wist men wat op: men had geschikte tools in huis om de teksten netjes te centreren. Zo was de zo zorgvuldig aangebrachte 1:6:9-verdeling dus deskundig om zeep geholpen!

LES 4: maak expliciet duidelijk dat aan het ontwerp niet zonder overleg gesleuteld mag worden.

LES 5: maak bij goedkeuring van een proefexemplaar duidelijk dat daarna geen veranderingen meer mogen worden aangebracht.

Figuur 8. een 1 in de eerste (links) en in de tweede druk



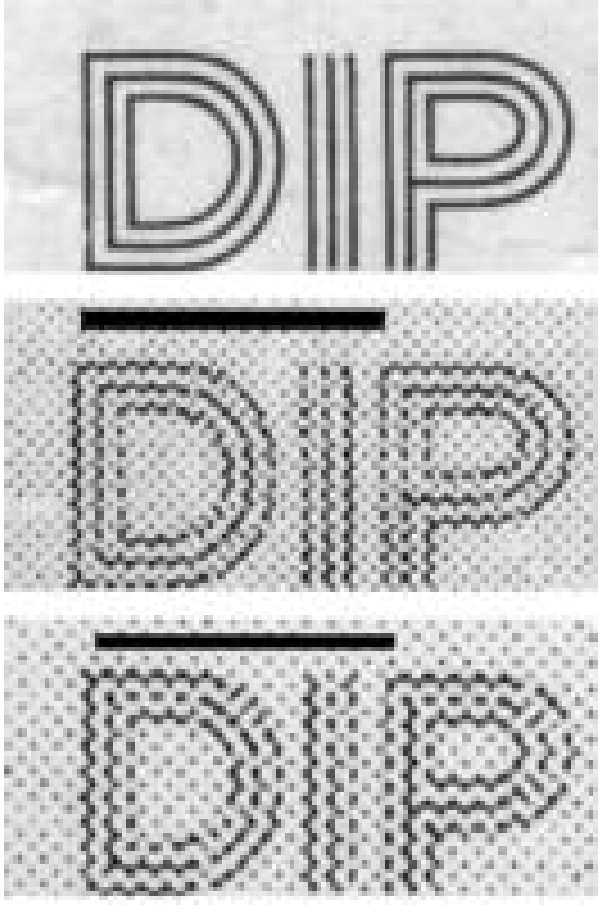
Er ging een e-mail op poten naar de kopieerinrichting. En ja, men gaf me volledig gelijk, het was misgegaan, had nooit mogen gebeuren, het zou worden overgedaan.

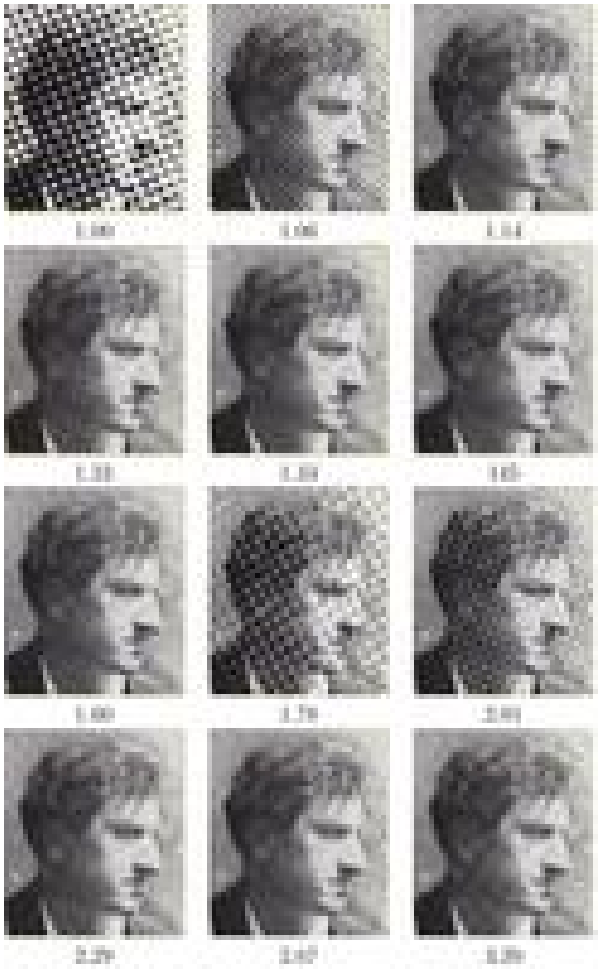
Tweede druk

Voor de tweede druk heb ik het omslagontwerp aangepast, door op het voorblad een verticale strook van 5 millimeter langs de rug in te voegen om te voorkomen dat de titeltekst onder de pers-ril terecht zou komen. De omslagkleur werd met behulp van teststroken aangepast en de (wederom losbladige) proefdruk zag er, voor wat betreft omslag en zetspiegel, goed uit.

Tot mijn onaangename verrassing echter bleek nu de kwaliteit van zowel de tekst als de foto's aanmerkelijk minder dan bij de eerste druk. Daardoor maakte het tekstwerk een schriele indruk en in de foto's waren details verloren gegaan. Daarbij speelt mijn sterke bijziendheid mij waarschijnlijk parten, want mijn echtgenote zag het in eerste instantie niet en de kopieer-

Figuur 9. detail in een foto; van boven naar beneden: de scan, de eerste druk, de tweede druk

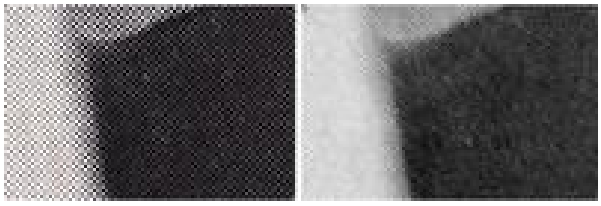




Figuur 10. interferentieverschijnselen in een gerasterde foto

inrichting kon er zich weinig bij voorstellen. Daarom werden microscoop en digitale camera maar weer eens te hulp geroepen.

Figuur 8 laat zien dat een cijfer 1 in de eerste druk (links) er aanmerkelijk robuuster uitziet dan in de tweede druk (rechts). Figuur 9 laat, van een detail, van boven naar beneden de oorspronkelijke scan, het resultaat in de eerste druk en dat in de tweede druk zien. De twee zwarte balkjes werden ingevoegd als referentie om de raster-frequenties te kunnen tellen; daaraan is te zien dat de resolutie in de tweede druk 20% lager was dan in de eerste druk. De kopieerinrichting hield het erop dat kopieerwerk nu eenmaal minder reproduceerbaar was dan drukwerk, en dat het een kwestie van temperatuur, vochtigheid, tonerniveau of iets dergelijks moest zijn. Het bleek niet mogelijk de oorspronkelijk kwaliteit opnieuw te bereiken.



Figuur 11. verschil tussen een 1200 dpi scan en een 5 megapixel camera-kopie

Een bijkomend probleem van de verandering in resolutie bleek te zijn dat de hiervoor al genoemde interferentieverschijnselen in een van de foto's veel ernstiger was geworden. Dat effect van de afdruk-resolutie bleek mooi gedemonstreerd te kunnen worden door die foto met behulp van het GIMP-programma in verschillende vergrotingen op het beeldscherm te zetten. Figuur 10 laat het resultaat van die vergrotingen, na terugschaling tot een constante afmeting, voor een oplopende reeks zien.

Het probleem werd opgelost door de betreffende foto te kopiëren met een digitale camera die, door een lagere resolutie en een minder nauwkeurige registratie, de afzonderlijke rasterpunten in de foto minder goed ziet. Figuur 11 laat het verschil tussen die twee zien in een detail van die foto.

Hoewel dus zoals gezegd de kwaliteit van tekst en foto's minder was dan eerder mogelijk was gebleken, besloten we toch maar daar genoegen mee te nemen. De tweede druk werd afgeleverd…

Derde druk

Maar helaas! Weliswaar zag alles er nu uit als verwacht, maar toen we de boeken begonnen uit te delen aan de doelgroep viel al meteen op dat iedereen zijn boek slechts half opende en weliswaar enthousiast, maar toch wel erg moeizaam tussen de bladzijden begon te turen. Het probleem bleek dat er nogal wat kracht nodig was om het boek te openen en dat bij de doorzetters bleek, dat wanneer het echt met kracht op tafel werd uitgespreid, de bladzijden loslieten! Dit was bij de eerste druk niet het geval geweest, hoewel daarvoor dezelfde bindmethode (gebrocheerd) en hetzelfde papier was gebruikt.

LES 6: vraag een *volledig afgewerkt* proefexemplaar.

Wederom een e-mail naar de kopieerinrichting, de tweede druk werd teruggehaald, er werd geëxperimenteerd en men moest, ook daar, tot de conclusie komen dat de boeken bij normaal leesgedrag tot losbladigheid vervielen. Dit was ook het moment waar bleek dat de grote ervaring die men met boeken beweerde

te hebben losbladige boeken betrof die in ringbanden werden geleverd.

LES 7: vraag voorbeelden van soortgelijke eerder geleverde boeken.

Besloten werd nog één poging te wagen waarbij het gebruikte 115-grams papier vervangen zou worden door 90-grams papier. Het resultaat was een boek dat nog steeds moeilijk te openen was, maar tenminste niet snel losliet. Wij hebben er intussen, na vijf maanden, zo genoeg van dat we maar besloten hebben het hierbij te laten.

Conclusies

- ☐ ga naar een echte drukker als geld niet heel belangrijk is
- ☐ gaat u toch naar een kopieerinrichting, maak u dan ook niet druk om de layout — het is de moeite niet waard
- ☐ trek u bij twijfel in een vroeg stadium terug — naarmate het aantal proefexemplaren toeneemt wordt dat moeilijker
- ☐ onderhoud telefonisch contact — e-mail wordt niet gelezen, of in ieder geval meestal niet beantwoord
- ☐ verifieer van tevoren dat:
  - het geplande boekformaat geen probleem is
  - voorbeelden van soortgelijke boeken getoond kunnen worden
- ☐ maak van tevoren schriftelijk duidelijk dat:
  - het boek zonder druk open moet kunnen
  - zonder overleg niet aan het ontwerp gesleuteld mag worden
  - proefexemplaren volledig afgewerkt moeten zijn
  - na goedkeuring van het proefexemplaar geen wijzigingen meer mogen worden aangebracht

Wybo Dekker
wybo@servalys.nl

Object-Oriented Graphics with MetaObj

Abstract
MetaOBJ is a macro package for MetaPost [1], a programming language for graphics producing PostScript output, based on the well-known MetaFont. MetaOBJ is written and maintained by Denis B. Roegel. It has been released under the LPPL (LaTeX Project Public License) and is available from CTAN.
The cool thing about MetaOBJ is that it provides very high-level object-oriented macros which simplify the construction of complicated drawings by defining objects of arbitrary complexity and combining them to larger structures. This is already reflected in the name of the package: MetaOBJ is a shortcut for "MetaPost Objects". The cover picture of the MetaOBJ manual [2], reproduced in figure 1, gives a first impression of the kind of graphics which can be produced.

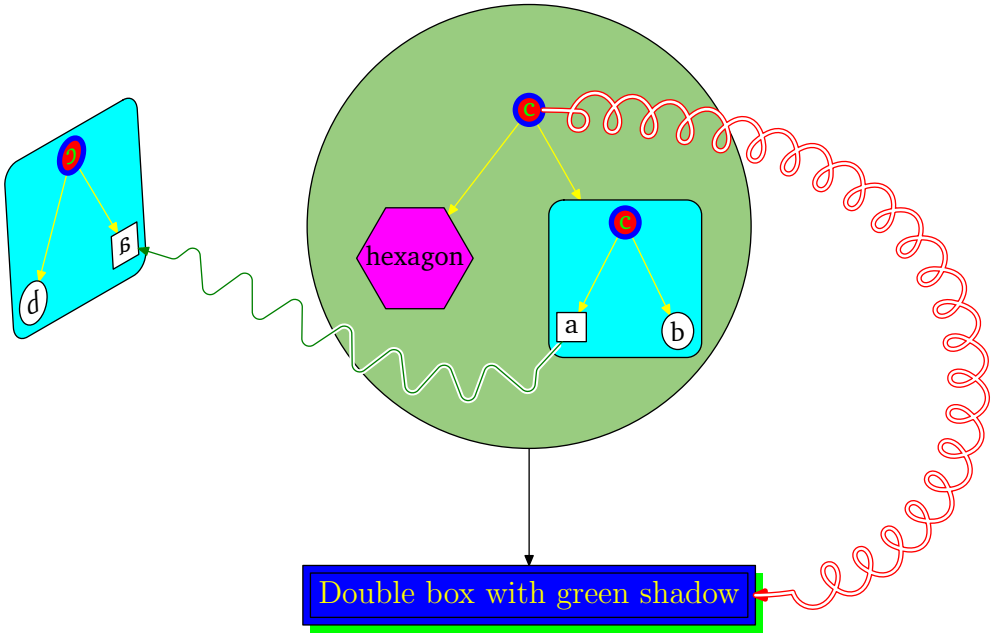


Figure 1 Title graphic of the MetaOBJ manual, taken from [2].

Introduction

It is not the aim of this paper to cover all possibilities and intricate details of MetaOBJ. Although many examples and topics are borrowed from the excellent user manual [2], the following cannot be and is not intended as a replacement. I will rather try to portray MetaOBJ from my point of view – the point of view of a user interested in getting qualitatively excellent graphics done relatively easily. So, let me start by saying some words about my TeX-background.

I would characterize myself as an experienced, but not expert user of TeX. During my studies, I used LaTeX for nearly all kinds of documents I had to produce. For

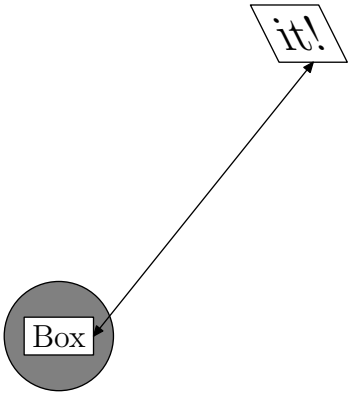
graphics, I used either PSTRICKS or external software (which is of course annoying because of difficulties concerning consistency and exactness). About three years ago, I made first contact with ConTEXt and was immediately mesmerized by its possibilities, the rapid development and future potential. And of course, I came to know MetaPost this way. Since the few graphics I have to produce today happen to be of a rather schematical and technical kind, there can presumably be no better and more satisfactory way to create them than by using a graphic programming language like MetaPost. Therefore, the system has become my favored choice.

When I first saw MetaOBJ, I started to use it because it provides features with similar output to many PSTRICKS commands, works well with ConTEXt's MetaPost integration, and does not hamper direct PDF output. Anybody plunging deeper into MetaOBJ will notice that this does not do full justice to its true strengths and intentions, but, as stated above, in the same way my intention is to give an impression of what is available to the user easily, without focussing too much on object-orientation, implementation or extensions.

