MAPS

NUMMER 27 • VOORJAAR 2002

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

Voorzitter
Hans Hagen
pragma@wxs.nl

Secretaris
Jules E van Weerden
Jules.vanWeerden@vet.uu.nl

Penningmeester
Wybo Dekker
Servalys, Deil
wybo@servalys.nl

Bestuursleden
Erik Frambach
erik.frambach@stelvio.nl

Frans Goddijn
frans@iaf.nl

Karel H Wesseling
k.h.wesseling@planet.nl

Postadres
Nederlandstalige TEX Gebruikersgroep
Utrechtsestraatweg 64
3445 AV Woerden

Postgiro
1306238
t.n.v. NTG, Deil
000-1662209-17
t.n.v. NTG, Veldegem, België

E-mail bestuur
ntg@ntg.nl

E-mail MAPS redaktie
maps@ntg.nl

WWW
www.ntg.nl

De **Nederlandstalige TEX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft het bevorderen van de kennis en het gebruik van TEX. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde document-opmaak 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 m.b.t. TEX en 'TEX-producten'.
De NTG biedt haar leden onder andere het volgende:

- Tweemaal per jaar een NTG-bijeenkomst.
- Het NTG-tijdschrift MAPS.
- De 'TEX Live CD-ROM' en de 'CTAN CD-ROM' met de complete TEX software-archieven.
- 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); 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.

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 letterzet-mogelijkheden van TEX. Voorbeelden zijn LATEX van Leslie Lamport, $\mathcal{AMS}$-TEX van Michael Spivak, en ConTEXt van Hans Hagen.

Inhoudsopgave

Nummer 27, voorjaar 2002

Redactioneel

Johannes Braams
jlbraams@cistron.nl

Voor je ligt het zevenentwingste nummer van de MAPS. Het eerste nummer na de EuroTEX conferentie van vorig jaar. Het eeste nummer ook met een nieuwe redactie. Hoewel, ... nieuw? De meest leden van de nieuwe redactie zijn oude bekenden binnen de NTG. Ik stel ze even voor:

Johannes Braams hoofdredacteur. Voor veel mensen die al wat langer lid zijn geen onbekende persoon, ik heb ze als penningmeester jarenlang lastig gevallen met acceptgiro's voor het betalen van hun lidmaatschap.

Piet van Oostrum Ook al bepaald geen onbekende persoon in onze vereniging en daarbuiten. Zoals nagenoeg iedereen wel zal weten is Piet onze wandelende vraagbaak die de meeste vragen die op tex-nl@nic.surfnet.nl doeltreffend weet te beantwoorden.

Karel Wesseling Karel komt van buiten het 'TEX wereldje' en zal ons team met frisse nieuwe ideeën versterken.

Siep Kroonenberg Siep is alweer geruime tijd lid van de redactie en is onder andere verantwoordelijk geweest voor het ontwerp van de vormgeving van de MAPS.

Taco Hoekwater Ook Taco is al geruime tijd lid van de redactie, hij is enige tijd hoofdredacteur geweest en kent dus het klappen van de zweep.

Wij zijn blij dat we weer in staat zijn geweest een nummer samen te stellen met, naar we hopen, voor iedereen leesbare en interessante artikelen.

Van de hand van Siep Kroonenberg hebben we drie artikelen opgenomen met als gezamenlijk thema het 'in de lucht houden' van een TEX omgeving.

Karel Wesseling vertelt in twee artikelen over zijn positieve ervaringen met ConTEXt. Hij gaat in op het opnemen van een lijst met literatuur verwijzingen in een ConTEXt document. In zijn tweede artikel schrijft Karel over zijn meer dan 10 jaar lange ervaring met TEX.

Uit TUGboat 20-3 hebben we twee artikelen mogen overnemen van Jean-Luc Doumont; één over zijn professionele ervaringen met het gebruik van TEX, de andere over zijn macro-pakket voor het maken van grafieken.

Voor díe lezers die geïnteresseerd zijn in Desktop Publishing met LATEX kunnen we het artikel van Ernst van der Storm over dit onderwerp van harte aanbevelen.

Ook in deze MAPS zijn weer bijdragen van Hans Hagen te vinden. Een groot deel van deze MAPS is gewijd aan het MathML manual. Daarnaast heeft Hans een database ontwikkeld waarmee een verzameling van figuren eenvoudig vanuit ConTEXt ontsloten kan worden.

Het onderwerp euro wordt door zowel Hans Hagen als Patrick Grundlach besproken.

Uit de 'Cahiers Gutenberg' is een artikel overgenomen over de praktijk van het werken met Metapost. Het artikel is uit het frans vertaald met hulp van Karel's echtgenote, waarvoor dank!

Last but but not least heeft Piet van Oostrum weer een selectie gemaakt van bijdragen in het CTAN archief.

Colofon

De MAPS wordt gezet met behelp van een LATEX class file en een ConTEXt module. De gebruikte fonts zijn Adobe Times-Roman met Old Style Figures en Frutiger, met een versmalde Courier voor de 'verbatim' omgevingen. Compilatie gebeurt met web2c versie 7.3.3.7 onder Linux 2.4 en we drukken op 90-grams coated papier vanaf een PDF bestand dat wordt gemaakt met pdf[e]tex 1.00 en dvips versie 5.86, gedestilleerd met Adobe's Distiller 4.05, draaiend onder Windows NT.

info

TEX Gebruikersgroepen

Erik Frambach
erik.frambach@stelvio.nl

abstract

Een overzicht van alle ons bekende TEX gebruikersgroepen met bijbehorende contactgegevens.

keywords

gebruikersgroepen, user groups

Associaciia Polźovatelei Kirillicheskogo TEXá
Short name: CyrTUG (Russia)
Website: http://www.cemi.rssi.ru/cyrtug/
Email: cyrtug@mir.msk.su
Contact addresses:
Policy matters: President Eugenii V. Pankratiev
email: cyrtug@cemi.rssi.ru
General matters: Secretary Irina Makhovaya, Executive Director
Mir Publishers
2, Pervyi Rizhskii Pereulok
Moscow 129820
Russia
email: irina@mir.msk.su
phone: +70952860622, +70952861777 *fax*: +70952889522

Association pour la diffusion de logiciels scientifiques liés TEX
Short name: AsTEX
Target language: French
Website: http://www.univ-orleans.fr/EXT/ASTEX/astex/doc/en/web/html/astex000.htm
Email: astex-admin@univ-orleans.fr
Contact addresses:
Policy matters: President Michel Lavaud
Association AsTEX
BP 6532
45066 Orleans cedex 2
France
email: Michel.Lavaud@univ-orleans.fr
phone: +33238640994
Finance / member admin.: Treasurer Delombera Negga
email: Delombera.Negga@wanadoo.fr
Mailing list: astex@univ-orleans.fr

Vietnamese TEX Users Group
Short name: ViêtTUG
Email: viettug@iris.ltas.ulg.ac.be
Contact addresses:
Policy matters: President Nguyên-Dai Quý
LTAS-University of Liège
Rue des Chevreuils, 1
Bât B52, Local 522B
B4000 Liège
Belgium
phone: +3243669098 *fax*: +3243669311

Catalan TEX Users Group
Short name: Tirant lo TeX
Target language: Catalan
Website: http://www-lsi.upc.es/~valiente/tug-catalan.html
Contact addresses:
Policy matters: President Gabriel Valiente Feruglio
Technical University of Catalonia
Jordi Girona Salgado, 1-3
E-08034 Barcelona
Spain
email: valiente@lsi.upc.es
Mailing list: catala-tex@aliga.cesca.es

CervanTEX Grupo de Usuarios de TEX Hispanohablantes
Short name: CervanTeX
Target language: Spanish
Website: http://apolo.us.es/CervanTeX/
Email: josera@us.es
Contact addresses:
Policy matters: Presidente José Ra Portillo Fernández
Universidad de Sevilla Facultad de Informática y Estadìstica Avenida de la Reina Mercedes, s/n
E-41012 Sevilla
Spain
email: josera@us.es
phone: +34954554382 *fax*: +34954557878
General matters: Secretario Roberto Herrero Santos
email: rhs@hersim.com
Finance / member admin.: Tesorero Eugenio Degroote Herranz

email: edegroote@agricolas.upm.es
Publication: TEXemplares
Editor: Javier Bezos
Editor email: jbezos@wanadoo.es
Mailing list: spanish-tex@eunet.es

TEX Users Group
Short name: TUG
Website: http://www.tug.org
Email: office@tug.org
Contact addresses:
Policy matters: President Mimi Jett
1466 NW Front Avenue, Suite 3141
Portland, OR 97209
U.S.A.
phone: +15032239994 *fax*: +15032233960
General matters: Secretary Arthur Ogawa
40453 Cherokee Oaks Drive
Three Rivers CA 93271
U.S.A.
email: ogawa@teleport.com
phone: +15595614585 *fax*: +15596228915
Finance / member admin.: Treasurer Don DeLand
Publication: TUGboat
Editor: Barbara Beeton
Editor email: TUGboat@tug.org

Group francophone des Utilisateurs de TEX
Short name: GUTenberg
Target language: French
Website: http://www.gutenberg.eu.org
Email: secretariat@gutenberg.eu.org
Contact addresses:
Policy matters: President Michel Goossens
c/o Irisa
Campus universitaire de Beaulieu
F-35042 Rennes cedex
France
email: gut@irisa.fr
phone: +33130870625 *fax*: +33130870625
Publication: Cahiers GUTenberg

Indian TEX Users Group
Short name: TUGIndia
Website: http://www.river-valley.com/tug/
Email: tugindia@river-valley.com
Contact addresses:
Policy matters: President K.S.S. Nambooripad
Kripa, TC 24/548, Sastha Gardens
Thycaud, Trivandrum 695014
India
phone: +91471324341 *fax*: +91471333186
Publication: TUGIndia

The Greek TEX Friends Group
Target language: Greek
Website: http://obelix.ee.duth.gr/eft/
Email: eft@ocean1.ee.duth.gr
Contact addresses:
Policy matters: President Apostolos Syropoulos
366, 28th October Str.
GR-671 00 Xanthi
Greece
email: apostolo@obelix.ee.duth.gr
phone: +3054128704
Publication: Eutupon
Editor: Prof. Basil K. Papadopoulos

Ceskoslovenské sdruzení uzivatelu TEXu
Short name: CsTUG
Website: http://www.cstug.cz
Email: cstug@cstug.cz
Contact addresses:
Policy matters: President Petr Sojka
CsTUG, c/o FI MU
Botanická 68a
CZ-602 00 Brno
Czech Republic
email: cstug@cstug.cz
phone: +420541212352
General matters: Secretary
email: secretary@cstug.cz