After some small examples to quicken the reader's appetite, I am going to show how objects can be combined into more complex graphics like the MetaOBJ cover graphic shown in figure 1 and how some PSTRICKS-like graphics can be produced. Finally, a custom class of objects will be defined to produce schematic representations of certain molecules.

Example Graphics

Let us start right now with a first MetaOBJ graphic. The code is virtually self-explanatory. But as simple as it is, figure 2 already shows a lot of MetaOBJ features:

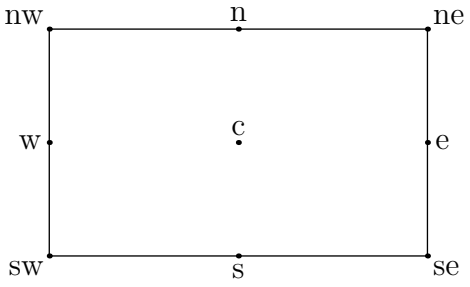



```
newBox.box1(btex Box etex);
newCircle.circ(box1) "circmargin(2mm)",
  "filled(true)", "fillcolor(.5white)";
newBox.box2(btex it! etex);
scaleObj(box2, 3/2);
slantObj(box2, -1/2);
box2.w - circ.e = (2cm, 4cm);
box1.c = origin;
drawObj(circ, box2);
drawdblarrow box1.e -- box2.s;
```

Figure 2 A first example.

- Objects are created as instances of classes by `newSomething` constructors. In the example, we have objects of classes `Box` and `Circle`.
- Objects have names by which they can be accessed: `box1`, `box2` and `circ` in the example.
- Most objects of the MetaOBJ standard library can be customized by the use of options like `filled` or `fillcolor`.
- Objects accept and store all kind transformations supported by MetaPost: `scaleObj`, `slantObj` and so on with self-explanatory names.
- Objects can contain pictures (and more, like paths and other variables, as we will see later). Most importantly, they can contain other objects: `box1` is a subobject of `circ`.
- Objects are floating, relative positioning to each other is possible: `box2.w - circ.e = (2cm, 4cm)`. Thus, objects can be only partly defined.

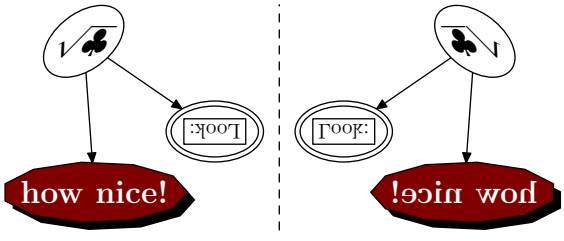
□ Standard MetaOBJ objects provide a standard interface. They have a set of points in the directions of the compass, as shown in figure 3. The interface for accessing the points and setting the position of the object is compatible to the the `boxes.mp` package by John D. Hobby.




```
show_empty_boxes := true;
newEmptyBox.e(5cm, 3cm)
  "framed(true)";
e.c = origin;
drawObj(e);
```

Figure 3 Points of the MetaOBJ standard interface.

□ Specifying one point of an object defines the position of all points of the object (at least for the standard objects): `box1.c = origin`. As we will see, objects store equations determining the relative positions of their points.
□ Once absolutely positioned, they can be drawn with a call to `drawObj`, which internally calls the respective drawing function of the class, i. e. `drawCircle` or `drawBox`.
□ Finally, the line `drawdblarrow box1.e - box2.s` shows that making an object a subobject of another object does not hamper us from accessing it directly. This is true for any depth of nesting and is an important advantage compared to approaches which do not preserve the object structure, e. g. by storing just pictures inside.




```
newDEllipse.ell(new_Box_(btex Look: etex)());
save t;
rotateObj(ell, 180);
scaleObj(ell, 2/3);
newPolygon.pol(btex \bf how nice! etex, 11)
  "filled(true)", "fillcolor(.5red)",
  "picturecolor(white)", "shadow(true)";
newCircle.root(btex $\sqrt\clubsuit$ etex);
slantObj(root, 1/2);
newTree.t(root)(pol, ell);
rebindVisibleObj(t);
t.c = origin;
drawObj(t);
duplicateObj(t2, t);
reflectObj(t2)(origin, (0,1));
draw t.se + (2mm, 0) -- t.ne + (2mm, 0)
  dashed evenly;
t2.e = t.e + (4mm, 0);
drawObj(t2);
```

Figure 4 Another example graphic.

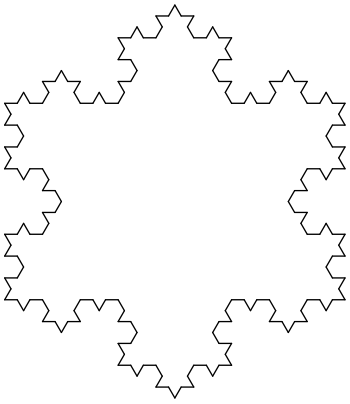
In addition to the above, the second example (figure 4) shows some more points worth mentioning:

□ There are *streamlined* versions of the standard constructors, returning an object of the respective type. They can be used to pass the new object to another object immediately, like the `Box` in `newDEllipse.ell(new_Box_(btex`

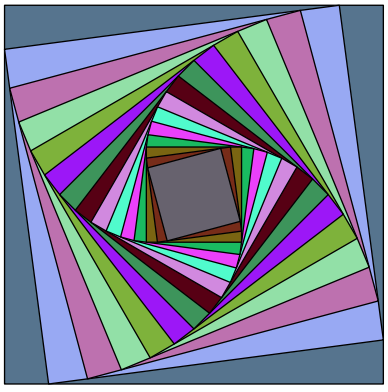
`Look: etex)())`, or to assign the object to a variable. Streamlined constructors are distinguished from the normal ones by underscores as in `new_Box` and have a slightly different syntax.

- An in-depth, identical yet independent clone can be created by `duplicateObj`.
- Classes for regular arrangement of objects can be defined, such as the `Tree` and `Matrix` classes of the standard library.

There are of course many more features which cannot all be commented here. Two last examples in figure 5 show that it is also possible to define recursive objects. `VonKochFlake` and `RecursiveBox` are part of the standard MetaOBJ library.



```
newVonKochFlake.flake(3);
scaleObj(flake,.45);
flake.c=origin;
drawObj(flake);
```



```
newRecursiveBox.rb(14)
  "rotangle(7.5)";
randomizeRecursiveBox(rb);
scaleObj(rb, .2);
rb.c = origin;
drawObj(rb);
```

Figure 5 `VonKochFlake` and `RecursiveBox`

Actually, I have cheated a little bit, since the following code has also been used to produce the colorful `RecursiveBox`.

```
def randomcolor =
     (uniformdeviate 1, uniformdeviate 1, uniformdeviate 1)
  enddef;

  vardef randomizeRecursiveBox(suffix n) =
    if known n.sub :
      randomizeRecursiveBox(obj(n.sub));
    fi;
    ExecuteOptions(n)("fillcolor(randomcolor)");
  enddef;

  setObjectDefaultOption("RecursiveBox")("filled")(true);
```

Connections and Graphs

The previous examples have already shown many of the basic classes of the MetaOBJ library, and how these can be combined to composite objects. A common need that has not yet been addressed is how to draw connections between objects. And, what about that promised PSTRICKS functionality? Indeed, the commands for connecting objects in MetaOBJ are very similar to those from PSTRICKS. Figure 6 summarizes

some of the possibilities. As you can see, the connections can operate on objects as in `ncline(A)(B)` as well as on points, which means that you can write something like `nccurve(A)(origin)`.

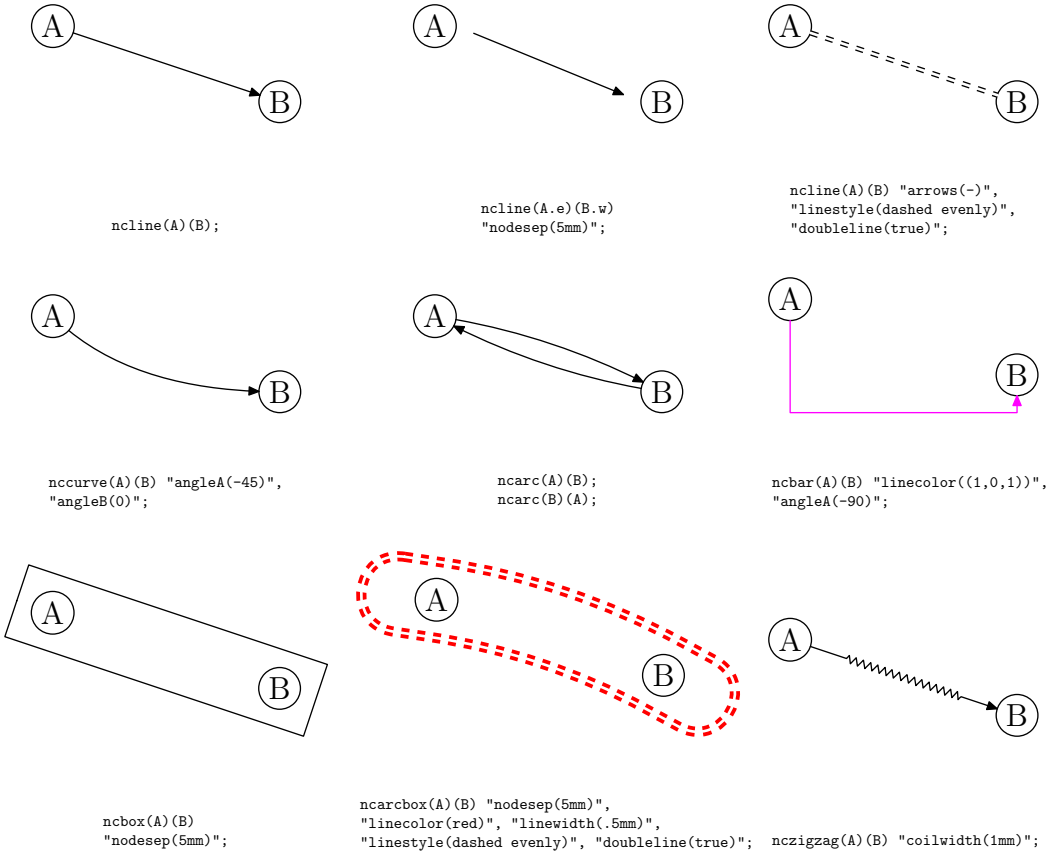


Figure 6 Some of the node connections defined by MetaOBJ. For a complete overview of types and options, please refer to the manual.

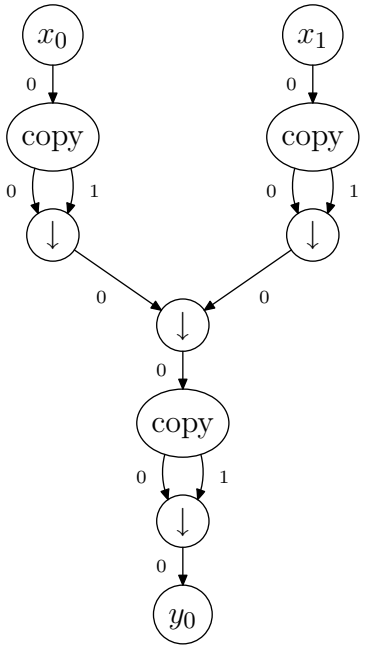
Several options trigger the appearance of the connections. It should also be mentioned that the examples show only one type of connections: the *immediate* ones, which are drawn instantly, meaning that the object positions have to be already defined. MetaOBJ also provides *deferred* versions of all connections, which are memorized with an object and are drawn when the object is drawn. The commands are identical except for the object name as a suffix: `ncline.A(A)(B)` stores the connection between objects `A` and `B` together with object `A`.

The MetaOBJ manual title graphic (see figure 1) uses nearly everything mentioned so far and a little bit more, so it may be a good time to reveal its source. The code should be easily understandable by now, and hopefully, you will be delighted to see how a relatively complicated graphic can be programmed in such an intuitive and simple manner.

```
newBox.a("a");
  newEllipse.b("b");
  newEllipse.c("c") "filled(true)", "fillcolor(red)","picturecolor(green)",
                    "framecolor(blue)","framewidth(2pt)";
  newTree.t(c)(a,b) "linecolor((1,1,0))";
  newBox.aa(t) "filled(true)", "fillcolor((0,1,1))","rbox_radius(2mm)";
  aa.c=origin;
```

```
newHexagon.xa("hexagon") "fit(false)","filled(true)","fillcolor((1,0,1))";
newEllipse.xc("c") "filled(true)", "fillcolor(red)","picturecolor(green)",
                   "framecolor(blue)","framewidth(2pt)";
newTree.xt(xc)(xa,aa) "linecolor((1,1,0))";
newCircle.xaa(xt) "filled(true)", "fillcolor((.6,.8,.5))";
newDBox.db(btex Double box with green shadow etex)
        "shadow(true)", "shadowcolor(green)",
        "filled(true)","fillcolor(blue)","picturecolor((1,1,0))";
newTree.nt(xaa)(db);
drawObj(nt);
nccoil(xc)(db) "angleA(0)","angleB(180)",
  "coilwidth(5mm)","linetension(0.8)","linecolor(red)",
  "doubleline(true)","posB(e)";
duplicateObj(dt,aa);
reflectObj(dt,origin,up);
slantObj(dt,.5);
rotateObj(dt,30);
dt.c=nt.c-(6cm,-1cm);
drawObj(dt);
nczigzag(a)(treepos(obj(dt.sub))(1))
  "angleA(-120)","coilwidth(7mm)","linecolor(.5green)","linearc(1mm)",
  "border(2pt)";
```

Apart from trees, which have been used in the examples so far, matrices are another way of arranging objects in a regular way provided by MetaOBJ (as by PSTRICKS). For simplification of connections between tree nodes and matrix elements, there are special connection commands which refer to the connected elements by their coordinates in the container object. The example in figure 7 shows how this is done for a matrix. In addition, you can see that labels can be added to connections (and to all other objects) easily.




```
newCircle.xnull(btex $x_0$\strut etex);
newCircle.xeins(btex $x_1$\strut etex);
newCircle.ynull(btex $y_0$\strut etex);
for i=1 upto 4 :
  newCircle.pfeil[i](btex $\downarrow$\strut etex);
  newEllipse.copy[i](btex copy\strut etex);
endfor
newMatrix.mat(7,3)(
  xnull,nb,xeins, copy1,nb,copy2, pfeil1,nb,pfeil2,
  nb,pfeil3,nb,   nb,copy3,nb,    nb,pfeil4,nb,
  nb,ynull,nb) "hsep(5mm)", "vsep(5mm)";
mcline.Obj(mat)(1,1,2,1) "name(a)";
ObjLabel.Obj(mat)(btex $\scriptscriptstyle0$ etex)
  "labpathname(a)", "labdir(lft)";

% ... (other labels, not shown here)

mcline.Obj(mat)(6,2,7,2) "name(l)";
ObjLabel.Obj(mat)(btex $\scriptscriptstyle0$ etex)
  "labpathname(l)", "labdir(lft)";
mat.c=origin;
drawObj(mat);
```

Figure 7 Example graph showing the use of matrices, matrix connections and labels.

Custom Objects

At some point, you will probably wish to create your own classes. In principle, all you need to define for a `Nice` class is a constructor `newNice`, a drawing function `drawNice` and a bounding path `BpathNice`. MetaOBJ helps you with many predefined standard components and functions.

Before creating new objects, let us first look at a simple standard object, `EmptyBox`. Here is the constructor:

```
vardef newEmptyBox@#(expr dx,dy) text options=
     ExecuteOptions(@#)(options);
     assignObj(@#,"EmptyBox");
     StandardInterface;
     ObjCode StandardEquations,
       "@#ise-@#isw=(" & decimal dx & ",0)",
       "@#ine-@#ise=(0," & decimal dy & ")";
   enddef;
```

The constructor takes two dimensions as arguments – used as width and height of the box – as well as a potentially empty list of options. These are processed by `ExecuteOptions`. `assignObj` takes care of making the new object a member of the correct class (and some more internal things). Then, `StandardInterface` declares the standard points shown in figure 3 as well as, unmentioned before, a second set of these points (`ine`, `in`, `ine`, and so on) for use from within the object (don't bother about the reasons for now). The equations determining the relations between the object points are stored as strings using `ObjCode`, where `StandardEquations` is an abbreviation for `(PureStandardEquations & StandardInnerEquations)`, which in turn are defined as

```
def PureStandardEquations=
     ("@#se-@#sw=@#ne-@#nw;" & % parallelogram equation
     "xpart(@#se-@#ne)=0;" &
     "ypart(@#se-@#sw)=0;" &
     "@#n=.5[@#ne,@#nw];" &    % North
     "@#s=.5[@#se,@#sw];" &    % South
     "@#e=.5[@#ne,@#se];" &    % East
     "@#w=.5[@#nw,@#sw];" &    % West
     "@#c=.5[@#n,@#s];" )      % Center
   enddef;

   def StandardInnerEquations=
      ("@#ine=@#ne;@#inw=@#nw;@#isw=@#sw;@#ise=@#se;@#in=@#n;@#is=@#s;" &
     "@#ie=@#e;@#iw=@#w;@#ic=@#c;")
   enddef;
```

Drawing an `EmptyBox` is simple – it is shown only if enabled (and you won't see much unless it is filled or framed). In any case, paths stored with the object are drawn:

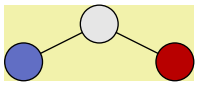
```
def drawEmptyBox(suffix n)=
     if show_empty_boxes:
       drawFramedOrFilledObject_(n);
     fi;
     drawMemorizedPaths_(n);
   enddef;
```

The bounding path (used e. g. for filling) is also very simple:

```
def BpathEmptyBox(suffix n)=StandardBpath(n) enddef;
```

an abbreviation for `(n.inw-n.isw-n.ise-n.ine-cycle)`. And that's it. Of course, the definitions get more complicated for more sophisticated objects, but the principle remains the same.

Now we have everything to go ahead and create our own classes. I once came across the need to draw a schematic representation for a cyclic peptide. Maybe you know that peptide molecules are polymers, composed of monomers called amino acids. Anyway, I wanted to represent each amino acid as three circles (the main chain atoms) connected by lines (covalent bonds), with a color box in the background, as shown in figure 8.



```
newAminoAcid.X;
X.c=origin;
drawObj(X);
```

Figure 8 Schematic representation of an amino acid.

Admittedly, the code is not very nice, and of course I would do it differently if I did it again. But due to lack of time for a rewrite, here it is. After some setup:

```
% ad : atom circle diameter
  % dx : x distance between two atom centers
  % dy : y distance

  numeric ad, dx, dy, hyp, offset;
    ad = 5mm;
    dx = 2ad;
    dy = 1/2dx;
    hyp = sqrt(dx**2+dy**2);

  % ahlength : length of arrow heads
  ahlength := 3/4ad;

  % colors for atoms N, C, CA and background
  color NColor, CColor, CAColor, AAColor;
    NColor = (.38,.431,.769);
    CAColor = .9white;
    CColor = (.722,0,0);
    AAColor = (.95,.95,.67);
```

we define the constructor for an `AminoAcid`:

```
vardef newAminoAcid@# =
    assignObj(@#,"AminoAcid");
    StandardInterface;
    save N,CA,C; string N,CA,C;
    N=newobjstring_; CA=newobjstring_; C=newobjstring_;
    newCircle.obj(N)(nullpicture) "circmargin(.5ad)",
      "filled(true)","fillcolor(NColor)";
    newCircle.obj(CA)(nullpicture) "circmargin(.5ad)",
      "filled(true)","fillcolor(CAColor)";
    newCircle.obj(C)(nullpicture) "circmargin(.5ad)",
      "filled(true)","fillcolor(CColor)";
    SubObject(N,obj(N)); SubObject(CA,obj(CA)); SubObject(C,obj(C));
    ObjCode StandardEquations,
      "@#isw=obj(@#N).sw",
      "@#ine=(xpart obj(@#C).e,ypart obj(@#CA).n)",
      "obj(@#CA).c-obj(@#N).c=(dx,dy)",
      "obj(@#C).c-obj(@#CA).c=(dx,-dy)";
    StandardTies;
    ncline.@#(obj(@#N))(obj(@#CA)) "arrows(-)";
    ncline.@#(obj(@#CA))(obj(@#C)) "arrows(-)";
  enddef;
```

As you can see, three `Circles` are created for the atoms and registered as subobjects of the `AminoAcid` using the `SubObject` function. `newobjstring_` provides a new

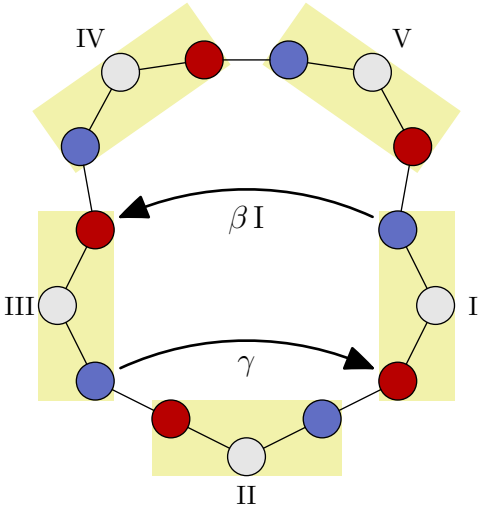
unique name for each subobject. In addition to the `StandardEquations`, there are some relations arranging the subobjects. `StandardTies` memorizes the connection between the object and its subobjects. Finally, the lines representing the bonds are added to the `AminoAcid` object using `ncline`.

```
def BpathAminoAcid(suffix n)=StandardBpath(n) enddef;

  def drawAminoAcid(suffix n)=
    fill BpathAminoAcid(n) withcolor AAColor;
    drawObj(obj(n.N),obj(n.CA),obj(n.C));
    drawMemorizedPaths_(n);
  enddef;
```

The bounding path is not special at all. `drawAminoAcid` is straightforward, too: the background is filled, and the subobjects as well as the stored paths are drawn.

Now, the amino acids had to be combined to a peptide, as shown in figure 9 for a pentapeptide.




```
newPentaPeptide.p;
lowerTurn.p(btex $\gamma$ etex);
upperTurn.p(btex $\beta\,$I etex);
p.c=origin;
drawObj(p);
```

Figure 9 Schematic drawing of pentapeptide.

The `PentaPeptide` class was defined as follows.

```
vardef newPentaPeptide@#=
     assignObj(@#,"PentaPeptide");
     StandardInterface;
     save I,II,III,IV,V; string I,II,III,IV,V;
     forsuffixes s = I, II, III, IV, V :
       s=newobjstring_;
       newAminoAcid.obj(s);
       SubObject(s,obj(s));
     endfor
     rotateObj(obj(IV),35); rotateObj(obj(V),-35); rotateObj(obj(I),270);
     rotateObj(obj(II),180); rotateObj(obj(III),90);
     ObjCode StandardEquations,
        "xpart .5[obj(obj(@#III).C).c,obj(obj(@#I).N).c]=xpart obj(@#II).c",
       "ypart obj(obj(@#III).C).c=ypart obj(obj(@#I).N).c",
        "obj(obj(@#I).C).c-obj(obj(@#II).N).c=(dx,dy)",
       "xpart .5[obj(@#V).c,obj(@#IV).c]=xpart obj(@#II).c",
        "obj(obj(@#V).N).c-obj(obj(@#IV).C).c=(hyp,0)",
       "ypart obj(obj(@#IV).N).c-ypart obj(obj(@#III).C).c="
       &"sqrt(hyp**2-(xpart obj(obj(@#III).C).c-xpart obj(obj(@#IV).N).c)**2)",
        "@#isw=(xpart obj(@#III).n,ypart obj(@#II).n)",
       "@#ine=(xpart obj(@#I).n,ypart obj(@#IV).n)";
```

```
    StandardTies;
  ncline.@#(obj(obj(@#I).C))(obj(obj(@#II).N)) "arrows(-)";
   ncline.@#(obj(obj(@#II).C))(obj(obj(@#III).N)) "arrows(-)";
  ncline.@#(obj(obj(@#III).C))(obj(obj(@#IV).N)) "arrows(-)";
  ncline.@#(obj(obj(@#IV).C))(obj(obj(@#V).N)) "arrows(-)";
  ncline.@#(obj(obj(@#V).C))(obj(obj(@#I).N)) "arrows(-)";
  ObjLabel.obj(@#I)(btex I etex) "labpoint(n)", "labshift((.5ad,0))";
  ObjLabel.obj(@#II)(btex II etex) "labpoint(n)", "labshift((0,-.5ad))";
  ObjLabel.obj(@#III)(btex III etex) "labpoint(n)", "labshift((-.5ad,0))";
  ObjLabel.obj(@#IV)(btex IV etex) "labpoint(n)", "labshift((-.5ad,.5ad))";
  ObjLabel.obj(@#V)(btex V etex) "labpoint(n)", "labshift((.5ad,.5ad))";
enddef;

def drawPentaPeptide(suffix n)=
  drawObj(obj(n.I),obj(n.II),obj(n.III),obj(n.IV),obj(n.V));
  drawMemorizedPaths_(n);
enddef;
```

And finally two convenient abbreviations.

```
vardef lowerTurn@#(expr name)=
    nccurve.@#(obj(obj(@#.III).N))(obj(obj(@#.I).C)) "name(turnlow)",
      "angleA(30)", "angleB(-30)", "nodesepA(.5ad)", "nodesepB(.5ad)",
      "linewidth(2pt)";
    ObjLabel.@#(name) "labpathname(turnlow)", "labdir(bot)";
  enddef;

  vardef upperTurn@#(expr name)=
    nccurve.@#(obj(obj(@#.I).N))(obj(obj(@#.III).C)) "name(turnup)",
      "angleA(150)", "angleB(-150)", "nodesepA(.5ad)", "nodesepB(.5ad)",
      "linewidth(2pt)";
    ObjLabel.@#(name) "labpathname(turnup)", "labdir(bot)";
  enddef;
```

Several wishes remain unaddressed so far, for example parameterization of labels. But this is another story, which may be told another day.

Final Remarks

In the course of writing this article, I had to realize that it is not trivial to point out the distinctive features of MetaOBJ without getting too lengthy or quoting the whole manual. I hope that some of the Maps readers will feel like trying MetaOBJ themselves after reading. In this case, I have achieved my aim. My thanks and all credits for MetaOBJ go to Denis B. Roegel for creating this awesome package and making it publicly available.

References

[1] J. D. Hobby, A User's Manual for MetaPost, *AT&T Bell Laboratories Computing Science Technical Report 162*, **1992**.

[2] D. B. Roegel, The MetaOBJ tutorial and reference manual, **2002**, `http://www.tug.org/tex-archive/graphics/metapost/contrib/macros/metaobj/doc/momanual.pdf`.

Eckhart W. Guthöhrlein
eckhart.guthoehrlein@uni-bielefeld.de

contextgarden.net

Keywords
ConTEXt, documentation, Wiki, website, live, typesetting on demand, texshow, mailinglist archive, source browser

Abstract
The goal of the *contextgarden.net* project is to enhance the documentation of ConTEXt. It consists of several web services that together provide the technical framework behind the documentation. A large (and growing) percentage of the supplied content is actually provided by the visitors of the interconnected web sites.

Introduction

Many users have tried the LaTEX-alternative ConTEXt. But more than a few have given up almost right at the start, simply because they didn't know how to proceed any further. Unnecesarily so, since there is an active user group that is very helpful. And there is a vast amount of information and documentation on ConTEXt readily available. One just has to find it.

What is available?

The official documentation for ConTEXt is available on the web server of PRAGMA ADE.[1] There, you will find many PDF-files that can answer your questions as well as show you some interesting possibilities of typesetting with ConTEXt. There are two distinct ways to access the documentation on the web site.

The first possibility is to use PDF-based navigation. Just select *showcase* on the web site of PRAGMA ADE, and you will get a hierarchical overview of all available PDF-documents. The navigation itself shows some aspects of the vast possibilities of ConTEXt. This makes it plain to see that it will be worth having a closer look at the program, even before you have read the first document!

The second possibility is to is to select *overview* on the web page. This will present you with a rather simple listing of the available files based on their category. You can see all available manuals and some supplementary documentation at a glance.

The novice user should definitely have a look at the beginner's manual *ConTEXt, an excursion* and, after that, *ConTEXt, the manual*.

Besides these two important documents, there are different sets of manuals:

manuals This is the most important set of documentation files. Besides the beginner's manual and the main manual, there are manuals dealing with XML processing, grid-based typesetting, stepcharts, MetaFun and MathML, and more.

magazines This relatively new set describes smaller aspects of ConTEXt and typesetting. A specific volume, for example, is about formatting digits, and another volume is about hiding parts of section titles inside running heads.

qrcs quick references. Each one of these documents contains a list of all available user level commands within a language interface. For every command, its general syntax is described along with the list of allowed parameters and keywords, but nothing else.

sources sample documents with source code. Currently this set consists mainly of the presentation styles that are shipped with ConTEXt. The source of these styles is also well documented using TEX comments.

technotes At the moment there is only one article about graphic inclusion and positioning in PDF available.

uptodate like the special documents in the section *manuals*, the *uptodate* documents are about specific issues in ConTEXt. There are manuals about flowcharts, tables, typesetting Chinese and JavaScript. Nowadays most of the *uptodate* documents are renamed and put into the *manuals* section.