Magyar TEX Egyesület
Short name: MaTEX
Target language: Hungarian
Website:
http://neumann.math.klte.hu/~ludens/matex/
Email: huntex@math.klte.hu
Contact addresses:
Policy matters: President Antal Járai
Department of Numerical Analysis
Eötvös Loránd University
Pázmány Péter sétány 1/D
H-1117 Budapest
Hungary
email: ajarai@moon.inf.elte.hu
General matters: Secretary Gyöngyi Bujdosó
Institute of Mathematics and Informatics
University of Debrecen
H-4010 Debrecen, P.O. Box 12
Hungary
email: ludens@math.klte.hu
phone: +3652316666 *fax*: +3652416857
Finance / member admin.: Board member Norbert Bátfai
Institute of Mathematics and Informatics
University of Debrecen

H-4010 Debrecen, P.O. Box 12
Hungary
email: cassock@venus.math.klte.hu
phone: +3652316666 *fax*: +3652416857
Mailing list: tex-l@tigris.klte.hu
Subscribe at: http://neumann.math.klte.hu/~ludens/matex/jelentkezes.html

Philippines TEX Users Group
Short name: TUG-Philippines
Contact addresses:
Policy matters: President Felix P. Muga II
Ateneo de Manila University
Loyola Heights
Quezon City
Philippines
email: fpmuga@admu.edu.ph, fpmuga@philonline.com
phone: +6324266001 ext 2515 *fax*: +6324266008

Nederlandstalige TEX Gebruikersgroep
Short name: NTG
Target language: Dutch
Website: http://www.ntg.nl
Email: info@ntg.nl
Contact addresses:
Policy matters: President Hans Hagen
Pragma
Ridderstraat 27
8061 GH Hasselt
The Netherlands
email: ntg-president@ntg.nl
phone: +31384775369 *fax*: +31384775374
General matters: Secretary Jules van Weerden
Utrechtsestraatweg 64
3445 AV Woerden
The Netherlands
email: ntg-secretary@ntg.nl
phone: +31348481707
Finance / member admin.: Treasurer Wybo Dekker
Deilsedijk 60
4158 CH Deil
The Netherlands
email: ntg-treasurer@ntg.nl
phone: +31345652164 *fax*: +31345652383
Publication: MAPS
Editor: Siep Kroonenberg
Editor email: maps@ntg.nl
Mailing list: tex-nl@ntg.nl
Subscribe at: listserv@nic.surfnet.nl

Nordic TEX Users Group
Short name: NTUG
Target languages: Scandinavian languages
Website: http://www.ifi.uio.no/~dag/ntug/
Contact addresses:
Policy matters: President Dag Langmyhr
University of Oslo
PO Box 1080 Blindern
N-0316 Oslo
Norway
email: dag@ifi.uio.no
phone: +4722852450 *fax*: +4722852401
Mailing list: nordictex@ifi.uio.no

Polska Grupa Uzytkowników Systemu TEX
Short name: GUST
Target language: Polish
Website: http://www.gust.org.pl
Email: sekretariat@gust.org.pl
Contact addresses:
Policy matters: President Andrzej Borzyszkowski
Instytut Matematyki Uniwersytetu Gdanskiego
ul. Wita Stwosza 57
80–952 Gdańsk
Poland
email: prezes@gust.org.pl
Finance / member admin.: Treasurer Jolanta Szelatyńska
email: sekretariat@gust.org.pl
Publication: GUST Bulletin
Editor: Tomasz Przechlewski
Editor email: ekotp@univ.gda.pl
Mailing list: gust-l@man.torun.pl

Deutschsprachige Anwendervereinigung TEX e.V.
Short name: DANTE e.V.
Target language: German
Website: http://www.dante.de
Email: dante@dante.de
Contact addresses:
Policy matters: President Thomas Koch
Postfach 101840
D-69008 Heidelberg
Germany
email: dante-v@dante.de
phone: +49622129766 *fax*: +496221167906
General matters: Secretary Günter Partosch
Finance / member admin.: Treasurer Horst Szillat
Publication: Die TEXnische Komödie
Editor: Gerd Neugebauer
Editor email: dtk-redaktion@dante.de
Mailing list: tex-d-l@listserv.gmd.de

Lietovos TEXó Vartotojø Grupë
Contact addresses:
Policy matters: President Vytas Statulevicius
Akademijos 4
LT-2600 Vilnius
Lithuania

email: vytass@ktl.mii.lt
phone: +3702359609 *fax*: +3702359804

TeXCeH
Website:
http://vlado.fmf.uni-lj.si/texceh/texceh.htm
Contact addresses:
Policy matters: President Vladimir Batagelj
Jadranska 19
SI-61111 Ljubljana
Slovenia
email: Tex.Ceh@fmf.uni-lj.si

Estonian User Group
Contact addresses:
Policy matters: President
strophysical Observatory, Toravere
Enn Saar, Tartu
EE 2444 Estonia
email: saar@aai.ee

Danish TEX Users Group
Short name: DK-TUG
Target language: Danish
Website: http://sunsite.dk/dk-tug/
Email: dk-tug-bestyrelse@sunsite.dk
Contact addresses:
Policy matters: President Palle Jørgensen
Børglumvej 2F, v. 633
DK-8240 Risskov
Denmark
email: dk-tug-formand@sunsite.dk
phone: +4589379633
General matters:
Finance / member admin.: Treasurer Arne Jørgensen
Kollegium 5, 2., v. 222
Universitetsparken
DK-8000 Aarhus C
Denmark
email: dk-tug-kasserer@sunsite.dk
phone: +4589427222
Mailing list: dk-tug@sunsite.dk
Subscribe at: dk-tug-subscribe@sunsite.dk

TEX México
Website: http://ciencia.dcc.umich.mx/tex/
Email: tex@ciencia.dcc.umich.mx
Contact addresses:
Policy matters: President
Rayon No. 523, Centro 58000
Morelia, Michoacan
Mexico
email: tex@ciencia.dcc.umich.mx
phone: +52143128724 *fax*: +52143173945

Chinese TEX Users Group
Target language: Chinese
Website: http://www.rons.net.cn
Contact addresses:
Policy matters: President Hong Feng
RON's Datacom Co., Ltd.
79, DongWu Ave.,
Wuhan, Hubei Province
430040 China P.R.
email: info@mail.rons.net.cn
phone: +862783222108 *fax*: +862783222108

Grupo de Utilizadores de TEX
Short name: GUTpt
Target language: Portuguese
Website: http://hilbert.mat.uc.pt/~GUTpt/
Email: GUTpt@hilbert.mat.uc.pt
Contact addresses:
Policy matters: President Pedro Quaresma de Almeida
Coimbra University
Dep. Matemática, Largo D.Dinis
Apartado 3008, 3001-454 Coimbra
Portugal
email: pedro@mat.uc.pt
phone: +351239791181
Mailing list: gutpt@hilbert.mat.uc.pt

UK TEX Users' Group
Short name: UK-TuG
Target language: British English
Website: http://uk.tug.org
Email: uktug-enquiries@tex.ac.uk
Contact addresses:
Policy matters: Chairman Philip Taylor
email: P.Taylor@rhul.ac.uk
General matters: Secretary Dr James Foster
email: J.Foster@sussex.ac.uk
Finance / member admin.: Treasurer Peter Abbott
1 Eymore Close
Selly Oak, Birmingham B29 4LB
United Kingdom
email: PeterAbbott.Eymore@btinternet.com
Publication: Baskerville
Editor email: Baskerville@tex.ac.uk

ITALIC
Target language: Irish
Contact addresses:
Policy matters: Peter Flynn
University College
Cork
Ireland
email: pflynn@www.ucc.ie
Mailing list: Italic-L@listserv.heanet.ie

info
Agenda

Jules van Weerden

Feb 20–23
DANTE 2002 and 26th Annual Meeting of DANTE e.V. University of Erlangen-Nürnberg, Erlangen, Germany. More information at http://www.dante.de/dante2002/.

April 29 – May 3
EuroBachoTeX 2002 – 13th Annual Meeting of the European TeX users and 10th Polish TeX Conference will take place in Bachotek (Brodnica Lake District), Poland. The theme of the conference is this year "TEX and beyond". For more information, see the web site at http://www.gust.org.pl/EuroBachoTeX/.

22 Mei
29e NTG-DAG te Arnhem. Hotel Haarhuis. Het programma is nog niet bekend

May 27 – Jun 7
The Rare Book School announces the schedule of its May/June 2002 courses, now available on their web site at http://www.virginia.edu/oldbooks/rbs/schedule.html.

June 3–7
Short course entitled "Introduction to the History of Typography" by Archie Provan to be presented at The Rare Book School. See http://www.virginia.edu/oldbooks/rbs/schedule.html for more details.

July 15–19
The Rare Book School first summer session course: "Electronic Texts in XML," taught by David Seaman, with Christine Ruotolo and Matthew Gibson (all of the University of Virginia Electronic Text Center). For more information see http://www.virginia.edu/oldbooks/rbs/schedule.html.

July 15–19
The Rare Book School first summer session course: "Printing Design and Publication," taught by Greer Allen. For more information see http://www.virginia.edu/oldbooks/rbs/schedule.html.

September 4–7
The 23rd Annual Meeting and Conference of the TeX User Group, Stand up and be proud of TeX!, will be held in Trivandrum, Kerala, India. For more information, please visit http://www.tug.org.in/tug2002/

Algemeen

Mar 13 – Apr 23
San Diego State University, Malcom A. Love Library, San Diego, CA May 7 – Jun 27 San Francisco Public Library, San Francisco, CA "The Best of the Best" – a traveling exhibition of the Guild of Book Workers. This is a juried exhibition presenting 35 of the best works from the last part of the century.

The exhibition can be seen online at: http://palimpsest.stanford.edu/byorg/gbw.

nieuws

Een uittreksel uit de recente bijdragen in het CTAN archief

Piet van Oostrum

abstract

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

keywords

TEX, LATEX, packages, CTAN, Type1, fonts, PDF, pstricks, Postscript

PDFLATEX

PDFTEX, het programma waar Hàn Thế Thành mee bekend geworden is, en dat voortgekomen is uit het werk voor zijn dissertatie, heeft langzamerhand een belangrijke plaats ingenomen in de TEX wereld. Het zou best kunnen zijn dat het intussen meer gebruikt wordt dan de ouderwetse TEX, dat alleen DVI files kan produceren. Zeker voor het gebruik op het WWW is PDF een niet meer weg te denken bestandsformaat. Ook in de drukkerswereld wordt steeds meer overgegaan op PDF, hoewel hier af en toe nog grote problemen opduiken.