What is missing?

Considering the great number of manuals, it may sound strange that there could be something missing. But those who look at *the manual* will probably notice that it lacks a chapter about typesetting tables. There is a section about a simple table variant (*tabulate*), but it is not explained in any detail. Another variant (*table*) appears often in the examples, but a more helpful explanation is missing. And it lacks an overview of the many different options that can be used when trying to typeset a table.

Another thing you will not find in the printed manuals are practical hints. Things like installation issues, design discussion, tricky problems that are due to misunderstanding or misconfiguration, et cetera.

Yet another problem is that the descriptions in the manuals are partly outdated. The *uptodate* (or the descendants) are, despite their name, already several years old. And because the development of ConTEXt is quite fast, the descriptions of the commands are often incomplete and in some areas, even the underlying concepts have changed. You can only keep track of these things by keeping an eye on the mailing list and watching the changes in the ConTEXt distribution.

Moreover, because there are so many documents, you can actually lose the grand picture. Where are the rules for grid typesetting? What line separators am I allowed to use in *tabulate?* The answers are spread throughout the documents. A global index would be helpful.

A specific request that frequently arises on the ConTEXt mailing list is the lack of sample documents with source code. On the web pages of PRAGMA ADE you can download a few: the magazines (ThisWay); the pdfTEX manual; and the commented sample presentation styles. But for many users this is not sufficient.

Yet another issue that some users are unhappy about has nothing to do with documentation. ConTEXt is not perfectly supported by the TEX-distributions, although the situation has greatly improved over the last years. "Just" trying out ConTEXt sometimes fails because the local distribution lacks some files or because it contains a way too old version of ConTEXt. At present there is a change in the TEX directory structure in *web2c* and accordingly a change of the directory layout of ConTEXt. This brings a whole new set of compatibility problems.

In the garden

With the project *contextgarden.net*, I provide some applications and web services that jointly try to address the problems mentioned above. Right now, the following services are available: the wiki, texshow-web, live ConTEXt, source browser and archive. The services are all linked from the main page.[2] *contextgarden.net* is kindly sponsored by DANTE e.V., the German TEX user group.

Wiki

A wiki is a service, where the visitors of the web page create content. Every visitor can add, change and delete the web pages. The following definition is taken from the wikipedia, a free encyclopedia that is based on the same principle:[3]

> A Wiki or wiki (pronounced "wicky" or "weekee") is a web site (or other hypertext document collection) that allows a user to add content, as on an Internet forum, but also allows that content to be edited by any other user [...] Wiki wiki comes from the Hawaiian term for "quick" or "super-fast".

On every wiki page there is a button labeled "edit", that lets you modify the page content. The wiki syntax is simple, plain ascii text with some extensions for mark-up, such as = for heading and *, # for items in an unordered and ordered list respectively. The following example should be easy to understand for all LaTEX and ConTEXt users.

```
== This is the main title ==
Every page should begin with an introductory
paragraph.

* first item in an unordered list
* and the second

<code>
an environment like verbatim or \starttyping ...
\stoptyping
</code>
```

A table of contents is inserted automatically if there are enough headings on a page.

The main application on *contextgarden* is a wiki especially for ConTEXt. Its content is mainly filled by only a few, but quite active, users of the site. There are about 100 pages on the different topics that cannot be (easily) found in the manuals. One of the most obvious differences between the wiki and the manuals is that in the wiki, you *can* find practical texts like installation experiences and overviews such as a list of text editors with ConTEXt support. Every user can upload TEX and PDF files to the site, and therefore the wiki is a good platform for the publication of sample documents.

One of the important features in a wiki is that you can get a list of recent changes. This leads to a pragmatic way of editing content: one user creates a page on a specific subject, that is not perfectly worded nor 100% complete yet. Afterwards, other users complete this page by supplying their own knowledge on the subject. As a result, the quality of the page increases over time.

Even news items are put into the wiki. For example, the users can see the list of included changes whenever a new ConTEXt distribution is released. The wiki was started around July 2004. In the long term it should be a full complement to the official manuals.

The wiki on *contextgarden* has several features that makes it well suited for ConTEXt documentation. One

extension is the direct rendering of ConTEXt input. Taking the following input:

```
<context>
\defineoverlay
   [tea]
   [{\green \ss \bf GREEN TEA }]
\framed [height=40pt,
         background=tea,
         align=middle]%
    {\em today \blank for sale}
</context>
```

you get

The wiki internally passes the text to ConTEXt to be typeset. This creates a PDF file from the input, and then the wiki calls on ghostscript to convert this PDF file to a PNG image. The end result is that you can view the typeset example directly from within your web browser.

Also included is a pretty printer for TEX and XML source. This formats and colors source code for readability. Perhaps you are familiar with this behaviour from your text editor. The final extension to the wiki is the ability to create hyperlinks to *texshow-web* (see below). With

```
<cmd>adaptlayout</cmd>
```

a link will be created. When that link is clicked, the page that contains the definition of the command *adaptlayout* within *texshow-web* will be opened.

texshow-web

texshow-web is an alternative implementation of the perl/Tk program *texshow* that comes with every ConTEXt distribution.

What follows is a quick summary for those of you who are unfamiliar with *texshow*: This program gives an overview of all user commands. The full set of the parameters and arguments belonging to a specific command can be shown in a syntactical overview, like this one:

```
\adaptlayout[...,...,...][...,...=...,...]

   [...,...,...]    number
   height           dimension max
   lines            number
```

The output is colored, so that you can easily see what parameters you can use in each argument. In this example the first parameter accepts a list of numbers whereas the second parameter takes a list of assignments. Allowed keywords for the assignment are *height* and *lines*, and the allowed values for height are a *dimension* or the word *max* and for lines a *number* (within TEX's limits of course).

The new web-based variant, *texshow-web* makes it possible to add a comment, a description and any number of examples to the a specific command. Of course *texshow-web* also offers a full-text search, so that you can find the command you need with more ease. As with the wiki, all users have the possibility to complete missing information or to correct pages when needed.

ConTEXt users often struggle because one not only has to know all of the possible parameters for a command by name, but one also has to understand their effects on typesetting. In the manuals the parameters are only partly described. For example, there is no explanation of the parameter *beforehead* in an *itemize* environment. That is why *texshow-web* has a field *description* where this kind of information can be stored. Currently, there are only a few entries with that additional content, but progress is slowly being made.

The original *texshow* program from the distribution has recently been adapted to show all the comments, descriptions and examples from *texshow-web* as well, but does not allow editing.

Features currently in development are: a way of categorizing commands into logical units (graphic inclusion, section and headers, typographic commands), a multi-lingual user interface, and documentation for ConTEXt's programming interface (API). The API documentation will cooperate with the *source browser* (see below) that shows the definition of the commands within ConTEXt's source code.

Until now, *texshow-web* has only been used for ConTEXt documentation. But it should be possible to use it for LaTEX documentation without big difficulties.

Archive

There were already several searchable archives for the ConTEXt mailing list: the NTG has one and the newsmail portal Gmane has one as well. Slavek Zak used to host one, too. There are some other, less popular, archives as well. But none of the those is complete as well as easily searchable. Therefore I have installed yet another mailing list archive at *contextgarden.net*. This one is almost complete, very quick and searchable. The two new lists are also archived on this server: the ConTEXt developer list and the *foxet* (a ConTEXt based XML-FO processor) list.

Live
Live ConTEXt is an on-line ConTEXt typesetting service. You type your document into a web-form and after you submit the form to the server, your source gets processed by *texexec* and typeset. The resulting pdf document as well *texexec*'s screen output can be viewed online or saved to your harddisk. This makes it possible to use ConTEXt without actually installing it. The underlying TEX system uses the latest teTEX beta and ConTEXt distribution. *Live ConTEXt* is also used as a sort of reference installation. Errors that do occur on a local system but cannot be reproduced on *Live ConTEXt* are most likely a local problem only.

Source
With the *source browser* you can view ConTEXt's source code. Using a simple navigation system you have access to the almost 600 files from the current distribution. This is especially interesting to the programmers that need to see the definition of the commands. The included full-text search helps finding commands. You can access the definition of the commands via hyperlinks: `http://source.contextgarden.net/tex/context/base/core-pos.tex#setpositions` and `http://source.contextgarden.net/core-pos.tex#setpositions` points to the definition of *setpositions*, without having to know the line number in advance.

Future work

The services mentioned here are provided by *contextgarden.net*. The most important one is certainly the *wiki*. Until now it was not necessary to structure the information in the wiki. If it continues to grow as it did in the past, a more formal structure will be necessary. *texshow-web* should become the preferred reference for the ConTEXt commands. The resources provided by *texshow-web* will be used by the existing tools (*texshow*) and documentation in future releases. Because the number of descriptions, comments and examples is still low, the ConTEXt community is asked to fill in the missing information.

Currently a search engine is in development that allows the user to search *texshow-web*, the *wiki*, the manuals and the mailing list archive all from one place. This should provide an even more powerful tool in your quest to find the necessary information.

Notes

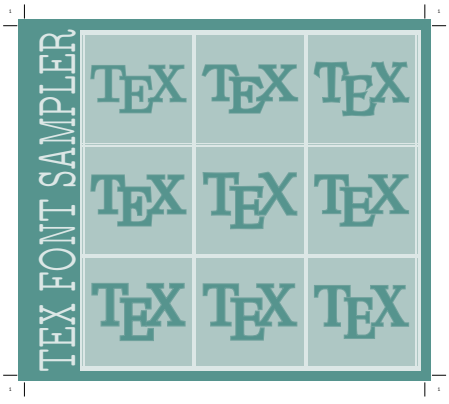
1. `http://www.pragma-ade.com`
2. `http://contextgarden.net`
3. `http://en.wikipedia.org/wiki/Wiki`

Patrick Gundlach
patrick@gundla.ch

Fonts, more than a sample

how to use the fonts shown in the font sampler

Keywords
ConTeXt, fontsampler, howto, TeXlive



Some time ago the NTG members received a colorful little booklet showing a lot of fonts. Since these fonts come with TEXlive a ConTEXt user may be tempted to use them. The bad news is that fonts are always a bit troublesome in TEX distributions and recent changes in the TEX directory structure haven't made life easier. However, the good news is that it is doable to get these fonts working for you. Here I will present a few recipes, but I avoid discussing the 'dirty details'. These are covered in the manuals.

Let's assume that you have a recent version of ConTEXt on your system. In that case, you can set up Palatino as follows:

```
\usetypescript[palatino][\defaultencoding]
```

after this you can choose this font with:

```
\setupbodyfont[palatino,10pt]
```

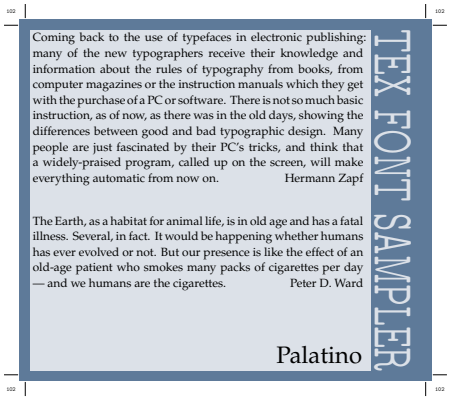
This Palatino typescript is defined as follows (in `type-exa.tex`):

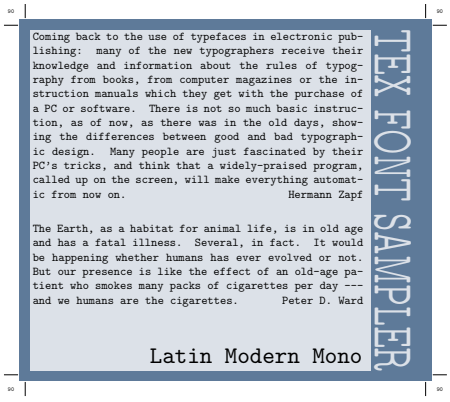
```
\starttypescript [palatino] [texnansi,ec,8r,t5,uc]

  \definetypeface [palatino] [rm] [serif] [palatino]
      [default] [encoding=\typescripttwo]
  \definetypeface [palatino] [mm] [math]  [palatino]
      [default] [encoding=default]
  \definetypeface [palatino] [tt] [mono]  [modern]
      [default] [encoding=\typescripttwo,rscale=1.075]
```

```
\stoptypescript
```

So, in fact we have a combination of Palatino Roman, Palatino Math and Computer Modern Typewriter fonts. From this definition you may conclude that other combinations can be made as well.






```
\usetypescript[times][\defaultencoding]
```

This will provide you with a combination of Times Roman, Times Math, Helvetica and Computer Modern Typewriter. If you really want Courier, you can choose

```
\usetypescript[postscript][\defaultencoding]
```

In one document you can mix such font collections. When you define your own collection, you need to be aware of the fact that they may need relative scaling. In the Famous Postscript Three, the Helvetica is scaled to 90% of Times, and Courier is 10% larger than Times.

As demonstrated in the font booklet, Palatino and Times have companion math fonts. A fourth alternative is the Fourier/Utopia combination:

```
\usetypescript[fourier][\defaultencoding]
```

For this to succeed, you need to copy the Utopia fonts from the additional `texmf-extra` tree to the main `texmf` tree or one of your local trees.

In addition to these fonts, there are a few more on TEXlive. These can be defined as follows:

```
\definetypeface[nicefont][rm][serif][schoolbook][default][encoding=ec]
```

This means as much as: define a typeface called `nicefont`, and use for its roman incarnation the serif `schoolbook`. Some font families have serif as well as sans versions so we need that detail. We define the default styles and sizes and use `ec` encoding.

The next definition also defines `nicefont` but this time we use Bookman for serif and Chancery as calligraphic.

```
\definetypeface[nicefont][rm][serif]       [bookman]  [default][encoding=ec]
\definetypeface[nicefont][cg][calligraphy][chancery][default][encoding=ec]
```

The calligraphic variant in the `nicefont` typeface is activated with (for instance):

```
\setuptypeface[nicefont,cg,11pt]
```

Of course, when making a booklet like the font sampler, a few more tricks are involved. For instance, we use some special encodings in order to access the umlauts in the German sample texts. When you look carefully, you will also notice that hanging punctuation is used.

We execute the typescript:

```
\usetypescript[serif][hanging][pure]
\stoptypescript
```

which comes down to:

```
\starttyping
\setupfontsynonym [Serif]             [handling=pure]
...
\setupfontsynonym [SerifBoldSlanted] [handling=pure]
```

Next we define a typeface:

```
\usetypescript[palatino][\defaultencoding]
```

Now we can enable hanging punctuation and switch to that font:

```
\setupbodyfont[palatino,10pt]
\setupalign[hanging]
```

In a similar fashion one can enable *hz* optimization (aka font expansion). You can control both mechanisms in several ways, and we will dedicate a manual to that once pdftex's font extensions are stable.