De LATEX versie van het programma, PDFLATEX, kan waarschijnlijk op een nog grotere populariteit beroep doen, naast Context natuurlijk. Een van de dingen die de PDF versies van de gewone TEX onderscheiden is dat het invoegen van plaatjes hierbij in de praktijk meer mogelijkheden biedt. Weliswaar is de traditionele TEX op zich helemaal niet geinteresseerd in plaatjes. Het gebeurt met het `\special` commando en TEX bemoeit zich niet met wat hierin staat; de verwerking gebeurt door externe dviprocessors zoals `dvips`. Echter in de praktijk betekent dit dat men zich meestal beperkt tot EPS bestanden. Op sommige PC systemen kunnen ook nog BMP of andere Windows formaten gebruikt worden, op Mac systemen vaak ook nog TIFF, maar de uitwisseling tussen verschillende systemen wordt dan al snel een probleem.

PDFTEX daarentegen weet wel degelijk intern iets over plaatjes en kan o.a. PDF, PNG en TIFF aan (afhankelijk van de versie). De nieuwste versies kunnen zelfs afzonderlijke pagina's uit een PDF bestand halen en die afzonderlijk invoegen.

Pdfpages

Pdfpages is een LATEX package, dat deze faciliteit gebruikt om afzonderlijke pagina's of series pagina's uit een PDF bestand in een LATEX bestand op te nemen. Hierbij kan de pagina van de uitvoer geheel bestaan uit een pagina van het PDF bestand (dus zonder extra headers en footers), of ze kunnen gewoon binnen de bestaande pagina opgenomen worden. Ook allerlei rangschikkingen zoals 4 op een pagina kunnen zeer eenvoudig gerealiseerd worden. Daarmee krijgt pdfpages de mogelijkheden die in de Postscript wereld traditioneel door programma's zoals `nup`, `pstops` e.d. verzorgd worden. In Context is hetzelfde mogelijk met het programma `texexec`.

Met pdfpages is het echter mogelijk flexibeler te werk te gaan dan bijvoorbeeld met `pstops`. Immers bij `pstops` kies je in principe voor één bepaalde rangschikking (bijvoorbeeld 2 rijen van 2) en dat geldt dan voor het hele document. Met pdfpages kan voor het ene deel een andere rangschikking genomen worden dan voor het andere deel en bovendien kan allerlei extra informatie worden toegevoegd, hetzij op de pagina zelf of op tussenliggende pagina's.

Ook zijn er opties om links naar de pagina's op te nemen wat handig is voor index-achtige constructies. Bovendien kunnen pagina's uit verschillende documenten opgenomen worden.

Met behulp van pdfpages is het simpel op een soort pdftopdf programma (bijvoorbeeld als shell script) te maken. Net als met `texexec` overigens.

Een (kleine) tekortkoming is dat op dit moment hyperlinks in de opgenomen pagina's niet in het hoofddocument actief zijn. Dit is een beperking van `pdftex`.

Pdfpages is te vinden op CTAN in `macros/latex/contrib/supported/pdfpages`.

CM-Super

Cm-super is een grote verzameling Type1 fonts. Het bevat de EC en TC fonts die belangrijk zijn voor het gebruikt van

correcte afbreekpatronen voor West-Europese talen. Ook Cyrillische fonts zijn erin opgenomen. En verder nog want minder gebruikte fonts.

De aanwezigheid van Type1 versies van deze fonts is belangrijk. Het gebruik van Knuth's CM fonts (die algemeen als default voor TEX en zijn afgeleide varianten worden gebruikt) heeft als nadeel dat hier geen letters met diacritische tekens voorhanden zijn. Weliswaar kunnen deze opgebouwd worden uit afzonderlijke latijnse letters en accent-tekens, maar ze kunnen dan niet in afbreekpatronen gebruikt worden. In het Nederlands is daar misschien nog wel mee te leven, maar in het Duits, de Scandinavische talen en de Oosteuropese talen is dat een ramp. Ook het Frans en het Spaans lijden hieronder.

De EC (European Computer Modern) fonts zijn speciaal hiervoor ontwikkeld en bevatten dan ook de meeste tekens die in Europa gebruikt worden, althans in talen met een latijns schrift. Echter deze fonts waren tot voor kort alleen in Metafont formaat beschikbaar, tenzij men een commerciële versie kocht. De Metafont versies, die dus bitmaps opleveren hebben echter het nadeel dat ze slechte schermversies opleveren, bijvoorbeeld bij het gebruik van PDF als uitvoerformaat. Bovendien kan bij het afdrukken op een printer met een andere resolutie dan de gegenereerde bitmaps de kwaliteit ook sub-optimaal zijn. En zeker als de documenten op het Internet gepubliceerd worden is dit ongewenst.

Er waren enkele oplossingen voor dit probleem voorhanden die echter niet geheel voldeden. Zo waren er virtuele fonts (LATEX packages `ae` en `zefonts`) die de EC tekens opbouwden uit CM tekens. Door dan de Type1 CM fonts te gebruiken krijgt men een goede kwaliteit uitvoer, en doordat in TEX de EC fonts gebruikt worden is het probleem van de afbreekpatronen opgelost. Het is echter niet mogelijk alle tekens uit de EC en bijbehorende TC verzameling op deze manier op te bouwen. Het `ae` package laat sommige symbolen weg, `zefonts` vult ze aan met symbolen uit de Adobe standaard fonts (Times Roman e.d.) wat ook niet erg fraai is.

De komst van de CM-Super fonts lost deze problemen op. Het hele pakket kost wel zo'n 60 MB aan schijfruimte.

De naam CM-Super geeft aan dat elk Type1 font uit deze verzameling de tekens bevat uit een aantal TEX fonts. Verschillende encodings zijn samengevat in één PFB file. Volgens de bijgeleverde documentatie zijn er 415 Type1 fonts die tesamen 2422 TEX fonts omvatten. Dit heeft wel tot gevolg dat de PFB files vrij groot zijn. Door font-subsetting kan de grootte van de gegenereerde Postscript of PDF files echter beperkt worden. Hierbij komt echter een bug in `dvips` versie 5.86 in actie. Deze kan omzeild worden door de optie `-j0` mee te geven die de font-subsetting uitzet. Hierbij worden natuurlijk grote Postscript files gegenereerd. In versie 5.86d van `dvips` is deze bug opgelost.

De installatie van CM-Super is niet moeilijk en bestaat voornamelijk uit het copieren van files naar de juiste directories, het aanpassen van het `updmap` script (althans in teTeX), en het runnen hiervan. Bij het gebruik van `dvipdfm` moeten nog enkele handelingen uitgevoerd worden. Tenslotte moet de filename database herbouwd worden. Het zou echter te hopen zijn dat deze verzameling binnenkort in TeXlive opgenomen wordt. Een probleem kan hierbij zijn dat TeXlive nu al nauwelijks meer op een CDROM past.

Voor MikTEX en andere systemen die niet het `updmap` script hebben volgt hier de beschrijving van de installatie[1]

- De wijzigingen kunnen gemaakt worden in de texmf boom, maar het verdient de voorkeur dit te doen in de `localtexmf` boom. In dat geval moeten de ondergenoemde files die aangepast moeten worden, eerst naar deze boom gekopiëerd worden.
- Kopiëer de pfb, map en enc files zoals aangegeven in de handleiding. Kopiëer de afm files alleen indien nodig (TEX zelf gebruikt deze niet).
- Voeg toe in `dvips/config/config.ps`:

```
p +cm-super-t1.map
p +cm-super-ts1.map
p +cm-super-t2a.map
p +cm-super-t2b.map
p +cm-super-t2c.map
p +cm-super-x2.map
p +cm-super-t4.map
```

- Voeg toe in `pdftex/pdftex.cfg`:

```
map +cm-super-t1.map
map +cm-super-ts1.map
map +cm-super-t2a.map
map +cm-super-t2b.map
map +cm-super-t2c.map
map +cm-super-x2.map
map +cm-super-t4.map
```

- Voeg toe in `dvipdfm/config/config`:

```
f cm-super-t1.map
f cm-super-ts1.map
f cm-super-t2a.map
f cm-super-t2b.map
f cm-super-t2c.map
f cm-super-x2.map
f cm-super-t4.map
```

1. Gepost door Vladimir Volovic op fido7.ru.tex en overgenomen op comp.text.tex door Vlado Handziski.

- Herbouw de filename database via het Start menu (MikTEX maintenance).

De CM-Super fonts zijn gemaakt met behulp van het programma `textrace` [1]. Daarna zijn ze geoptimaliseerd met behulp van het programma Fontlab.

De fonts kunnen worden gedownload van CTAN in de directory `fonts/ps-type1/cm-super`.

Pdftricks

Het bekende latex package `pstricks` haalt allerlei trucken uit met Postscript die in normaal TEX niet mogelijk zijn. Bijvoorbeeld het tekenen van diagrammen, pijltjes van een deel van een pagina naar een ander deel.

Omdat `pstricks` pure Postscript commando's in de DVI file zet (met behulp van `\special`), kan dit niet in de standaard PDFTEX gebruikt worden[2]. Om PDF files te genereren moet met dus terugvallen op `dvips`, gevolgd door een of andere distiller, bijvoorbeeld `ghostscript`.

Pdftricks is een package dat probeert om het gebruik van `pstricks` met PDFTEX toch mogelijk te maken. Hiertoe vangt het package de gebruikte `pstricks` constructies op, schrijft deze naar een aparte file en roept een extern programma aan om dit gedeelte naar een PDF file te vertalen die dan ingevoegd wordt. Dit proces gebeurt zonder dat de gebruiker hier veel van merkt.

Om dit te kunnen doen moet een speciale extensie aanwezig zijn en geactiveerd zijn, de z.g. `\write18` methode. Dit is een uitbreiding van TEX waarmee externe commando's uitgevoerd kunnen worden. Deze extensie is in alle versies van TEX aanwezig die op `web2c` zijn gebaseerd (o.a. `teTeX` en `fptex`), maar staat meestal uit. De reden dat dit uitstaat is dat het gevaarlijk is om onbekende TEX documenten te verwerken met de optie aan, want deze zouden gevaarlijke commando's kunnen uitvoeren. Voor het gebruik van `pdftricks` is het nodig deze optie aan te zetten, bij voorkeur via de commandoaanroep, dus niet permanent.

Het is zelfs mogelijk om `pstricks` te gebruiken zonder de optie aan te zetten, maar dan moet men TEX of LATEX twee keer draaien met ertussen een ander programma `pst2pdf`.

De documentatie laat zich er niet direct over uit maar ik denk dat niet alle `pstricks` constructies door `pdftricks` verwerkt kunnen worden. In ieder geval kan men TEX documenten met `pstricks` constructies niet zonder wijzigingen met `pdftricks` gebruiken; er zullen enige wijzigingen gemaakt moeten worden. Zo moeten bijvoorbeeld alle `pstricks` constructies binnen een `pdfpic` omgeving geplaatst worden. Toch kan `pdftricks` voor degenen die graag van de faciliteiten van PDFTEX gebruik maken een welkome aanwinst zijn.

Pdftricks kan gevonden worden op CTAN in `macros/latex/contrib/supported/pdftricks`.