There are two kinds of possible errors indicating that something is wrong with your system or definitions.

When during the run, TEX complains that it cannot find a TFM file, this means that either the file is not present on your system, or that you asked for an encoding that has no associated metrics (yet).

Assuming that you use EC encoding, you can try one of the following:

```
\usetypescript[berry][ec]
```

This tells ConTEXt that it should map its internal font names on the eight character based naming scheme found on most distributions. Beware, other encodings may not be present on your system.

While the previous command maps fonts like Times and Palatino on the URW metrics, the next command will use metrics related to the Adobe fonts.

```
\usetypescript[adobekb][ec]
```

The URW fonts come with TEXlive, while the Adobe fonts normally live in printers, browsers and operating resources (what font is actually used, depends on your machine and configuration).

If you use TEXFONT to generate your metrics you don't need these commands. You can put these preferences in your local `cont-sys.tex` file.

When TEX does not abort on an error, but the log mentions that the font cannot be embedded, this can be due to a missing encoding file (happens rarely), a missing map file (more possible) or a missing PFB file (often due to a map file problem).

ConTEXt prefers to load map files at run time because this works best in more complex situations or when you install your own fonts. In your local copy of `cont-sys.tex` you can configure this. In its original `cont-sys.rme` you can read how to do this. Nowadays ConTEXt comes preconfigured for loading map files on demand. Future versions will further hide this nasty aspect of font usage.

More information on installing fonts can be found at

```
http://www.pragma-ade.com/general/manuals/mfonts.pdf
http://www.pragma-ade.com/general/manuals/mtexfont.pdf
http://contextgarden.net
http://home.salamander.com/~wmcclain/context-help.html
```

Examples of fonts in action can be found at:

```
http://www.pragma-ade.com/general/manuals/showfont.pdf
http://www.pragma-ade.com/specials/fonts/fontspecial-s.pdf
```

Hans Hagen
pragma@wxs.nl

Bloei der decadence

Johan Polak

Keywords
Maps, ConTeXt, Math-mode, Greek setup, layout, opmaak, screen-document, reference-list, symbol

Abstract
Het boek *Bloei der decadence* van Johan Polak was jarenlang uitverkocht maar het is als PDF weer beschikbaar gekomen. Via Internet is het vrij te downloaden. Voor wie het op het computerscherm wil lezen is er een interactieve schermversie en een andere versie is geschikt om uit te printen. Beide nieuwe edities van het boek zijn opgemaakt in de ConTEXt-omgeving. Dit artikel beschrijft enige aspecten van het opzetten van een project. Door de gigantische hoeveelheid referenties naar boeken, tijdschriften, personen, plaatsnamen en andere termen werd het een gecompliceerde klus. Doel was om de wijze van verwerking toch zo simpel mogelijk te houden. Dus geen plain-TEX hacking maar simpele methodes die gebruik maken van de typische ConTEXt-eigenschappen. Ook wilden we met een overzichtelijk klein aantal stuurbestanden het gecodeerde boek als scherm- en als papierversie kunnen compileren. Een bijzondere uitdaging vormde hiernaast het feit dat er in een voetnoot een stuk Griekse tekst is weergegeven.

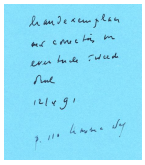
Introductie

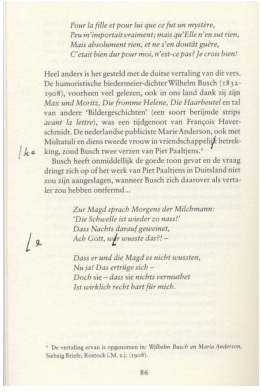
Begin 1988 ontving Johan Polak het eredoctoraat van de Universiteit van Amsterdam "omdat hij als onbaatzuchtige en erudiet uitgever de Nederlandse intelligentsia verrijkt heeft met talloze eminente klassieke en moderne werken in wetenschappelijk verantwoorde edities en vertalingen", aldus prof. dr. D. Leeman bij de uitreiking.

Johan Polak was eens begonnen aan een dissertatie over de antieke literaire kritiek maar zijn perfectionisme bracht hem ertoe telkens na overleg met zijn promotor het zojuist voltooide eerste hoofdstuk weg te gooien en opnieuw te beginnen. "Zijn kracht lag niet op het terrein der strenge wetenschap, maar op dat van het literair- en cultuurhistorisch essay [en] bovenal toch in de editoriale verbreiding van literaire scheppingen van anderen", schreef zijn promotor Leeman in Folia, 5 juni 1992.

Het eredoctoraat verloste Johan Polak misschien van de zelfopgelegde druk tot het voltooien van een proefschrift en met wat vrijere hand werkte hij daarna aan enkele boeken die de aandacht trokken. *Zeven kleine studies* verscheen in kleine oplage bij Ad ten Bosch in Zutphen maar bij verschijning van zijn 'levenswerk' *Bloei der decadence* in 1991 bij Uitgeverij Balans in Amsterdam kwam er voor zijn werk grote belangstelling en erkenning in dagbladen en andere media, tot en met populaire praatpogramma's van presentatoren als Sonja Barend, Hanneke Groenteman en Adriaan van Dis, persoonlijkheden die zich destijds tot de bekende Nederlanders konden rekenen. Het boek werd een bestseller. Een jaar later volgde *Het oude heden*, eveneens bij Uitgeverij Balans. Op 25 mei 1992 zag hij, met de trein in Arnhem aangekomen, het boek (enkele auteursexemplaren had hij al ontvangen) voor het eerst in de etalage van de Arnhemse Boekhandel Hijman liggen en diezelfde warme lenteavond is hij overleden.

In Arnhem had Johan Polak samen met Frans Goddijn een kleine kantoorwoning waar ze in een maatschap werkten aan columns, redactiewerk en andere publicaties. Het copyright voor zijn twee laatste boeken bracht Johan Polak onder in de maatschap. Toen *Bloei der decadence* uitverkocht raakte en een nieuwe druk bij de oorspronkelijke uitgeverij niet haalbaar was, besloot Frans Goddijn om het boek op andere wijze beschikbaar te stellen.





Pour la fille et pour lui que ce fut un mystère,
Peu m'importait vraiment; mais qu'Elle n'en sut rien,
Mais absolument rien, et ne s'en doutât guère,
C'était bien dur pour moi, n'est-ce pas? Je crois bien!

Heel anders is het gesteld met de duitse vertaling van dit vers. De humoristische Biedermeier-dichter Wilhelm Busch (1832–1908), voorheen veel gelezen, ook in ons land dank zij zijn *Max und Moritz*, *Die fromme Helene*, *Die Haarbeutel* en tal van andere 'Bildergeschichten' (een soort berijmde strips *avant la lettre*), was een tijdgenoot van François Haverschmidt. De nederlandse publiciste Marie Anderson, ook met Multatuli en diens tweede vrouw in vriendschappelijke betrekking, zond Busch twee verzen van Piet Paaltjens.[1]

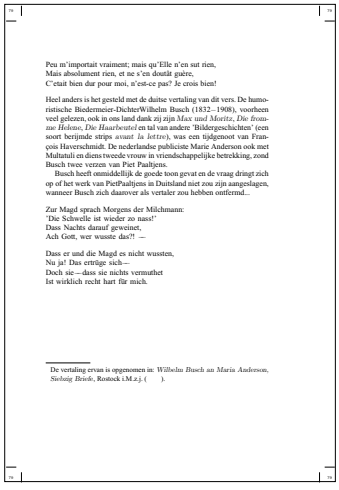
Busch heeft onmiddellijk de goede toon gevat en de vraag dringt zich op of het werk van Piet Paaltjens in Duitsland niet zou zijn aangeslagen, wanneer Busch zich daarover als vertaler zou hebben ontfermd...

Zur Magd sprach Morgens der Milchmann:
'Die Schwelle ist wieder zo nass!'
Dass Nachts darauf gewinnet,
Ach Gott, wer wusste das?! –

Dass er und die Magd es nicht wussten,
Nu ja! Das ertrüge sich –
Doch sie – dass sie nichts vermuthet
Ist wirklich recht hart für mich.

[1] De vertaling ervan is opgenomen in: *Wilhelm Busch an Maria Anderson*, Siebzig Briefe, Rostock i.M. z.j. (1908).

86



Peu m'importait vraiment; mais qu'Elle n'en sut rien,
Mais absolument rien, et ne s'en doutât guère,
C'etait bien dur pour moi, n'est-ce pas? Je crois bien!

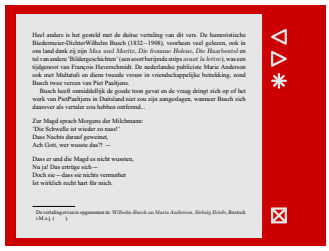
Heel anders is het gesteld met de duitse vertaling van dit vers. De humoristische Biedermeier-DichterWilhelm Busch (1832–1908), voorheen veel gelezen, ook in ons land dank zij zijn *Max und Moritz*, *Die fromme Helene*, *Die Haarbeutel* en tal van andere 'Bildergeschichten' (een soort berijmde strips *avant la lettre*), was een tijdgenoot van François Haverschmidt. De nederlandse publiciste Marie Anderson ook met Multatuli en diens tweede vrouw in vriendschappelijke betrekking, zond Busch twee verzen van Piet Paaltjens.

Busch heeft onmiddellijk de goede toon gevat en de vraag dringt zich op of het werk van PietPaaltjens in Duitsland niet zou zijn aangeslagen, wanneer Busch zich daarover als vertaler zou hebben ontfermd...

Zur Magd sprach Morgens der Milchmann:
'Die Schwelle ist wieder zo nass!'
Dass Nachts darauf gewinnet,
Ach Gott, wer wusste das?! –

Dass er und die Magd es nicht wussten,
Nu ja! Das ertrüge sich –
Doch sie – dass sie nichts vermuthet
Ist wirklich recht hart für mich.

De vertaling ervan is opgenomen in: *Wilhelm Busch an Maria Anderson*, Siebzig Briefe, Rostock i.M. z.j. ( ).



Heel anders is het gesteld met de duitse vertaling van dit vers. De humoristische Biedermeier-DichterWilhelm Busch (1832–1908), voorheen veel gelezen, ook in ons land dank zij zijn *Max und Moritz*, *Die fromme Helene*, *Die Haarbeutel* en tal van andere 'Bildergeschichten' (een soort berijmde strips *avant la lettre*), was een tijdgenoot van François Haverschmidt. De nederlandse publiciste Marie Anderson ook met Multatuli en diens tweede vrouw in vriendschappelijke betrekking, zond Busch twee verzen van Piet Paaltjens.

Busch heeft onmiddellijk de goede toon gevat en de vraag dringt zich op of het werk van PietPaaltjens in Duitsland niet zou zijn aangeslagen, wanneer Busch zich daarover als vertaler zou hebben ontfermd...

Zur Magd sprach Morgens der Milchmann:
'Die Schwelle ist wieder zo nass!'
Dass Nachts darauf gewinnet,
Ach Gott, wer wusste das?! —

Dass er und die Magd es nicht wussten,
Nu ja! Das ertrüge sich —
Doch sie — dass sie nichts vermuthet
Ist wirklich recht hart für mich.

De vertaling ervan is opgenomen in: *Wilhelm Busch an Maria Anderson*, Siebzig Briefe, Rostock i.M. z.j. ( )

Er moest een schermversie komen en ook een versie die met een gewone printer uitgedraaid kon worden. Verschillende externe mensen hebben meegeholpen om dit project op te starten.

Om te beginnen zorgde Dimitri Vanoverbeke, een zoon van NTG-lid Philippe, ervoor dat de papieren pagina's van het boek werden gescand en hij converteerde deze met zijn OCR-programmatuur naar een tekstbestand dat op de computer kan worden verwerkt.

Intussen bood Hans Hagen (NTG, Pragma) hulp bij installatie van zijn ConTEXt en bij het opzetten van de omgeving, de basiscommando's en settings voor de bladspiegel en een slimme manier waardoor van hetzelfde bronbestand telkens naar keuze een PDF voor scherm of printer kon worden gecompileerd. Ook werd er veel tijd en zorg besteed aan typografische zaken zoals de expert set van het gekozen font Times, het kiezen voor 'oldstyle' cijfers in voetnoten, jaartallen, bladzijnummers en dergelijke.

Na het aldus verwerken van enkele hoofdstukken kwamen er andere dagelijkse beslommeringen die het werk lange tijd verhinderden.

Begin 2004 heeft Willi Egger het voortouw genomen en hij kon op basis van zijn expertise en werkervaring met bijzondere boekproductie en vooral dankzij de investering van veel energie en tijd op korte termijn de klus afmaken.

Inmiddels kan het boek als schermversie en ook als versie voor het afdrukken van het internet geplukt worden.

Het boek is te vinden onder http://www.johanpolak.nl. Op deze plaats zullen ook voorbeeldbestanden worden geplaatst zodat lezers die zelf op de hier beschreven manier aan de slag willen de eerste bestanden niet zelf hoeven over te tikken.

Eigenheden van het boek

Bloei der decadence is in 1991 uitgegeven als gebonden boek. Die druk is niet meer nieuw te verkrijgen. Een exemplaar waarin Johan Polak zelf nog enkele correcties aantekende ligt ten grondslag aan de nieuwe versie.

Aangezien in de essays talloze namen en titels voorbijkomen, waarvan de meeste door de boekverzorgers destijds zijn opgenomen in enkele registers, hebben we dit ook gedaan en waar mogelijk iets beter: sommige verwijzingen in het oorspronkelijke register strookten niet met de tekst en er waren inconsequenties. De productietijd was in 1991 wellicht wat te kort. Wat ook meespeelde was een nadeel van het fenomenale geheugen van Johan Polak, die vele honderden gedichten van buiten kende en zinnen, alinea's en pagina's in zijn hoofd schreef en rangschikte voor hij aan het schrijven begon: aantekeningen van bronnen hield hij niet steeds op papier bij.

Er bleek veel tijd te gaan zitten in het werk aan de registers, terwijl we aanvankelijk nog dachten dat het werk voornamelijk zou bestaan uit het opzoeken en corrigeren van woorden die tijdens het scannen en tekstherkennen waren vervormd en uit plaatsen van ConTEXt opmaakcommando's.

In een van de voetnoten staat een deel van een vers uit de *Odyssee* van Homerus. Hierop komen we straks nog terug.

Setup van het project

Er zijn drie stuurbestanden aangemaakt. De eerste daarvan bevat alle opties die van toepassing zijn op de codering van het boek en de uitvoer. Twee andere stuurbestanden herbergen de specifieke instellingen die nodig zijn voor respectievelijk de schermversie en het papieren document. Het vierde bestand bevat de

gecodeerde tekst van het boek. Door middel van twee kleine bestandjes, die ervoor zorgen, dat naar wens de scherm- of papier-versie wordt gecompileerd, kan de desbetreffende versie worden gegenereerd met een bestandsnaam waaruit blijkt dat het de scherm- of printversie betreft.

In het bestand met de algemene opties staan onder andere drie registers gedefinieerd. Het zijn de registers voor boektitels, personen en algemene zaken:

`\definieerregister [persoon][personen]`	Personenregister
`\definieerregister [zaak][zaken]`	Algemene zaken register
`\definieerregister [boektitel][boektitels]`	Register voor Literatuur

Voor de eenvoudige codering in de tekst worden enkele macros gebruikt die zowel de weergave in de tekst regelen als ook voor een regel in het desbetreffende register zorgen:

`\def\BT#1{{\em #1}% \boektitel{#1}}`	Boektitel weergave in de tekst cursief en toewijzen aan het register boektitel
`\def\Zaak #1{{\em #1}% \zaak{#1}}`	Weergave van algemene zaken in de tekst al dan niet cursief en toewijzien
`\def\Nzaak #1{{#1}% \zaak{#1}}`	van de uitdrukking aan het zaken register

De schermversie heeft een vrij strakke opmaak die met betrekkelijk weinig code opgemaakt kan worden. Hier volgt het volledige bestand:

```
\startomgeving optscherm

\useencoding[ibm]
\stelpapierformaatin
  [S6][S6]
\stellayoutin
  [hoogte=midden,
   kopwit=15pt,
   hoofd=30pt,
   voet=30pt,
   marge=20pt,
   rechterrand=30pt,
   rechterrandafstand=15pt,
   rugwit=50pt,
   breedte=400pt]
\stelkleurenin
  [status=start]
\definieerkleur[one]   [donkerrood]
\definieerkleur[two]   [lichtgrijs]
\definieerkleur[three][lichtgrijs]
\stelachtergrondenin
  [tekst][tekst]
  [achtergrond=kleur,
   achtergrondkleur=two]
\stelachtergrondenin
  [pagina]
  [achtergrondoffset=30pt,
   achtergrond=kleur,
   achtergrondkleur=one]
\stelnummeringin
  [plaats=]
\stelinteractiein
  [status=start,
   kleur=one,
   contrastkleur=one,
   menu=aan]
\stellijstin[hoofdstuk]
  [interactie=paginanummer]
\stelsubpaginanummerin
  [wijze=perhoofdstuk,
   status=start]
\steltolerantiein
  [zeersoepel]
\stelkopin
  [paragraaf]
  [nummer=nee]
\stelinspringenin
  [middel]
\stelkapitalenin
  [sc=ja,
   titel=nee]
\stelhoofdin
  [letter=\bfa,
   kleur=two,
   linkerbreedte=.6\hsize,
   rechterbreedte=.4\hsize]
\stelvoetnootdefinitiein[plaats=links]

\startreusableMPgraphic{Next}
  drawoptions (withpen pencircle scaled 5
    withcolor \MPcolor{three}) ;
  path p ;
  p := (0,0)--(30,15)--(0,30)--cycle ;
  draw p ; % fill p ;
  scale_currentpicture(30pt,30pt) ;
\stopreusableMPgraphic
\startreusableMPgraphic{Prev}
  drawoptions (withpen pencircle scaled 5pt
    withcolor \MPcolor{three}) ;
  path p ;
  p := (30pt,0)--(0,15pt)--(30pt,30)--cycle ;
  draw p ; % fill p ;
  scale_currentpicture(30pt,30pt) ;
\stopreusableMPgraphic
\startreusableMPgraphic{Cont}
  drawoptions (withpen pencircle scaled 5pt
```

```
    withcolor \MPcolor{three}) ;
  path p ; p := (15pt,0)--(15pt,30pt) ;
  for i=-45 step 45 until 90 :
    draw p rotatedaround(center p, i) ;
  endfor ;
  scale_currentpicture(30pt,30pt) ;
\stopreusableMPgraphic
\startreusableMPgraphic{Exit}
  drawoptions (withpen pencircle scaled 5pt
    withcolor \MPcolor{three}) ;
  draw
  (0,0)--(30pt,0)--(30pt,30pt)--(0,30pt)--cycle ;
  draw (0,0)--(30pt,30pt) ;
  draw (30pt,0)--(0,30pt) ;
  scale_currentpicture(30pt,30pt) ;
\stopreusableMPgraphic

\stelinteractiemenuin
  [rechts]
  [onder=,
   boven=,
   onderoffset=-2.5pt,
   bovenoffset=-2.5pt]
\startinteractiemenu[rechts]
  \raw [vorigepagina]   \reuseMPgraphic{Prev} \\
  \raw [volgendepagina] \reuseMPgraphic{Next} \\
  \raw [inhoud]         \reuseMPgraphic{Cont} \\
  \vfilll
  \raw [SluitDocument]  \reuseMPgraphic{Exit} \\
\stopinteractiemenu
\stopomgeving
```

Dit is een mooi voorbeeld om te laten zien hoe METAPOST grafieken doeltreffend gemaakt en in ConTEXt geïntegreerd kunnen worden.

Griekse tekst in een voetnoot

Tijdens het coderen van de tekst realiseerden we ons dat er op een plek enkele woorden in het klassieke Grieks opgemaakt moeten worden. Eerst schrik je er wel even van, want dat heb je natuurlijk niet al een hoop keren mogen oefenen. Toch was het belangrijk die paar woorden weer in de juiste Griekse letters in de nieuwe versie te laten verschijnen.

Nu bestaan er voor TEX wel pakketten en fonts voor het zetten van Grieks. Echter op dit moment is er in ConTEXt nog geen volledige ondersteuning van de Griekse taal. Wat te doen? In de wiskundeomgeving bestaat wel een Grieks alfabet. Dus we wagen een poging om die letters te gebruiken voor ons doel.

De uitdaging bestaat natuurlijk niet in het opmaken van een pi (π) of rho (ρ), want dat kan eenvoudig. Maar hoe krijg ik accenten op de vocalen? Dat lijkt ook niet erg moeilijk, want in TEX-wiskunde bestaan ook accenten, zoals grave (`) en aigu (´). Toch lijkt het nog te mislukken: er zit in die korte tekst een epsilon (ε) met een dubbel accent!

Gelukkig bestaat in ConTEXt de mogelijkheid om symbolen te definiëren. Hierbij dient eerst een fontsynoniem aangemaakt te worden, waardoor het goede wiskunde-font aan een symbolische naam wordt gekoppeld.

Daarna kan men een symbool definiëren door aan een naam voor het symbool een definitie van dat symbool te hangen: door enig puzzelen ontstaat zo de letter ε met de gewenste spiritus lenis en de accent aigu: $\grave{\varepsilon}$.

```
\def\MathFont{times}
\def\MathFontFile{rtxmi}

\definefontsynonym[Grieks][\MathFontFile]
\definesymbol[Epsacc][\getglyph{Grieks}%
   {\char34\kern-.40em\raise.15ex%
   \hbox{\getglyph{\MathFont}{\char146}}%
   \kern-.15em
   \hbox{\getglyph{\MathFont}{\char180}}}]
```

De hele regel laten we vervolgens door TEX in mathmode opmaken:

```
\def\Grieksetekst{%
  $\pi\varepsilon\varrho\grave{\iota}$
  $\gamma\grave{\alpha}\varrho$
  $\chi\alpha\chi\grave{\alpha}$
  $\pi\acute{\alpha}\nu\tau o
     \vartheta\varepsilon\nu\$
  \symbol[Epsacc]
  $\sigma\tau\eta$}
```

De gezochte tekst ziet er dan als volgt uit:

$\pi\varepsilon\rho\grave{\iota}\ \gamma\grave{\alpha}\rho\ \chi\alpha\chi\grave{\alpha}\ \pi\acute{\alpha}\nu\tau o\vartheta\varepsilon\nu\ \grave{\varepsilon}\sigma\tau\eta\ldots$ Odyssee, 14, 270.

Conclusie

Nu is het boek er weer en later kan het als de omstandigheden het toelaten nog eens mooi gedrukt en gebonden worden: de bronteksten zijn er klaar voor. Eigenlijk zou nu ook *Het oude heden* onder handen moeten worden genomen. Wie weet.

Johan Polak wordt door velen gemist en op deze wijze is een van zijn levenswerken opnieuw beschikbaar gekomen.

Frans Goddijn
Willi Egger
frans@goddijn.com
w.egger@boede.nl

Keys and Values

new developments and mechanisms in key processing

Abstract
This article introduces the xkeyval package as an extension of the well-known keyval package. The package provides more flexible commands, syntax enhancements, and a new option processing mechanism for class and package options using the *key=value* syntax.

Keywords
keyval, xkeyval, package options, pointers, presets, category codes, PSTricks

Introduction

The keyval package [2] written by David Carlisle is widely used by package authors to provide the means for users to easily specify a lot of optional arguments for macros. The main advantages of using keyval are that the number of optional arguments is no longer limited to 9 and that the arguments are named and hence there is less chance of confusion about the syntax of a macro.

The package provides means to define key macros which handle the input of the user. These key macros have the form `\KV@family@keyname` and take one argument to handle user input. A key macro can, for instance, be defined by

```
\define@key{family}{pi}%
  {\setlength{\parindent}{#1}}
```

The key macros will be called at the use of `\setkeys` and this will set the keys. When `pi` is used, the key macro will set `\parindent`. Another typical example of its use is

```
\setkeys{family}{pi=10pt,pn=Page~\thepage}
```

The packages keyval and xkeyval are mainly directed to class and package authors. The `\define@key` command usually goes into the preamble or the package and the main interface for users is given by `\setkeys`.

Why a new package?

When working on another package, the need arose to have multiple families in the package. Each family would provide keys for a particular macro or environment. This provided the means to block the use of illegal keys in a macro argument, which could have a destructive effect on the rest of the document. However, it would also be nice to be able to allow the user to set specific keys of each macro or environment globally in the preamble. One could, for instance, think of allowing the user to set the markup of all `example` and `exercise` environments in the document in the preamble, but disallowing changing the markup of `example` environments in `exercise` environments and vice versa. In more complicated settings, specifying keys in macros which are not designed to handle those keys can easily lead to almost untraceable errors. That was the start of the xkeyval package [1].

However, in the process of generalizing keyval, we noticed that a lot of packages had already tried extending the features, all in their own way. Quite some packages, for instance, include a system to allow the use of keys and values in `\usepackage` commands. Most famous examples are the hyperref, geometry and beamer packages. All of these approaches differ in details and are not portable to other packages without reprogramming. This called for a unified approach.

Another extra feature, found for instance in the hyperref package, is the availability of boolean keys which can only be true or false. hyperref actually implements this within the ordinary key system, using `\define@key`. However, since the function that needs to be executed on the use of the key is known (namely, set an “if” command to true or false depending on the input), the system can be simplified.

A final motivation for the new package is based on the fact that the development of the keyval package seemed to have paused since 1999 and that fundamental changes and improvements to the system could more easily be made with a new package. Among the improvements, we find the pointer syntax, the preset

system, the use of multiple families in \setkeys, robust input parsing and the support for the PSTricks family of packages. The remaining sections of this article will discuss these new developments.

Keys and values in package options

First of all, the package supplies macros to declare class or package options, execute them and process them. The macros are available under the usual LaTeX names, but all with a postfix, namely

```
\DeclareOptionX
\ExecuteOptionsX
\ProcessOptionsX
```

These commands allow the user to assign a value to an option just as when using \setkeys. The first macro is based on \define@key and the final two are based on \setkeys. When mypack is using these commands, a user could for instance do

```
\usepackage[textcolor=red,font=times]{mypack}
```

These macros are fully integrated with the LaTeX option system. This, for instance, allows packages to copy global options specified in the \documentclass command, to pass options to other classes or packages and to update the list of unused global options that will be displayed by LaTeX in the log file.

However, key values like author=\textit{Me} in class or package options are not allowed, although they could easily be processed by \setkeys. This restriction results from the design of LaTeX's option processing mechanism which expands the entire option list (keys and values) completely, causing obvious trouble.[1]

To avoid these premature expansions, several kernel macros need to be redefined. xkeyval includes the xkvltxp package which contains these new definitions. Loading this package before loading the class or package which uses xkeyval for option processing, will allow class and package options to contain expandable macros. This file will not be included in the LaTeX 2_ε kernel since it might introduce compatibility conflicts for those using an old kernel but new packages which might depend on this new functionality.

Prefixes, families and pointers

The package provides extended syntax for all of the commands provided by keyval.[2] The syntax for defining keys has been extended with an optional argument to set the prefix of the key macro. It is a good custom for package authors to use a package specific prefix for all internal macros as to avoid possibly redefining a macro of another package. Moreover, this optional argument allows for defining and setting keys in specialized systems such as implemented in the PSTricks package. More details about this system will be discussed in the section about the pst-xkey package.

The syntax for setting keys using \setkeys has been adjusted accordingly. Also, one can specify a list of families which should be scanned when setting keys, as discussed in the introduction. For instance,

```
\setkeys{font,page}{fs=10pt,pn=Page~\thepage}
```

Part of the new syntax is the possibility to use pointers to keys. Pointers allow to assign to keyb the value that has been assigned to keya, irrespective of what that value is. For example

```
\setkeys{family}{\savevalue{keya}=red,%
  keyb=\usevalue{keya}}
```

Here, \savevalue will make xkeyval save the value submitted to keya. \usevalue will use this value again. Note that one can use the \savekeys command to avoid typing \savevalue every time. If, in this example, red is changed to blue no changes are necessary to the value of keyb to assign it blue as well. This is an obvious similarity to TeX's behaviour in the macro case \def\cmdb{\cmda}.

This pointer system can be used as well in the default value system. This system submits a default value to the key macro in case the user has used the particular key, but didn't assign a value to it. One could, for example, define the keys

```
\define@key{fam}{keya}{keya: #1 }
\define@key{fam}{keyb}[\usevalue{keya}]{keyb: #1 }
```

Then the following use of \setkeys

```
\setkeys{fam}{\savevalue{keya}=test,keyb}
```

would result in typesetting

```
keya: test keyb: test
```

We will discuss some technical details regarding the pointer syntax. First of all, the control sequences \savevalue and \usevalue are not defined. Instead, the package considers these as delimiters. A simple parsing step will determine if \savevalue has been used in the key name part. Parsing is also used to substitute occurrences of \usevalue by the saved value. When a pointer is replaced, its replacement will also be scanned again for pointers. This allows for nested pointers in key values. Moreover, it makes sure that, once the value is submitted to a key macro, this value does not contain pointers anymore.[3]

The replacement process is a little bit more tricky when the user did not submit a value to the key. In this

case, the default value should be scanned for pointers. The default value macro for a key macro looks like `\prefix@fam@key@default` and has been defined to expand to

```
\prefix@fam@key{the default value}
```

This system has been introduced by keyval and a lot of packages use it. However, some packages do not use it in the way intended by keyval. For instance, the fancyvrb package defines default value macros to execute some code rather than to call the key macro and submit the default value to that. To retain compatibility with existing packages, it is impossible to change the setup of the default value system, only save the default value in the default value macro and submit this value to the key macro.