Tot Slot

Een kleine selectie van wat andere aanwinsten of vernieuwingen van de laatste tijd:

Euro-ce Een Metafont definitie van het Euro-teken in verschillende smaken. In `fonts/euro-ce`.

Gloss Een LATEX package voor het maken van glossaries. In `macros/latex/contrib/supported/gloss`.

Eurosans Een LATEX package om de Adobe Euro symbolen te gebruiken. Adobe heeft een aantal gratis fonts ter beschikking gesteld die alleen het Euro symbool bevatten in verschillende uitvoeringen. Deze fonts kunnen gratis bij Adobe gedownload worden, maar mogen helaas niet verder verspreid worden en kunnen dus niet in bijvoorbeeld TeXlive opgenomen worden. In `fonts/euro/latex/eurosans`.

Ttf2tex Shell scripts om het gebruik van TrueType fonts te vergemakkelijken. Het genereert alle files die TEX nodig heeft. In `support/ttf2tex`.

Web-O-Mints Freeware fonts met verschillende ornamentale tekens, o.a. voor randen. In `fonts/webomints`.

Referenties

[1] Péter Szabó, *Conversion of TEX fonts into Type1 format*. Proceedings of the EuroTEX conference, 2001.

2. Het commerciële VTeX kan dit wel omdat het een Postscript interpreter ingebouwd heeft.

aktueel

Patenten, copyright en ‘intellectual property’

Siep Kroonenberg
siep@elvenkind.com

abstract

In dit stuk wordt aandacht gevraagd voor software patenten en andere aanslagen op onze elektronische vrijheid.

keywords

Software patenten, WIPO, EPO, standaarden, Europa

In Europa zijn software patenten officieel niet mogelijk. We willen ook geen software patenten. Ideeën zijn er om uit te wisselen, niet om te monopoliseren. Om met Knuth te spreken: ‘There are better ways to earn a living than to prevent other people from making use of one’s contributions to computer science’[1]. En vrije software zoals TEX en Linux kan geen gepatenteerde technieken implementeren zonder op te houden vrij te zijn.

In de VS zijn patenten al veel langer een probleem. Eén van de eerste software patenten waar veel ophef over is geweest is het Unisys patent op het gif formaat en op LZW compressie. Dit formaat was al jaren in gebruik voordat Unisys op de proppen kwam met hun aanspraken. Dit zelfde LZW patent was overigens ook toegekend aan IBM. Lees http://lzw.info over de tragikomische geschiedenis van dit patent.

Nu is het EPO (European Patent Office) is in hoog tempo patenten aan het toekennen die verdacht veel op software patenten lijken. Kijk eens op http://swpat.ffii.org/vreji/pikta/mupli/index.en.html voor een lijst van door het EPO toegekende patenten.

Hoe belachelijk een patent ook is, aanvechten kost geld. Patenten werken sterk in het voordeel van grote bedrijven, en nog meer in het voordeel van juristen die geld verdienen met patentzaken. Patenthouders kiezen vaak voor een schikking inplaats van een rechtszaak opdat ze het vermeende patent niet hoeven te onderwerpen aan een serieus onderzoek.

Het EC voorstel over software patenten

Op 20 februari is een Europees voorstel gepubliceerd over patenteerbaarheid[2] van computer-gerelateerde uitvindingen: ‘Proposal for a DIRECTIVE OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the patentability of computer-implemented inventions’, http://europa.eu.int/comm/internal_market/en/indprop/com02-92en.pdf.

Hierin wordt geadviseerd tegen patenten op computer programma’s ‘as such’ maar een patent kan wel worden toegekend aan een technische vinding waaraan computers te pas komen.

Het voorstel distantieert hierin van de praktijken van het EPO: ‘It should be noted that the proposal has not followed the practice of the EPO in permitting claims to computer program products either on their own or on a carrier, as this could be seen as allowing patents for computer programs “as such”’.

Je zou wel wat voorbeelden willen zien van welke software wel en welke niet patenteerbaar zou zijn. De Eurolinux Alliantie (www.eurolinux.org) is in elk geval zeer kritisch ten opzichte van dit voorstel.

Het laatste woord is er nog niet over gesproken.

Andere ontwikkelingen

□ Het internet is gebouwd op open standaarden. Niettemin zijn er initiatieven om daar een einde aan te maken; zie www.w3.org, ‘Patent Policy’. Het doel van het W3C is misschien om de schade van patenten te minimaliseren, maar natuurlijk willen we die patenten helemaal niet.

□ WIPO is een internationaal verdrag voor copyright bescherming; zie http://www.wipo.org/treaties/ip/copyright/. Dit moet nog door de individuele landen van de EU geratificeerd worden. Hierin zitten clausules die het illegaal maken om ‘digital rights management’ te omzeilen. Dit verdrag heeft een soortgelijke strekking als de Amerikaanse DMCA, waaraan Napster ten onder is gegaan. Er zijn ook pogingen geweest om op grond van de DMCA research te onderdrukken die zwakheden van copy-protektie technieken aan de kaak stelt (http://www.eff.org/IP/DMCA/Felten_v_RIAA/). Dit lijkt met een sisser af te lopen. Laten we hopen dat het WIPO verdrag niet soortgelijke gevolgen krijgt.

1. Uit het voorwoord van ‘Art of Computer Programming, Volume 3: Sorting and Searching’, Donald E. Knuth

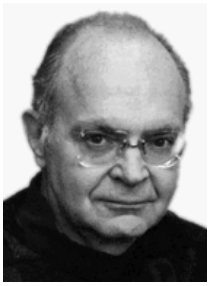
2. Het Nederlands woord is octrooi, maar omdat het Europese wetgeving betreft en het internationale woord ‘patent’ goed is ingeburgerd gebruikt ik toch maar het woord ‘patent’.

□ Daarnaast wordt gewerkt aan internationale 'harmonisering' van wetten, ook op het gebied van Intellectual Property. De Haagse Conventie gaat hierover; zie `http://www.inta.org/europe/hague.shtml`. Sommigen maken zich zorgen dat dit ons blootstelt aan slechte wetten elders, lees 'in de VS'.

Ik ben geen jurist en kan niet garanderen dat ik het allemaal goed heb begrepen, maar er is zich een onmiskenbare trend aan het aftekenen. Het is dringend nodig dat deze zaken meer aandacht krijgen.

Tenslotte, bij wijze van illustratie wat er zou gebeuren als vrije software ineens niet meer vrij zou zijn, een stuk dat we eerder in MAPS 20 hebben afgedrukt. Op 1 april 1998 bracht Richard Kinch bij wijze van één-april grap een verhaal in omloop dat Microsoft de rechten op TEX zou hebben opgekocht:

MS-TeX



PALO ALTO, CALIFORNIA, USA (CNEWS/MSNBC) In a major move into the scientific publishing market, Microsoft Corporation announced today that it has purchased all rights to the computer language and document compiler known as TeX (pronounced, "tech"), and plans a major new product line based on the 20-year-old software.

Stanford Professor Donald Knuth (pronounced, "kah-nooth"), the author of the widely-used TeX software, in a joint press conference at the university campus with Microsoft Chairman Bill Gates, acknowledged that the two had been negotiating for some months. "I felt that two decades of TeX in the public domain was enough. I am reasserting the copyright to my original work in TeX. Microsoft will carry the ball now, and I can get back to my computer science research." Knuth acknowledged he was paid a "seven-figure sum" from Microsoft, which he will use to finance his work on a project he has code-named "Volume 4".

At the press conference, Microsoft chairman Bill Gates said the acquisition was "the kind of cooperation between academia and industry that builds prosperity for both." He added that TeX would "finally give Microsoft a foothold in mathematical desktop publishing" that has eluded the software giant since its founding. Drawing gasps of surprise from the college audience, Gates asserted that "TeX will soon be biggest jewel in the Microsoft crown."

Apparently the jewel metaphor will include a hefty, unavoidable price tag for future TeX users. Gates outlined plans whereby all existing TeX compilers would be phased out, to be replaced by a new Microsoft master implementation written in C++. Beta versions for public testing on Windows 95 and NT platforms are expected in late 1998, issuing from a new 205-programmer project laboratory at Microsoft's Redmond campus. Microsoft TeX for other platforms, such as Unix workstations, will follow at an as-yet unspecified date. According to Gates, "the master TeX from Microsoft will ensure that the incompatibilities across platforms are once and for all eliminated." TeX software is widely used due its portability, although variations among operating systems have been troublesome due to uncoordinated development.

Unlike the technical aspects of the project, Gates explained that pricing for Microsoft TeX has already been firmly set. The single-user retail product is expected to have a street price of about $600 and consist of three CDs. When heckled by an graduate student complaining about a high price for a formerly free product, Gates seemed startled, explaining that a "student edition at $299 is likely" and that "Microsoft will use the revenue to make TeX better."

Most current users of TeX have paid nothing for their implementations, derived from Professor Knuth's formerly-free work. Before leaving the podium, Gates made a final comment that "TeX hasn't changed in years. What kind of a product can that be?", and then handed the microphone to an assistant, introduced only as the project leader for Microsoft TeX.

The assistant displayed an overhead presentation using the current test version of Microsoft TeX. Equations and tables could be seen dissolving into each other in a morphing action between frames. "No one has ever done that with TeX," Gates announced from an audience seat at one point. "It's the kind of sizzle that can really enliven a dull paper at an academic conference." Some onlookers were not convinced, especially when the program crashed midway through the demonstration, resulting in a five-minute delay while Windows 95 was restarted. Microsoft technicians later blamed a third-party display driver.

The impact on the large base of existing TeX users was unclear. During a question-and-answer period, Gates said that the "TeX" trademark would be registered as the exclusive property of Microsoft, and could not appear in any

competitive or free software. “We are granting of our own good will until the 3rd quarter of 1998, free use to any existing TeX vendors or public-domain authors. That’s plenty of time for an orderly phase-out and change-over to Microsoft TeX, or no TeX at all. After that, our legal department will be contacting them.”

A Microsoft attorney added that some of the project personnel would be dedicated to searching the Internet to find non-Microsoft TeX software. “Archives and collections of TeX-related programs will not be permitted. The standards must be enforced, or they become meaningless. We are rescuing a fine piece of work from being diluted into worthlessness. You would not believe the number of programs that have been based on TeX without any central, controlling authority. We will stop this infringement.”

Some large organizations dependent on TeX were stunned by the announcment and had not yet formed plans for dealing with the change. At the American Mathematical Society, whose publications largely depend on TeX for typesetting, editor Barbara Beeton was incensed. “I can’t believe Don [Professor Donald Knuth] sold us out like this. We should have never based a publishing enterprise of this scope on so-called public-domain software. What were we thinking?” Publication schedules for the rest of 1998 were on hold, and journal editors scrambled to reassure their authors that deadlines would not slip more than a few months.