This is an important restriction for the pointer system since we want to retrieve the default value from the default value macro and scan it for pointers. The way the package proceeds is the following. It first checks whether the default key macro starts as expected, namely with the key macro name. If that is the case, it locally redefines the key macro to save the value to a macro and it executes the key macro. The macro then contains the default value which can be scanned for pointers. If the default value macro is not of the expected form, then the package just executes it without attempting to retrieve the default value or replace pointers.

Preset system

The default value system operates when users specify keys, but no value for the keys. But the keyval package does not provide a system that assigns values to keys when keys have actually not been used at all by the user. In a lot of applications, one would like to implement default values for keys when they are not used. For instance, 'scale this figure with factor 1 unless specified otherwise by the user'. One could go ahead and call the key macro with a default value and afterwards, submit the user input to `\setkeys` and possibly overwrite the values that you have just set. This is possible (but quite cumbersome when there are many keys) in cases where keys do not generate material themselves, but, for instance, only set a length.

But what happens if we apply this scheme to keys which are defined as follows?

```
\define@key{fam}{keya}{Your input was: #1}
\define@key{fam}{keyb}{\edef\list{\list,#1}}
```

If we follow the scheme in the first example, both our default value as well as the user input (if present) will be typeset. In the second example, both the default value and the user input will be added to the list contained in `\list`.

To avoid this, xkeyval introduces the preset system. First one declares the keys that should always be assigned and their values using `\presetkeys`, for instance

```
\savekeys{fam}{head}
\presetkeys{fam}{head=red}{tail=\usevalue{head}}
```

The function of the two arguments of `\presetkeys` will become clear in a moment.

Now, when submitting user input for keys in family `fam`, the macro `\setkeys` will determine which keys will be set by the user and avoid setting them again with the preset values. Keys that are not set by the user will be set by the values specified in `\presetkeys`.

However, there is one thing that we should keep in mind in this system when pointers are used. If the pointer points to a key which is assigned a value afterwards, the pointer cannot know this value yet and errors will occur. Hence, it is best (in most situations) to execute preset pointers at the very end.

On the other hand, if a value for a key has been saved, let's say `blue`, and the user first issues a pointer to that key and later the preset value sets the key to `red`, the outcome of the pointer will of course be `blue`, which was actually not the intention when setting the preset value to `red`. Hence, for ordinary keys, it is best to execute them at the very beginning, before setting user input.

That is why the `\presetkeys` macro has two arguments: the first one (usually containing ordinary keys and values) will be inserted before setting user input keys, the second one (containing pointers to preset values or user input) afterwards.

This system is especially useful when you can't rely on key values remaining local to a macro or environment since the preset system will, at every use of your macro or environment, reset key values to the preset value unless overwritten locally by the user. This needs some more explanation. `\def` definitions (for instance made by key macros) will be destroyed by TEX when leaving a group or environment. Hence the values will remain local. However, if your keys are not using `\def`, but for instance, `\gdef`, this definition will escape the group or environment and might distort all following macros or environments. Hence, you will have to take care to reinitialize the key values at every use of the macro or environment.

This is, however, not necessary anymore with the preset system. Once the preset keys have been defined for a specific family, each time this family is used in the `\setkeys` command, the preset values will be taken into account together with the user input.

The following example will demonstrate the power

of the preset system in combination with pointers. Below the example, you can find its output and the explanation. Let's assume we want to create a simple frame/shadow box command with the following *default* behaviour:

- a shadow will be drawn if and only if the box is framed;
- the shadow color should be a 40% tint of the frame color, thus being clearly discernible;
- the shadow size (or width) should be 4 times the width of the frame.

Certainly, the user should be able to overrule each of these default parameter relations when the box command is actually applied.

```
\documentclass{article}
\usepackage{xkeyval}
\usepackage{calc,xcolor}

\makeatletter
\newdimen\shadowsize
\define@boolkey{Fbox}{frame}[true]
\define@boolkey{Fbox}{shadow}[true]
\define@key{Fbox}{framecolor}%
 {\def\Fboxframecolor{#1}}
\define@key{Fbox}{shadowcolor}%
 {\def\Fboxshadowcolor{#1}}
\define@key{Fbox}{framesize}%
 {\setlength\fboxrule{#1}}
\define@key{Fbox}{shadowsize}%
 {\setlength\shadowsize{#1}}
\savekeys{Fbox}{frame,framecolor,framesize}
\presetkeys{Fbox}%
 {frame,framecolor=red,framesize=0.5pt}%
 {shadow=\usevalue{frame},
  shadowcolor=\usevalue{framecolor}!40,
  shadowsize=\usevalue{framesize}*4}
\newcommand*\Fbox[2][]{%
 \setkeys{Fbox}{#1}%
 {\ifKV@Fbox@frame\else\fboxrule0pt\fi
  \ifKV@Fbox@shadow\else\shadowsize0pt\fi
  \sbox0{\fcolorbox{\Fboxframecolor}{white}{#2}}%
  \hskip\shadowsize
  \color{\Fboxshadowcolor}%
  \rule[-\dp0]{\wd0}{\ht0+\dp0}%
  \llap{\raisebox{\shadowsize}%
   {\box0\hskip\shadowsize}}}%
}
\makeatother

\begin{document}
\Fbox{demo1}
\Fbox[framecolor=blue]{demo2}
\Fbox[shadow=false]{demo3}
\Fbox[framesize=1pt]{demo4}
\Fbox[frame=false,shadow]{demo5}
\end{document}
```

demo1 demo2 demo3 demo4 demo5

First of all, lines 7 to 16 define the keys to be used in the example. The `\presetkeys` command in line 18 defines the presets: the frame will be set to true, its color to red and the frame size to 0.5pt, unless the user provides different specifications for these keys. The requirements listed above are then covered by the pointer expressions in the next argument.

The first box application now shows the default box without additional user input. We see a frame and a shadow, based on the color red. The second box shows that the user input for the frame color will overwrite the preset values and turn the box blue. But since the shadow color equals the frame color by default, the shadow is blue as well. In the third example, we have a frame, but no shadow. Notice that the color has returned to red, the default value. The fourth box has an increased frame size and hence an increased shadow size as well due to the pointer use when presetting the keys. The last example shows that it is possible to overwrite the default behaviour of linking shadows to frames: it displays a shadow without a frame.

Robust parsing

Just as with the pointer delimiters `\savevalue` and `\usevalue`, keyval and xkeyval treat the comma and the equality sign as delimiters. In the past, this has led to problems. A well known incompatibility exists between the Turkish language version of the babel package and all packages using keyval. Since Turkish babel changes the catcode of the equality sign for shorthand notation, the parsing macros of keyval cannot detect these characters anymore and will generate errors. [4]

xkeyval solves this by sanitizing (i.e. setting the catcode to 12) all characters necessary to parse the input properly. This is done using the `\@selective@sanitize` macro, which can sanitize one or more different characters in a single run. Moreover, the sanitize group depth can be controlled. xkeyval implements the macro such that only commas and equality signs appearing in the top level of a key value will be sanitized, since that is all that's needed for input parsing. Characters inside groups are left untouched and can hence possibly even contain babel shorthand notation without causing errors:

```
\usepackage[turkish]{babel}
...
\setkeys{fam}{key={some =text}}
```

In this example, the first '=' will be sanitized for parsing, whereas the second '=' will be left untouched and thus keeps its original meaning.

Redefining macros?

Obviously, redefining existing macros is dangerous in general. Still the xkeyval package redefines the two major keyval macros `\define@key` and `\setkeys`. The reason is that this will avoid any confusion of having several systems running next to each other, doing approximately the same things.

Although xkeyval supports all of the originally possible syntax of the keyval package, we still had to check the packages using keyval before we could make the decision to redefine the macros. Three major issues came up in that process.

First of all, we found that some packages were using keyval internals directly instead of the user interface formed by `\define@key` and `\setkeys`. To avoid any errors of undefined control sequences in these packages, xkeyval loads the keyval internals if keyval hasn't been loaded before.

Secondly, certain packages implemented a creative use of the default value system as has been discussed in the section about the pointer syntax. The solution of xkeyval has also been discussed there.

Finally, we found that the pst-key package was redefining `\define@key` and `\setkeys` itself to provide the means of setting PSTricks keys. After discussing this with the PSTricks maintainer Herbert Voß, we agreed that xkeyval would develop a unified approach to keys and values and that the pst-key package would be abandoned. More information on the development related to PSTricks is provided in the final section of this article.

After redefining the necessary macros, xkeyval will make sure that the keyval package cannot be loaded anymore in order to avoid again redefining the xkeyval macros. This was the final step necessary in safely redefining the keyval macros and to provide a system which all package authors can convert their package to without too much effort.

The pst-xkey package

An important stream of packages will be using xkeyval already in the near future. These are the PSTricks packages [3, 4], currently relying for key and value processing on a combination of private definitions in `pstricks.tex` and pst-key, the latter being a modification of the keyval package.

Due to the popularity and flexibility of the PSTricks package, several people have contributed extensions to the original distribution. Unfortunately, all PSTricks keys have the standard form `\psset@somekey`, thus package authors need to check all existing packages to be sure not to redefine one of the existing keys.

The PSTricks maintainer Herbert Voß has recognized this problem and soon the work on xkeyval started to provide a way to define and set PSTricks keys via this package. The major advantage would be the possibility for individual package authors to nest their keys in a well chosen family (for instance, the package name) and avoid the need to check other packages for existing keys.

In order to make this possible, `\define@key` and `\setkeys` needed to be adjusted a bit. Further, the `\psset` macro needed to be redefined to use the new `\setkeys` and let this scan all families available. When a PSTricks package is loaded, it adds all families used in the package to a list and this list will be used in `\setkeys`. Since all separate packages will use different families, reusing key names is not a problem anymore. The redefinition of `\psset` and some other macros necessary to do the job, is available in the `pst-xkey` package which comes with the xkeyval package.

Due to the vastness of the PSTricks collection of packages, the conversion of all packages to use pst-xkey instead of pst-key will take some time, but will be done in the near future.

Footnotes

1. Note that `author=\protect\textit{Me}` is *no* solution for this problem.
2. Please refer to the documentation of the xkeyval package to learn about further syntactical details which are not discussed in this article.
3. Except if the pointer is hidden for xkeyval inside a group.
4. See for more information concerning this problem of keyval and babel: `http://www.latex-project.org/cgi-bin/ltxbugs2html?pr=babel/3523`

References

[1] Hendri Adriaens. xkeyval package, v1.8c, 2005/01/01. `CTAN:/macros/latex/contrib/xkeyval`.

[2] David Carlisle. keyval package, v1.13, 1999/03/16. `CTAN:/macros/latex/required/graphics`.

[3] Herbert Voß. PSTricks website. `http://www.pstricks.de`.

[4] Timothy Van Zandt et al. PSTricks package, v1.04, 2004/06/22. `CTAN:/graphics/pstricks`.

Hendri Adriaens
`http://stuwww.uvt.nl/~hendri`
Uwe Kern
`http://www.ukern.de`

Boekbespreking Vormwijzer

Keywords
boekbespreking, Treebus, Vormwijzer, Tekstwijzer

Boekbespreking
Vormwijzer: een gids bij het vormgeven en produceren van drukwerk.
K.F. Treebus
's Gravenhage, SDU Uitgeverij
3de druk, 1997

Altijd als ik bij de ramsj-boeken iets zie liggen met een harde kaft en een stofomslag, kijk ik even wat het precies is. Vooral als het bij boekhandel 'De Bengel' hier in Dordrecht op de Voorstraat is, want daar heb ik in het verleden al een aantal aardige boeken vandaan gesleept. Zo vond ik afgelopen september de derde druk van de "Vormwijzer", voor 17.50 euro.

Inleiding

K.F. Treebus is tevens de auteur van "Tekstwijzer", waarvan de ondertitel is 'Een gids voor het grafisch verwerken van tekst'. Dat de "Vormwijzer" in hetzelfde genre valt blijkt overduidelijk uit de ondertitel: 'Een gids bij het vormgeven en produceren van drukwerk'. Beide boeken behoren tot een reeks van boeken over het uitgeef-proces, waarin onder andere ook Renkema's "Schrijfwijzer" en Van der Horst's "Leestekenwijzer" zijn verschenen.

Wie bekend is met de Tekstwijzer zal het meteen opvallen dat dit boek sprekend op zijn voorganger lijkt voor wat betreft de layout. Dezelfde citaatjes in de extra brede linker kantlijn, dezelfde fonts, dezelfde witruimtes rondom de koppen, dezelfde bladvulling. Het enige dat dit boek anders maakt is dat een groot gedeelte van de illustraties in kleur is gezet, en dat er veel meer illustraties opgenomen zijn.

Inhoudelijk is dit boek evenwel heel anders dan de Tekstwijzer. In dit boek ligt de nadruk sterk op het 'de weg wijzen', en veel minder op 'onder-wijzen'. Dat maakt dat het boek zich wat minder leent voor van voor tot achter doorlezen, en juist meer voor naslag. Er is een heleboel informatie opgenomen, en een groot gedeelte van die informatie staat op zichzelf, los van de andere onderwerpen in het boek. Wellicht daarom heeft het boek maar vier hoofdstukken en een literatuurlijst, maar wel een heleboel (sub-)secties binnen die hoofdstukken.

1: Makers, medewerkers, middelen

Dit korte hoofdstuk is meer een vergrote inleiding. Eerst worden functie, inhoud en vorm van de "Vormwijzer" uiteengezet. Daarna legt de schrijver in het globale termen uit hoe het produceren van drukwerk in elkaar steekt en welke taken daarbij komen kijken. De rollen van de initiator, uitgever, vormgevers, begeleiders en uitvoerders worden kort besproken. Als afsluiter wordt aan de hand van de productie van de Vormwijzer zelf een voorbeeld gegeven van hoe deze rollen in de praktijk met elkaar samenwerken.

2: Prenten, platen en puntrasters

Tachtig pagina's worden besteed aan het beschrijven van de grafische processen die nodig zijn om de gewenste informatie op een medium vast te leggen.

De paragrafen aan het begin van dit hoofdstuk gaan vooral over technieken uit het verleden. De toenemende mate van industrialisatie is duidelijk gemaakt in

het verloop van papyrusrollen naar middeleeuwse manuscripten, naar houtsnijwerk, naar blokboeken, naar metaalgravures, naar lithografie, naar fotografie, om te eindigen bij galvanoplastiek.