Certain small businesses are also expected to feel the impact of the Microsoft ownership of TeX. Palo Alto restaurant owner Wu Chen appeared unhappy at the news, stating that “for ten year I print new menu every day with TeX, now I will pay big time.” He displayed a crumpled, grease-spotted take-out flyer, and with tears in his eyes explained how multiple columns, exotic typefaces, and daily price changes could all be printed by TeX in a multi-lingual format. “In Wordperfect this would be a long journey.”

Commercial vendors of TeX software stand to lose everything in the face of the new Microsoft monopoly. While most derivatives of TeX were freely published, several companies had made a business of publishing proprietary versions. One anonymous source from a leading TeX firm said that “publishing TeX was a gold mine while it lasted, and the Internet let us mine it deeper and deeper. Now this is a cave-in right on our heads. TeX was a monumental work of beauty and utility, freely given to the world by one of the finest and most generous minds of the 20th century. Now it belongs to a lucky dropout. We’re finished.”

STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

DONALD E. KNUTH
PROFESSOR EMERITUS OF THE ART OF
COMPUTER PROGRAMMING

26 Nov 2001

Dear Siep Kroonenberg,

Congratulations on a most beautiful design and production of the EuroTEX 2001 Proceedings!

My wife and I both love its look and feel.

Sincerely

Don Knuth

Patrick Grundlach

fonts
meta-euro

The new currency, the euro, has not only brought new coins and bills to a lot of people in europe, but it has also introduced a new symbol, which looks like this: €. It is not just a capital C with two horizontal bars. There are angles at the left and right side of the bars as well as at the upper left corner of the C. Even the opening angle of the C should be the same in all the symbols. The european union (eu) tries to guide the font designers and provides an official version of the symbol on their webpage. It looks like a geometric construction that can be drawn with a ruler and a compass. And—since we are all used to solve such tasks with a computer—this can certainly be done with METAPOST. METAPOST is a companion to Don Knuth's METAFONT and outputs encapsulated postscript (eps) rather than fonts coded in bitmaps. METAPOST was written by John Hobby. It uses a language that is very close to the one described in the METAFONT book. METAPOST (as well as the changes to METAFONT) is described in the METAPOST user's manual (available as `mpman.ps` on ctan or your local TEX installation) It is advisable, if you plan to program in METAPOST, to have the users manual available as well as Knuth's METAFONT book. But the user's guide will probably suffice at the beginning. There is also a nice introduction to METAPOST in the MetaFun handbook. MetaFun is a macro package that is designed to work well with the TEX macro package ConTEXt. But beginners should be aware of the differences of plain METAPOST and the MetaFun extensions when reading the MetaFun manual.

But back to the euro symbol. I have stated that there is an official version provided by the eu, that looks as it can be constructed by geometrical means. It is depicted in figure 1. You will probably notice that there are some angles and some distances shown. But if you look close enough, there are also some sizes unknown. For example it is not clear how far the horizontal bars stick out to the left side. There is another construction by the european central bank (ecb). You can see their version in figure 2. This construction is about the same as the one mentioned before, but it adds some additional values. You can see that the horizontal bars now stick out a well defined amount. Most other constructions in the media are more or less exact copies of these two. The eu one seems to be in favour. For the rest of this article that one will be used.

I have mentioned earlier that there are some weaknesses of the official construction, that makes it impossible to make an exact reproduction without measuring distances by hand. The problems are

1. the length of the horizontal bars is unknown
2. the shape of the surrounding "circle" is not clear. Only the height and the relative position of the horizontal bars is defined
3. the thickness of the "circle" is unknown
4. the thickness of the horizontal bars is unknown
5. and, not really a problem, the lables (x) on both angles are misleading and do not seem to fulfill a real purpose.

Perhaps these points are unclear on purpose, so the font designer has some freedom to let the euro symbol fit nicely into a given font. Whatever the reason is, we have to make assumptions about the unclear parts, in order to be able to express the shape in terms of a METAPOST program. I will, for the sake of clearness, assume that the surrounding shape is a circle. While it looks like it in the eu construction, it looks a bit more squeezed in the

ecb one. In the official construction there is a variable x that seems to be some kind of a unit. The diameter of the circle is $11x$ if the horizontal bars and the circle have a width of $1x$.

I will now show the METAPOST code. See below for an explenation on how to use it in LaTeX or ConTeXt.

```
1   prologues    := 2 ;
2   beginfig(1);

3   def tand expr x = (sind (x) / cosd (x)) enddef;

4   path unitcircle; unitcircle:=fullcircle scaled 2;
5   %unitcircle has radius of 1
```

Every METAPOST picture is enclosed in `beginfig`/`endfig`. The number in the parentheses denotes the file extension. Say, for example, you have named the METAPOST-file maps-euro.mp, the figure gets written as maps-euro.1. In plain METAPOST there is no function to calculate the tangens. Since it is needed below, I provide a definition here. I also provide a circle with radius 1. I could of course have used the (predefined) fullcircle and just multiply all values by two.

```
6   boolean show_dots, show_lines,show_labels ;

7   show_lines  := false;
8   show_dots   := false;
9   show_labels := false;
```

These lines are only for debugging purpose. The euro-symbol, when used in text, should not have any labels or lines. But while constructing the symbol I would like to see if the points are in the right place. And, of course, it should make it easier for the reader to follow my thoughts as expressed in this code. Just set these variables to true if you want to see more output.

```
show_lines  := true;
show_dots   := true;
show_labels := true;
```

What follows is declaration of some variables. In METAPOST you have to tell the system whether your variable is a path, a pair or ....

```
10  path inner, outer; % the big circle: inner and outer part
11  path hbar;         % horizontal bar
12  path a[];          % aux paths for intersections
13  path clippath;     % path for clipping
14  pair topbarleft, bottombarleft;
15  pair o[];          % official points
16  pair c[];          % clip points
17  picture cp;        % currentpicture for clipping

18  x :=1cm;
19  radius:=5.5x;
20  topbarlength := 10x;
21  thickness:=1x;
```

These are the only user-changable parameters I provide. In a typical METAPOST program, there are probably more values that can be influenced. x is our unit as given in the

official construction. It determines the overall size of the symbol. But since the picture is scalable, it really does not matter too much to what value you set it.

Now I will determine the points that are needed to draw the symbol. In the official construction, the points o1, o2 and o3 are given, as well as the right slant. Alpha is the angle between the horizontal line and the right slant.

```
22  o1 = (0 ,-radius-.5thickness);
23  o2 = dir  40 * (radius-.5thickness);
24  o3 = dir -40 * (radius-.5thickness);
25  alpha := angle (o2-o1);

26  a1 := o1 -- o2;      % the right slant
27  a2 := origin -- o3;  % the lower 40deg. angle
```

Now, let us actually draw something. Start with the circle. This circle is a fullcircle. The unwanted part at the right side will be clipped off later. After drawing the circle, we determine the shape of the horizontal bars. There again, we draw more than we need and clip the unwanted part off later.

```
28  draw unitcircle scaled radius
29        withpen pencircle scaled thickness;

30  hbar := unitsquare xscaled topbarlength yscaled thickness
31                  slanted (1/tand (alpha));

32  % top bar lrcorner:
33  c3 := ((-infinity,0.5x) -- (infinity , 0.5x)) intersectionpoint a1;

34  topbarleft    := (xpart c3 - topbarlength , 0.5x);
35  bottombarleft := (xpart c3 - topbarlength ,-0.5x - thickness);

36  fill hbar shifted topbarleft;
37  fill hbar shifted bottombarleft;
```

In line 33 we have found c3 just by asking METAPOST to find the intersection of the path a1 and an infinite long horizontal line shifted 0.5 units above the origin. We do not assign this line to a variable or even draw it. Now the shape of the horizontal bars are known, so we can draw them.

At this time everything needed is drawn. But since we have put more in the picture than necessary, some erasing must be done. There are several ways to accomplish this. You can overdraw the unwanted parts with the background color. But since the background color is not known at this time, we cannot use this method. Another solution is to clip the picture to a certain path. This is as if you take a pair of scissors and cut along the path. We use clipping now:

```
38  cp := currentpicture;

39  c2 := a1  intersectionpoint a2;
40  c4 := o2 + dir alpha *2thickness;
41  c5 := (xpart  o3, ypart lrcorner cp);

42  outer := unitcircle scaled (radius+.5thickness);
43  c1 := (o1--c4) intersectionpoint outer;
```

```
44  clippath := llcorner cp -- c5 -- o3 -- c2 -- c1 --
45              (xpart c1,ypart urcorner cp) -- ulcorner cp -- cycle ;

46  clip currentpicture to clippath;
```

What is being done here, is to find a path that can be used for clipping. It has to be as tight as possible to the symbol in order to avoid unnecessary space. Since the left side of the horizontal bars is as far as we have drawn yet, we can use the left boundary of the current picture (for example, `llcorner` denotes the lower left corner of a picture). c4 is not used for clipping, it is merely an auxiliary point to find c1.

That's it! With these few lines you draw the euro-symbol. But what is even more important, is that you have not merely presented the shape (this could be done with most drawing programs), you have expresed *how* to draw it. Contrary to the official version provided by the european union, there is no part in the construction unclear. We have even stated (although not derived from the official construction) which parts of the symbol may be changed at user option (the radius for example). This is a great advantage over using a simple (or not so simple) drawing program.

The rest of the program (except the `endfig` and `end` in the last two lines) is for debugging purpose. Remember the `show_line` (and the other variables) you have set either to true or false in the beginning? The `drawoptions` statement states a default for all following draw commands.

```
47  if show_lines:
48    inner := unitcircle scaled (radius-.5thickness);
49    pickup pencircle scaled 1pt;
50    drawoptions (withcolor .7white);

51    draw inner; draw outer;

52    draw a1 dashed evenly;
53    draw origin -- o2 dashed evenly;
54    draw a2 dashed evenly;
55    draw hbar shifted topbarleft;
56    draw hbar shifted bottombarleft;
57    draw clippath;
58  fi

59  if show_dots:
60   pickup pencircle scaled 2pt;
61   drawoptions (withcolor .5white);
62   draw origin;
63   draw c1; draw c2;
64   draw c3; draw c4;
65   draw c5;
66   draw o1; draw o2;
67   draw topbarleft;
68   draw bottombarleft;
69  fi
```

```
70  if show_labels:
71    defaultfont:="ptmr8r";
72    drawoptions (withcolor .5black);
73    label.bot("origin",origin);
74    label.bot("bottombarleft",bottombarleft);
75    label.bot("topbarleft",topbarleft);
76    label.rt ("c1",c1);
77    label.rt ("c2",c2);
78    label.rt ("c3",c3);
79    label.rt ("c4",c4);
80    label.bot("c5",c5);
81    label.bot("o1",o1);
82    label.bot("o2",o2);
83    label.rt ("o3",o3);
84  fi

85  if show_lines:
86     draw bbox currentpicture dashed evenly withcolor .7white ;
87  fi
88  endfig;
89  end;
```

The `endfig` is just the opposite of the `beginfig`. It denotes the end of a picture. The `end` instructs METAPOST to stop reading the input file and, in this case quit.

From this point on I will suppose that you have keyed in all the numbered lines into your editor and saved the file with the name maps-euro.mp. You can run METAPOST by typing the command

```
mpost maps-euro
```

into your favorite shell. METAPOST will generate a file called maps-euro.1. (For a description of the `prologues` command in line 1 see the METAPOST manual or the LaTeX-Graphics Companion.) This is a regular eps file, except when you have used fonts in your program. In our case if you turned on the `show_labels` switch. METAPOST does not include the fonts in the output file, it merely includes a reference in the eps file. So if you use a METAPOST picture with fonts, it is best to use it in conjuntion with TeX. The following simple LaTeX wrapper will suffice:

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
\includegraphics{maps-euro}
\end{document}
```

But this will work only if you have renamed maps-euro.1 to maps-euro.eps.

Using this symbol with ConTeXt is also possible. Just copy the lines starting after the beginfig and ending just before the endfig into a file, say maps-euro.tex. Enclose these lines in `\startuseMPgraphic{maps-euro}` and `\stopuseMPgraphic`. So you have (in maps-euro.tex)

```
\startuseMPgraphic{maps-euro}
def tand expr x = (sind (x) / cosd (x)) enddef;

path unitcircle; unitcircle:=fullcircle scaled 2;
```

```
...

if show_lines:
   draw bbox currentpicture dashed evenly withcolor .7white ;
fi
\stopuseMPgraphic
```

Actually you even do not need line 3 and 71, since tand is already defined in metafun (ConTEXt uses the MetaFun macro package when running METAPOST) and the default font is the current ConTEXt font. You can include the euro symbol as a picture by using `\useMPgraphic{maps-euro}` in your file or you can use it as 'character' that can be typeset in your text. It will be scaled to the current text size. What you have to do is write a small graphic-wrapper that allows scaling to the demanded height:

```
\startuniqueMPgraphic{euro}{height}
   \includeMPgraphic{maps-euro} ;
   currentpicture := currentpicture ysized \MPvar{height} ;
\stopuniqueMPgraphic
```

Now you can use this intermediate graphic to define a symbol:

```
\definesymbol[euro][\uniqueMPgraphic{euro}{height=1.5ex}]
```

To get a € you can use `\symbol[euro]` in your text.

Hans Hagen
Pragma ADE bv
pragma@wxs.nl

fonts

The euro symbol

abstract
When Patrick Gundlach posted a nice METAPOST version of the euro symbol to the ConTEXt discussion list, he added the comment "The official construction is ambiguous: how thick are the horizontal bars? How much do they stick out to the left? Is this thing a circle or what? Are the angles on the left side of the bars the same as the one on the right side? . . ." The alternative below is probably not as official as his, but permits a finetuning. You are warned: whatever you try, the euro *is* and *will remain* an ugly symbol.

We use a couple of global variables to control the euro shape within reasonable bounds. Next we define two circles. Next we define a vertical line that we use in a couple of cut and past operations. Watch how the top left point of the outer circle determines the slant of the line that we use to slice the vertical bars.

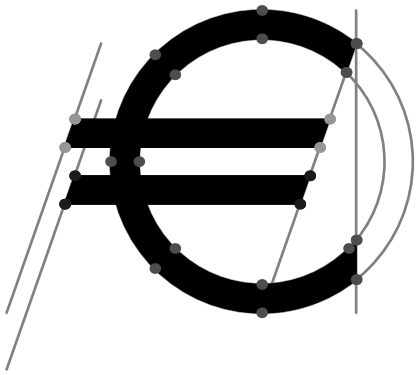
```
boolean trace_euro ; trace_euro := false ;

vardef euro_symbol = image ( % begin_of_euro

if unknown euro_radius   : euro_radius   := 2cm             ; fi ;
if unknown euro_width    : euro_width    := 3euro_radius/16 ; fi ;
if unknown euro_r_offset : euro_r_offset := euro_width      ; fi ;
if unknown euro_l_offset : euro_l_offset := euro_radius/32  ; fi ;
if unknown euro_l_shift  : euro_l_shift  := euro_r_offset   ; fi ;
if unknown euro_v_delta  : euro_v_delta  := euro_width/4    ; fi ;

save
  outer_circle, inner_circle, hor_bar,
  right_line, right_slant, top_slant,  bot_slant,
  euro_circle, euro_topbar, euro_botbar ;

path
  outer_circle, inner_circle, hor_bar,
  right_line, right_slant, top_slant, bot_slant,
  euro_circle, euro_topbar, euro_botbar ;

outer_circle := fullcircle scaled euro_radius ;
inner_circle := fullcircle scaled (euro_radius-euro_width) ;

if trace_euro : for i = outer_circle, inner_circle :
  draw i withpen pencircle scaled 1pt withcolor .5white ;
endfor ; fi ;

right_line :=
  (lrcorner outer_circle -- urcorner outer_circle)
  shifted (-euro_r_offset,0) ;

outer_circle := outer_circle cutbefore right_line ;
```

```
right_slant :=
  point 0 of outer_circle
  -- origin shifted (0,ypart lrcorner outer_circle) ;

euro_circle := buildcycle(outer_circle, right_line,
  reverse inner_circle, reverse right_slant) ;

hor_bar := (-euro_radius,0) -- (euro_radius,0) ;

top_slant :=
  right_slant shifted (-euro_radius+euro_r_offset-euro_l_offset,0) ;

bot_slant :=
  top_slant shifted (0,-euro_l_shift) ;

if trace_euro : for i = right_line, right_slant, top_slant, bot_slant :
  draw i withpen pencircle scaled 1pt withcolor .5white ;
endfor ; fi ;

euro_topbar := buildcycle
  (top_slant, hor_bar shifted (0, euro_v_delta),
   right_slant, hor_bar shifted (0, euro_v_delta+euro_width/2)) ;

euro_botbar := buildcycle
  (bot_slant, hor_bar shifted (0,-euro_v_delta),
   right_slant, hor_bar shifted (0,-euro_v_delta-euro_width/2)) ;

for i = euro_circle, euro_topbar, euro_botbar :
  draw i withpen pencircle scaled 0 ;
endfor ;
for i = euro_circle, euro_topbar, euro_botbar :
  fill i withpen pencircle scaled 0 ;
endfor ;

if trace_euro :
  drawpoints euro_circle withcolor red   ;
  drawpoints euro_topbar withcolor green ;
  drawpoints euro_botbar withcolor blue  ;
fi ;

) enddef ; % end_of_euro
```

We only set a parameter when it is not yet set. This has the advantage that we don't have to set them when we change one. This way of manipulating paths (cutting and building) does not always work well because of rounding errors, but here it does work.

```
euro_radius := 4cm ; trace_euro := true ; draw euro_symbol ;
```

For educational purposes, we have added as a bit of tracing. When enables, the euro shows up as:



Of course it would be best to define the euro as one shape, but we won't go though that process right now. By packaging the combined paths in an image, we can conveniently color the euro symbol

```
draw euro_symbol withcolor .625red ;
```

You may wonder why we both draw and fill the euro, using a pen with zero width. We've done this in order to demonstrate the `redraw` and `refill` macros.