Na deze vogelvlucht wordt stilgestaan bij de technieken die heden ten dage massaal in gebruik zijn. Op dit punt is het overzicht even een beetje zoek: naast korte beschrijvingen van bijvoorbeeld fotografische technieken en lijnclichés wordt een heel stel pagina's gebruikt voor behoorlijk gedetailleerde informatie over rasters. Zonder meer nuttige informatie, maar het had wellicht beter apart neergezet kunnen worden. Andere onderwerpen in dit gedeelte zijn graveermachines en *under colour removal*.

Het overzicht keert weer terug wanneer de schrijver verder gaat met het beschrijven van de eisen die aan het aangeleverde materiaal gesteld worden, en op welke manier de noodzakelijke reproductie-instellingen worden gecommuniceerd. Logisch genoeg volgt daarna een gedeelte over kwaliteitscontrole van het resultaat, zoals het verifiëren van de gebruikte kleur aan de hand van de meetstrip op geleverd drukwerk.

De laatste paragrafen van dit hoofdstuk gaan beknopt, maar met aardig wat voorbeelden, in op 'photoshoppen' (door de auteur ietwat deftig aangeduid als 'electronische beeldverwerking'), DTP (de auteur bedoelt daarmee opmaken van teksten op een personal computer), en CAD/CAM (wat Treebus duidelijk ziet als een volwassen broer van DTP).

De laatste paar pagina's gaan over datatransmissie, en op dit punt is het boek sterk gedateerd te noemen. Deze uitgave, uit 1997, bevat hier nog de tekst uit de eerste uitgave (1991). Hoeveel mensen herinneren zich anno 2005 *teletex* nog?

3: Papier, pers, produkt

Het derde hoofdstuk gaat over het drukproces zelf. Het begint met een flink stuk over papier. Hoe dat vroeger met de hand gemaakt werd en hoe dat nu (volledig geautomatiseerd) gaat, over papiersoorten en hun prijzen, over verschillende papierformaten en gewichten, en eigenschappen als opdikking en looprichting.

Hierna volgt een fikse paragraaf over drukken en druktechnieken. Ook hier is er eerst weer plaats gereserveerd voor een stukje geschiedenis, gevolgd door een beschrijving van de verschillende technieken: hoogdruk (boekdruk, blinddruk, stempels en flexdruk), vlakdruk (lithografie en offset), diepdruk (rasterdiepdruk, plaatdruk en staalstempeldruk), doordruk (sjabloondruk, stencildruk en zeefdruk).

Na deze heldere maar technische uiteenzetting volgen een paar pagina's over de keuzes die gemaakt kunnen worden, en de effecten die zo'n keus heeft op bijvoorbeeld de katernindeling en aflooprandeen. Hier komen ook wat speciale effecten zoals irisdruk en speciale inkten aan bod.

Het woord 'produkt' uit de hoofdstuktitel wordt ingevuld door een leuk sub-hoofdstuk over onder andere vouwen, snijden en inbinden. Dit is mijns inziens één van de beste gedeelten van het boek. Het is helder geschreven, er zijn duidelijke tekeningetjes ter illustratie van lastige concepten zoals vouw-vormen, er zijn kleurenfoto's van prachtige voorbeeldboeken en lekker weinig foto's van de grote of lelijke apparaten die in de rest van het boek vrij vaak voorkomen.

4: Vorm, functie, vondst

Het laatste hoofdstuk gaat over vormgeving. In verhouding tot de rest van het boek vind ik dit een wat minder waardevol onderdeel. Aan de hand van allerlei beroemde voorbeelden wordt getoond wat er zoal aan (on-)mogelijkheden is voor een vormgever. Al die voorbeelden van typografie en typografische grappen hadden van mij best overgeslagen mogen worden.

Een interessant onderdeel is wel de uiteenzetting over de wetenschappelijke (zowel natuurkundige als psychologische) eigenschappen van kleur waar het hoofdstuk mee begint, en er staat wel degelijk heel zinnige informatie tussen de voorbeelden door. Ik had dit liever wat ingetogener gezien, dat is alles.

Literatuuropgave, register, illustraties

De literatuuropgave is weer een juweeltje. Alleen al die enorme lijsten met verwijzingen maken de boeken van Treebus hun geld waard. In de "Vormwijzer" bestaat de lijst uit niet minder dan 692 verschillende publicaties, overzichtelijk gerangschikt op de paragraaf waar ze bij passen, met extra verwijzingen voor publicaties die gerangschikt staan onder een ander kopje.

Daarna volgt een register. Ikzelf zoek onderwerpen meestal op via de uitgebreide inhoudsopgave voorin, maar de verwijzingen zijn met name handig zijn voor het opzoeken van geciteerde personen en genoemde bedrijfsnamen.

Conclusie

Voor de zeventien-en-een-halve euro die ik eraan besteed heb, is het een prima inleidend werk over alles wat er bij de productie van drukwerk komt kijken. Het boek bevat meer dan genoeg informatie om als buitenstaander je wensen te kennen te geven aan een vakman, en de literatuurlijst maakt het makkelijk om boeken te vinden die dieper ingaan op één bepaald aspect.

Daarbij is het boek zelf stevig uitgevoerd en zeer verzorgd qua productie en vormgeving. Wel is het boek soms wat warrig en hier en daar ietwat gedateerd. Als je het te pakken kunt krijgen bij de ramsj is het boek zeker zijn geld waard. Maar als ik het oorspronkelijke bedrag (zo'n 40 euro) betaald had, zou ik me nu toch ietwat bekocht hebben gevoeld.

Taco Hoekwater

De TEX flyer: doe er wat mee!

Op de volgende twee bladzijden hebben we onze TEX flyer nog eens afgedrukt. De voorkant beschrijft de sterke punten van TEX, en de achterkant bevat de nodige informatie om mensen een snelle start te geven met TEX.

Als je denkt dat er potentiële TEX-gebruikers in je omgeving bestaan, vraag dan een stapeltje van deze flyers aan bij het bestuur (`ntg@ntg.nl`); we hebben er nog genoeg van. Wellicht kun je ze met deze flyer ze over de streep trekken.

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

TEX: meer dan tekstverwerken

TEX is een elektronisch zetsysteem. Oorspronkelijk speciaal voor wiskundig zetwerk ontwikkeld door Prof. Donald E. Knuth, wordt TEX nu gebruikt voor vrijwel alle denkbare tekstuele toepassingen: brieven, boeken, handleidingen, database uitvoer, flyers zoals deze, presentaties en interactieve documenten. TEX is gratis en is beschikbaar voor vrijwel alle computer-platforms. Er is een levendige online gebruikersgemeenschap.

TEX werkt met opmaakcodes, net als html/xml. Een voorbeeld van TEX-invoer:

```
\documentclass{article}
\begin{document}
Hello world!\par
$ \phi = \sqrt x $
\end{document}
```

Hello world!
$\phi = \sqrt{x}$

TEX maakt uit deze invoer naar keuze een pdf-bestand of een afdruk op papier.

Met een wysiwyg tekstverwerker bent u waarschijnlijk sneller aan de slag, maar als u mocht besluiten de sprong naar TEX te wagen dan zult u merken dat TEX een praktisch en efficiënt stuk gereedschap is met enorme mogelijkheden.

TEX is betrouwbaar

Technieken die goed werken voor een document van vijf pagina's werken even goed voor een boek van vijfhonderd. De extra inwerktijd verdient u ruimschoots terug omdat TEX gewoon zijn werk blijft doen, hoe lang en complex het dokument ook wordt.

TEX is professioneel

TEX levert kwaliteitszetwerk af, dankzij ondersteuning van kerning en ligaturen, en gebruik van het beste afbreekalgoritme ter wereld. TEX kan goede kwaliteit PostScript en pdf genereren. TEX is dan ook in gebruik bij wetenschappelijke uitgevers.

TEX is aktueel

- Rechtstreekse ondersteuning van pdf, met inbegrip van de interactieve voorzieningen hiervan. Met pdflatex en het pdfscreen-pakket kunt u gemakkelijk presentaties maken. Met ConTEXt is er nog veel meer mogelijk.
- XML/SGML: TEX wordt hier ingezet voor het genereren van pdf- en gedrukte uitvoer. Pakketten zoals xmlTEX en ConTEXt kunnen xml rechtstreeks verwerken.
- Het Omega-project werkt aan ondersteuning voor Unicode.

TEX kan in specialistische behoeften voorzien

Dankzij het open karakter en de programmeerbaarheid van TEX kunnen slimme TEX-gebruikers oplossingen bedenken voor hun specifieke problemen, die ze dan aan de gemeenschap ter beschikking stellen. Hieraan hebben wij het bestaan te danken van BibTEX voor het automatisch genereren van bibliografieën uit een database, en van Makeindex voor het automatisch genereren van registers. Voor niet-westerse talen, bladmuziek, presentaties en diagrammen zijn oplossingen van hoog niveau ontwikkeld voor gebruikers door gebruikers, getuige de voorbeelden op deze pagina.

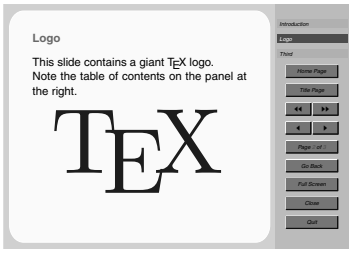
TEX leent zich voor automatische verwerking

In Unix-omgevingen wordt TEX dan ook dankbaar ingezet om automatisch pdf te genereren uit bijvoorbeeld DocBook (Linux Documentation Project) of texinfo (het bronformaat voor documentatie voor GNU software).

Herkomst van de figuren. Het arabtex voorbeeld is ontleend aan de ArabTEX documentatie. Het is de aanhef van een traditionele vertelling. Het xypic diagram en het lilypond muziekfragment zijn eveneens ontleend aan de documentatie van de corresponderende pakketten. De MetaPost-figuur is gebaseerd op een bijdrage van Huib van de Stadt.

$$\sum_{i=1}^{qT-1} u_i^2 + \sum_{i=0}^{T-1} \mu_i \left\{ \left[\sum_{j=0}^{q-1} x_j u_{iq+j} \right] - v_i \right\}$$

Wiskunde



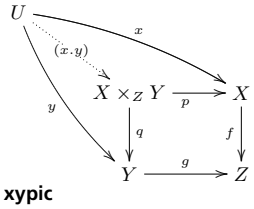
pdfscreen

جُحَا وَحَمِيرُهُ العَشَرَةُ

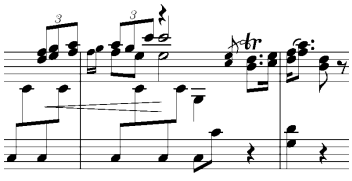
إِشْتَرَى جُحَا عَشَرَةَ حَمِيرٍ. فَرِحَ بِهَا وَسَاقَهَا أَمَامَهُ، ثُمَّ رَكِبَ وَاحِدًا مِنْهَا. وَفِي الطَّرِيقِ عَدَّ حَمِيرَهُ وَهُوَ رَاكِبٌ، فَوَجَدَهَا تِسعَةً. ثُمَّ نَزَلَ وَعَدَّهَا فَرَآهَا عَشَرَةً فَقَالَ:

أَمْشِي وَأَكسِبُ حِمَارًا، أَفضَلُ مِن أَن أَركَبَ وَأَخسَرَ حِمَارًا.

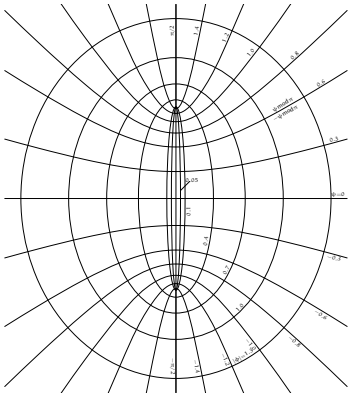
ArabTEX



xypic



Lilypond



MetaPost

Beginnen met TEX

Websites om u nader te oriënteren:

- `www.ntg.nl`, de website van de Nederlandstalige TEX Gebruikersgroep
- `www.tug.org`, de website van de internationale TEX Users Group. Hier vindt u ook een goede 'Getting Started' pagina.
- Speciaal voor Mac gebruikers: `www.esm.psu.edu/mac-tex/`.

TEX op cd

Zowel commerciële als sommige niet-commerciële leveranciers bieden TEX-systemen op cd. Informatie hierover vindt u op hun respektievelijke websites.
De TEX Live cd, die u toegestuurd krijgt als u lid wordt van de NTG, ondersteunt een groot aantal systemen: Windows, Mac OS X, Linux en diverse Unix varianten. U kunt naar keuze TEX vanaf deze cd draaien of op uw harde schijf installeren.
U kunt de TEX Live cd ook bestellen bij bijvoorbeeld Lehmanns Bookshop (`www.lob.de`). Deze boekhandel heeft ook een aantal boeken over TEX en LATEX in zijn catalogus.

TEX software downloaden

U kunt de meeste niet-commerciële en sommige commerciële TEX-systemen ook van het Internet downloaden. In de eerste plaats zijn er de *CTAN* servers. CTAN staat voor 'Comprehensive TEX Archive Network'. Vrijwel alles wat u met betrekking tot TEX zou willen downloaden is hier te vinden. CTAN servers in de buurt zijn:

- `ftp.ntg.nl/pub/tex-archive`
- `ftp.dante.de/pub/tex`
- `ftp.belnet.be/packages/CTAN/`

Al deze servers voeren dezelfde bestanden. De TEX Live cd-images vindt u onder `systems/texlive`.

Voor Linux is echter de meest praktische oplossing om de teTEX-packages uit de distributie te installeren.

LATEX en ConTEXt

Het gebruik van een formaat zoals LATEX neemt auteurs veel werk uit handen. LATEX voegt voorzieningen toe zoals puntenlijsten, automatische inhoudsopgaven en automatisch genummerde hoofdstukken en voetnoten.
Een modern alternatief is ConTEXt, dat uitblinkt in geavanceerde layout, in ondersteuning van de interactieve mogelijkheden van pdf en in integratie van grafische elementen. ConTEXt volgt aktuele ontwikkelingen op de voet. Neem een kijkje op `www.pragma-ade.nl` en abonneer u daar op de ConTEXt discussie-lijst.

De NTG

TEX-gebruikers vinden elkaar in TEX gebruikersgroepen; voor Nederland en België is er de NTG, voluit de Nederlandstalige TEX Gebruikersgroep.

De NTG houdt tweemaal per jaar een gebruikersbijeenkomst met interessante lezingen. Het lijfblad van de NTG, de MAPS, verschijnt eveneens tweemaal per jaar, en bevat zowel artikelen voor beginnende gebruikers als informatie over de nieuwste ontwikkelingen. Tevens steunt de NTG internationale initiatieven.

Onze website is `www.ntg.nl`. U kunt zich daar aanmelden als lid, of u abonneren op een aantal mailinglists, of kijken wat er zoal gebeurt op TEX-gebied in binnen- en buitenland.