```
redraw currentpicture withpen pencircle scaled 4pt withcolor .625yellow ;
refill currentpicture withcolor .625white ;
setbounds currentpicture to boundingbox currentpicture enlarged 2pt ;
```

design

Doing it my way: a lone TEXer in the real world

Jean-luc Doumont
JL Consulting
Achterenberg 2/10
B–3070 Kortenberg, Belgium
email: JL@JLConsulting.be

abstract

While a world-renowned standard in many academic fields, Don Knuth's much acclaimed typesetting system is almost unknown in most parts of the real world, where many a document designer has achieved professional success without ever hearing (let alone pronouncing) the word "TEX". Outside academia, the lone TEXer faces not only compatibility headaches, but also outright incomprehension from his customers, colleagues, or competitors: why would anyone want to use TEX to produce memos, two-color newsletters, full-color brochures, overhead transparencies, and other items – in short, anything but books that contain a lot of mathematics?

As a consultant in professional communication, I have been using TEX for all documents I have produced for my clients and for myself during the last ten years or so. Though it has turned out to be most successful, this approach is seen by most as a mere idiosyncrasy. And yet, the systematic use of my own TEX and PostScript programming gives me three unequalled advantages over using off-the-shelf software: I travel light, I can go anywhere I please, and I guarantee I'll get there.

Introduction

Being someone who (among other activities) produces documents for a living, I often have to discuss software issues with clients, colleagues (or competitors, as the case may be), and service bureaus or print shops. When I say I use "TEX," these people usually first ask me to repeat, then express their surprise. Some immediately dismiss it as "never heard of"; others first ask what exactly it is, before wondering why I would want to use something "that nobody else uses" instead of a WYSIWYG, "professional" application (the one *they* use, no doubt).

My using TEX (and PostScript) for virtually all documents I produce, successful as it may turn out, has puzzled many a TEX user, too. Is TEX really the best tool for documents other than long, structured, or heavily mathematical ones? Is a direct-manipulation, integrated application not better suited to producing short newsletters, graphs, or overhead transparencies? The most qualified answer is likely to be, "Well, it depends."

This paper relates the experience of a long-time lone TEXer in the real world. It explains choices, points out advantages and limitations, and draws lessons from the experience.

Approach and opportunities

The very varied documents my company produces can be grouped into two categories. Some are in-house documents, such as letters, reports, and teaching materials (handouts, lecture notes, overhead transparencies). The others are documents we create on behalf of clients, including newsletters, brochures, corporate reports, and overhead transparencies. Documents of both categories may be black-and-white, two-color, or full-color ones, printed in traditional (offset) or modern (digital) ways, in small or large runs.

To be able to guarantee the quality of the documents we produce for our clients, we have opted to give them non-editable deliverables only – in practice, paper, film, or ready-to-print electronic files (in PostScript, for example). We thus strive to preserve the quality of both the writing and the typesetting against two sources of undesirable alterations: unwise last-minute modifications by the clients themselves, often ruining a consistency they may not readily perceive, and accidental changes caused by an all-too-theoretical "seamless conversion" between platforms or versions of a given software application.

Our insistence on not giving away editable files is a tough one to maintain (admittedly, it has suffered exceptions): clients logically consider as theirs the text they have paid us to write or the format they have paid us to design and produce. Yet our name in the list of credits constitutes our best – if not our only useful – marketing tool: we cannot afford to take chances. We will, of course, ensure that what we create addresses the client's needs while attaining the quality level we strive for.

Though our approach is dictated by quality standards, not ease of production, it does simplify our work – to some extent, that is. Because we need not exchange editable files back and forth with external parties, we are free to choose not only the platform but also the software tools with which we work. More accurately, we must be able to process in-

coming ASCII files and occasional images in GIF or JPEG format, and to produce PostScript files for laser printers, imagesetters, or others still – all in all, minor constraints. Our software tools are therefore essentially limited to an ASCII editor, an image processor, and, of course, a TEX environment, which we use to produce virtually all our paper documents.

Surprise and frustration

The choice of TEX as all-purpose tool for producing very varied documents surprises those who know TEX and puzzles those who do not. Clearly, it does stem from my previous experience with TEX in an academic setting (Stanford University, which happens to be TEX's cradle), and from my technical education (applied physics) and consequent preference for analytical thinking. Admittedly, it may also partly originate in my pronounced taste for computer programming and, more generally, in my peculiar habit of wanting to (re)do things my own way. Still, the perspective provided by some ten years of successful professional practice vindicates my choice: over the years, I have satisfactorily moved to TEX for more and more types of documents.

Until I entered the "real world," I failed to realize how poorly known TEX is, at least over in Europe. It is known by the more technically minded participants at the training programs I teach in academia and national research centers, though even these people are often unclear about what TEX really is and typically confuse TEX and LATEX. In contrast, it is an unknown entity to clients who entrust us with document-creation projects, even those in research organizations, typically from the Corporate Communication or Public Relations department. These clients, used to mainstream direct-manipulation packages, have difficulties understanding the nature of TEX, the difference between TEX itself and its implementation on a specific platform, and the fact that TEX cannot do much (relatively speaking) until one programs it.

Irrelevant as it may seem, given the lack of compatibility issues in our case, the choice of TEX in an essentially non-TEX world proved a liability to our credibility. I had expected that clients who did not know TEX would be content to judge the tool by the result; I was wrong, for at least two reasons. First, most of our clients are little able to judge the quality of a typeset page; their eyes somehow seem blind to stretched out lines, uneven line spacing, and other basic points. Yet the eventual readers of the documents may well notice the difference (if unconsciously), so producing high-quality pages still matters. Second, our clients judge merit by popularity and hence distrust or even disdain what they do not know; they figure that, if they have not heard of TEX, then it cannot possibly be any good. Consequently, and independently from what they see, they are suspicious of the pages I typeset with TEX. Ironically, they seem infinitely more reassured when I simply tell them I use "a system I programmed myself" rather than "the system developed by Professor Donald E. Knuth".

Distrust of the unknown goes beyond the clients. Many of the print shops with which clients sometimes ask me to work dislike my giving them PostScript files instead of editable ones. PostScript being akin to Greek to them, they somehow feel deprived of their power and will not admit I have done half of the work for them. When anything goes wrong at any stage of the printing process, they are quick to pin the blame on me, insisting to the client that I use a software application nobody else uses, "so it's bound to create problems". (As an extreme case, one renowned print shop even resorted to this excuse after mistakenly using a Pantone color other than the one I had specified in writing.)

People who do understand what TEX is, after some explanations, wonder why on earth I prefer using such an intricate system rather than a much more user-friendly (and better known) package. At first, the advantages of TEX were so intuitively obvious to me I was unable to put them into words. Extolling the line-breaking algorithms, positioning accuracy, or logical markup was pointless: these people either saw no benefit in the features or insisted they could do all of that with their own text processor. To a point, they were right, of course, even if we all feel that "it's not the same".

A little help from my friends

At a loss for words in the defense of TEX, but convinced from experience that it suited my work so well, I set out to ask other users why they preferred TEX. I contacted the makers of my own TEX implementation, posted questions on `comp.text.tex`, and asked around. The answers were varied but can be grouped into five categories: output quality, flexibility, text-based formats, reliability, and portability.

Not surprisingly, many who answered my post on `comp.text.tex` summed it all up by saying that TEX's output "simply looks better," especially as regards mathematics. Among others, they pointed to such features as transparent use of optical sizes, better hyphenation algorithms, and line-breaking algorithms that act on whole paragraphs, not line by line.

Besides the quality of its output, users love TEX for its flexibility, allowing one to extend its capabilities almost ad infinitum. Some mentioned virtual fonts, custom hyphenation patterns, and non-European languages. Others lauded TEX's superior capabilities for floating inserts and cross-references. Following the theme of the present TUG confer-

ence, some also underlined the parallel writing of printed and HTML documentation.

Less expectedly perhaps, users praised TEX's ASCII roots. A plain-text format, they said, allows easy manipulation with any text editor, easy generation of TEX files by other applications (including preprocessors), and small files (hardly larger than the actual text) that compress well. Some also mentioned the small size of their TEX implementation, compared to that of popular text processors.

TEX's text files, batch operation, and stability over time were seen as the basis of its reliability and portability. In contrast to those of integrated applications, the TEX (source) files can be damaged only by the text editing, not by the formatting, the displaying, or any other downstream operation. Moreover, source files typeset with TEX ten years ago can be typeset again today – on any platform – to produce the exact same output, in the exact same device-independent format. TEX files, being plain ASCII, can also safely be exchanged by electronic mail across the world.

Besides the above observations, the (occasionally emotional) discussion on `comp.text.tex` as a result of my post pointed to two interesting differences. Trivial as these may seem in retrospect, they helped me understand much of the religious wars around software: they are the differences between a software tool's

- claimed and actual capabilities, and
- its potential and actual use.

Reliability and portability are typical issues that differentiate between claimed and actual capabilities. There is no intrinsic reason why an integrated application should be unreliable, although increased complexity and fast-changing versions certainly increase the odds. Similarly, there is no intrinsic reason why the claimed interchangeability between platforms and between versions should turn out untrue. Yet a pragmatic point of view suggests otherwise.

Furthermore, the fact an application offers a given feature does not imply that its users actually use it – often a question of usability. As an extreme example, any graphical application that allows one to position characters precisely anywhere on the page can produce output that matches TEX's – but at what cost? TEX, or so its users say, not only makes some operations even possible at all, it also makes many operations easier than direct-manipulation software.

Actual use is also largely affected by stability. In never-ending debates on software, protagonists who claim their own tool to be more efficient than that of others may well *all* be right, for the tool each masters best is the most efficient for him or her. Yet mastery requires time and users feel little motivated to learn to master a tool that will soon change: fast-evolving software, with pressure or even obligation to upgrade regularly, may offer ever-increasing capabilities, but discourages in-depth learning.

As a final point, the `comp.text.tex` discussion also clarified WYSIWYG (What You See Is What You Get), a concept often confused with that of direct manipulation. If WYSIWYG denotes faithful correspondence between screen view (what you see) and paper output (what you get), then some `.dvi` viewers are the closest to WYSIWYG one can get. Many people, however, actually mean that what they *do* affects the output in a way they can instantly *see*, a characteristic of direct-manipulation software. For accurate positioning (for example, to align two words exactly), direct manipulation may not be the most practical approach.

Agreed, but...

Answers from other TEX users expressed some of my intuitions in words, clarified some of my own mental shortcuts, and triggered new thoughts. I agreed on all advantages presented, but felt something was still missing. Most users enthusiastically put forward that TEX is unequaled for *some* documents; very few suggested they were, like me, using TEX for *all* documents they produce.

Users see TEX as particularly suited to the production of large, structured documents. Its batch process, ability to input files in other files, and virtually unlimited capabilities (such as number of paragraph styles) make large projects easier to handle. The separation of contents and visual appearance that it encourages (via macro programming or use) allows one to focus more easily on content, structure, and style.

To my surprise, many a TEX user I talked to admitted to resorting to direct-manipulation applications for short, less-structured documents (letters being the typical example) because "It's so much easier." I wonder what, specifically, they find easier. Letters – business letters, that is – may be seen, if not as one large, structured document, at least as a uniform collection of documents. Consistency, therefore, is as important to the 250 business letters I write every year as a one-off long report. Such a level of consistency, it seems to me, is more easily and more reliably achieved with TEX than with direct-manipulation software. As an example, the instruction `\JL99001 TME/MDe` typesets a letter header with my company's logo, my name, the date (in the proper language), the letter's reference number (here, `99001`), and the complete address block of my addressee (here, person `MDe` from company `TME`, as specified in a data file).

The major perceived obstacle to using TEX, to which some `comp.text.tex` members rapidly pointed, is its steep learning curve. TEX, or rather TEX packages, are often quite immediate to *use*. Despite a marked aversion for computer programming, my business partner used my TEX macros very readily, even without the user manual I never

The two pages on our using TEX and PostScript, out of a short document explaining our approach to our clients.

BECAUSE WE STRIVE FOR THE HIGHEST QUALITY, we invested in the best typesetting tools. All the paper documents we produce therefore rely on two highly acclaimed standard codes: TEX and PostScript. TEX is a typesetting system developed by Donald E. Knuth at Stanford University for "the creation of beautiful books—and especially for books that contain a lot of mathematics." PostScript is a standard page-description language developed by Adobe Systems Inc. "for imaging high-quality text and graphics."

Software tools

Taking advantage of their complementarity, we use TEX and PostScript in combination. We use TEX programming and encoding for all aspects of typesetting (arranging type on the page). We use PostScript programming and encoding for elements other than type, such as drawings and graphs. For simple illustrations, we write PostScript code directly. For complex ones, we use PostScript-oriented Adobe products: Illustrator™ for vectorial descriptions such as drawings, Photoshop™ for pixel descriptions such as sampled images.

Software compatibility

Since we go from scratch to final pages, compatibility issues are few, if any. As input from clients, we favor the simplest form of all: unformatted ASCII-encoded text or data (often referred to as "plain text"), readily obtainable with most software today. And as output, we convert the whole document to PostScript code, for printing or imaging on a PostScript device. Our clients, in other words, do not have to know anything about TEX to work with us.

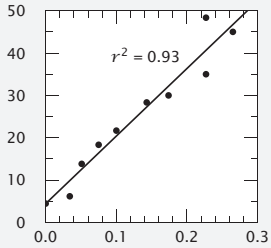
Advantages of own tools

Programming TEX and PostScript—a route on which few professionals venture—gives us three unequaled advantages over using off-the-shelf software: our work is *fast*, *flexible*, and *reliable*. First, we know our tools in and out, and are not slowed down by countless unnecessary options: we travel light. Second, whenever a feature seems to be missing, we program it: we can go anywhere. Third, whenever a feature does not work as we intended, we can and do fix it: we'll get there. We can therefore guarantee a well-done job by a certain date, without fearing that the software irremediably let us down at the last minute.

Freedom has its price: programming TEX and PostScript requires skill and dedication. Skill we have acquired through a high-level technical education, then refined through ten years of practice. Dedication is fueled by our (and our clients') satisfaction with the results. Though we seldom recommend this approach to others, we will continue to develop our own tools, while guaranteeing full compatibility of the output pages with printing devices.

The beauty of TEX

TEX (pronounced “tech” or “teck”) is not so much a word processor as a programming language, complete with typed variables, block structure, executable statements, and a powerful macro facility. The TEX compiler turns a one-dimensional source program into a two-dimensional typeset page. Below is an example of source code (right) and resulting output (left).




```
\draw[\white]{8\pc}{8\pc}
\xrange03{10} \xaxis01{10}34
\yrange0{50}1 \yaxis0{10}151
\markerdata
0.034 6.1667  0.051 13.833  ...  \enddata
\linedata
0.0 4.45  0.3 52.28 \enddata
\an{0.2}{35}{$r^2=0.93$}rb
\enddraw
```

TEX’s advantages are numerous. Dedicated to high-quality typesetting, it produces the best output available today, especially for such tasks as kerning lines, hyphenating paragraphs, and displaying mathematics. It allows accurate positioning (better than 1 µm) and can tackle virtually any language, in any alphabet. As a clearly defined language, it is platform-independent and stable over time, allows a high level of automation and extension, nicely separates contents from format (thus encouraging logical design, rather than visual one), and works with plain-text source files (smaller, easier to manipulate, edit, transfer, and compress, and much harder to mangle by accident than word-processor files).

The example graph above exemplifies some of TEX’s advantages. The plain-text code is compact and compiles fast to produce a consistent graph. Typeface, tick lengths, and line widths are defined at the top of the document, and apply by default to all graphs. The graph is exactly eight interlines high, and is shifted down by $^1/_4$ of an interline; bottom labels are aligned to a line of text, contributing to the overall harmony of the page.

Because typesetting is a complex process, so, unavoidably, is TEX. While merely using existing macros is simple—much simpler, in fact, than using a word processor—writing one’s own set of macros is not. Though we would personally settle for nothing less, we consider TEX a tool for professionals and a priori do not recommend its widespread business use around the office.

got around to writing. By contrast, TEX, or rather the fine programming of TEX, is not that immediate to *learn*, as confirmed by Knuth's "dangerous bend" signs in *The TEXbook* (1984). The same business partner was often at a loss when something unexpected occurred: the mental paradigm she had built through practice was insufficient to understand those cases.

Conclusion

Faced with people who equate "most popular" with "best", I stopped saying I work with TEX in casual conversation. Today, if people ask, I usually say I use "my own system" and refrain from explaining further. If people insist, I give them a short document that explains my company's approach, including its choice of software tools (Figure 1). Because they may care little about output quality and because our approach bypasses portability issues, the two pages on TEX and PostScript emphasize the three other categories mentioned above: we say it allows our work to be *fast*, *flexible*, and *reliable*.

While I have been a TEX enthusiast for over ten years, I have never been a TEX evangelist. More importantly, while I am convinced that using TEX my own way is one of my best professional moves, I have never advised anyone to develop his or her own package from scratch, let alone use mine. Some clients, who do notice the superior output TEX allows me to produce, have asked what software tool I was using in order "to buy it for themselves". I have to disappoint them, telling them my "tool" could not really be bought. Other clients, who had heard about TEX, asked me for the documents' source files, so they could convert them automatically to HTML. How could their HTML converter know, for example, that my TEX command `\Cs[137]` produces ^{137}Cs?

Using TEX in the real world, where time and money matter much, may require a dedicated TEX wizard. A well-oiled macro package may save considerable time and money by yielding consistently beautiful documents fast. It may, however, not account for the admittedly limited but nonetheless important special cases. In those cases, real-world users may want to call upon a TEX wizard, sometimes on short notice. How severe a limitation this is depends on management strategy. Learning TEX, in my case, certainly paid off, but it was quite an investment, one that was eased by personal motivation but that may not make sense from a pure business point of view. Still, it has given me an edge in many demanding professional cases, in which I can – actually – do what others can not.

I may remain a lone traveler, but my mind is made up: I will go on traveling light, going anywhere I please, and resting assured I'll get there.

Reference

Knuth, Donald E.. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1984.

graphics

Drawing effective (and beautiful) graphs with TEX

Jean-luc Doumont
JL Consulting
Achterenberg 2/10
B–3070 Kortenberg, Belgium
email: JL@JLConsulting.be

abstract

A standard approach to producing documents that include illustrations consists in typesetting text with specialized typesetting software (such as TEX) and inserting illustrations created with different, equally specialized software. To better integrate the illustrations into the typeset page, it would be nice to be able to produce or modify them directly with the typesetting software. Drawing graphs with TEX, for example, would allow one to set them `\hsize` wide and `0.75\hsize` high, position labels exactly `\baselineskip` below the horizontal axis, and, especially, typeset all annotations with the same fonts, sizes, and mathematical beauty as the rest of the document.

The hybrid TEX and PostScript macros presented in this paper take advantage of TEX's power to graph and annotate data sets in a variety of ways in order to produce effective, beautiful, well-integrated graphs. They use TEX to draw all horizontal and vertical lines (axes, tick marks, grid lines) and set all annotations, and PostScript to draw the data, as markers, lines, and areas. While fairly simple, they have been successfully harnessed to appear in a wide range of real-life applications, up to logarithmic graphs and (with some patience) complex multipanel displays. Of course, the macros are a tool for drawing final graphs rather than exploring or transforming data sets.

Most designers produce documents by assembling components produced with different tools: typically, they typeset text with a text-oriented application and create illustrations with one or several drawing or graphing applications. This approach, it would seem, offers the best of both worlds, by using the best-suited tool for each part of the job. Unfortunately, it often suffers one – in some cases, major – drawback: the poor integration of the illustrations in the typeset page.

Though increasingly sophisticated, graphing applications still offer but limited control over some details of the display, all the more so if their focus is data manipulation rather than data visualization. Among the less accessible parameters are:

- the size of the display and the size and position of the graphing area within the display, etc.;
- the thickness of the axes, the length of the major and minor tick marks, the length and spacing of line segments in dashed lines, etc.;
- the typeface used for text elements and their relative or absolute position (including sub- and superscripts).

As a consultant on technical communication, including graphing, I am disappointed with the output of many graphing applications (or at the considerable efforts needed to produce acceptable output). In line with the recommendations of such authors as Jacques Bertin (1973), William Cleveland (1985), and Edward Tufte (1983), I encourage the participants of my training programs to produce simple, intuitive, visually correct representations that favor data over decoration. Yet I find the default output of many applications to be overdecorated or hard to decipher. I was in search of a system that would help one focus on data, not decoration, much in the way TEX can help one focus on logical structuring, not visual rendering.

The idea of harnessing TEX to draw figures is nothing new, going back at least to Leslie Lamport's original LATEX package. The basic principle is always the same: define a coordinate system and position objects with respect to it by utilizing all of TEX's accuracy. The macros presented in this paper follow this principle but introduce some new ideas, such as using stretchable glue (rather than coordinate calculations) to position tick marks.

My graphing macros were developed in several steps. Originally, I was using dedicated graphing software but became frustrated with the poorly set text elements it offered, especially as soon as these involved any level of mathematics (symbols, sub- or superscripts, etc.). I then began to produce graphs without any annotation, insert them into my TEX source files, and overprint all text elements, including numerical labels along the axes. I soon realized that if I was able to place numbers along axes properly, I could as easily add a tick mark next to each, so I decided to display the data themselves with graphing software and draw axes and grids with TEX. Displaying the data with PostScript code inserted into TEX was the logical next step.

This paper first explains the principles underlying the macros for drawing axes, data, and annotations, then dis-

cusses extensions, limitations, and advantages of the approach.

Coordinates and axes

As you might expect, the macros eventually produce the graph as a TEX box of given width and height, which one specifies with the delimiting commands `\draw{`⟨*width*⟩`}{`⟨*height*⟩`}` . . . `\enddraw`. The `\draw` command accepts as optional argument (in square brackets à la LATEX) a specification of the graph's background color, the default being transparent (*not* white).

```
\draw{8pc}{8pc}
\xrange{0}{3}{10}
\yrange{0}{50}{1}
\xaxis{0}{1}{10}{3}{4}
\yaxis{0}{10}{1}{5}{1}
\markerdata ... \enddata
\linedata ... \enddata
\an{.2}{36}{$r^2=0.93$}rb
\enddraw
```

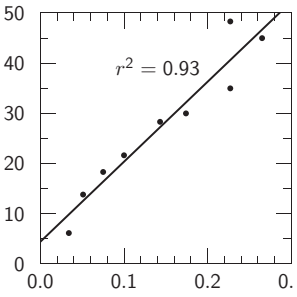


Figure 1 A simple graph and (part of) the code that generated it. Fractionary numbers are specified as fractions; for example, `\xrange{0}{3}{10}` specifies the width of the graph to correspond to the range [$^0/_{10}$, $^3/_{10}$].

The macros then position drawing elements with respect to, but not necessarily within, the reference box by using a coordinate system defined as a *scale* and an *offset*. In the horizontal direction, for example, the position `\x` (a dimension) is calculated from the coordinate ⟨*x*⟩ (a number) by the linear transformation `\x=`⟨*x*⟩`\xsc \advance\x+\xoff`. The scale `\xsc` and offset `\xoff` – two dimensions – are not specified as such by the user but are calculated from the specified graph width (for example, `8pc`) and axis range (for example, from 10 to 50 on the y-axis). Unlike the calculation of the position `\x`, which involves *multiplying* a dimension by a (possibly fractionary) number, that of the scale `\xsc` from the graph width and axis range requires *dividing* a dimension – an operation TEX restricts to integers. To allow fractionary ranges nonetheless (say, from 0.0 to 0.3, as in Figure 1), the range specifications foresee a denominator: `\xrange{0}{3}{10}`, for example, specifies the horizontal axis to range from "zero divided by ten" to "three divided by ten" or [0.0, 0.3].

While the `\xrange` and `\yrange` macros specify the range of the horizontal and vertical axes (corresponding to the graph width and height), they do not specify how axes should be drawn; the `\xaxis` and `\yaxis` macros do. These macros take as arguments the coordinate of the first major tick mark, the increment between major tick marks, a denominator (as for `\xrange` and `\yrange`), the number of major tick marks after the first one, and the number of minor tick marks between two major ones. The horizontal axis of Figure 1, for example, was drawn with the command `\xaxis{0}{1}{10}{3}{4}`. The denominator (10, in this example) indirectly specifies the number of decimal places for the numerical labels: the command `\xaxis{0}{10}{100}{3}{4}` would have labeled the axis with values "0.00" to "0.30."

Rather than positioning each tick mark and corresponding value with a scale and offset system, the macros place them in a box of the proper width or height and separate them with equal amounts of stretchable glue, so tick marks end up equidistant. By using different amounts of glue (specifically, $\log 2 - \log 1 =$ `0.3010fil`, $\log 3 - \log 2 =$ `0.1761fil`, and so on), they can even produce logarithmically spaced tick marks without having to calculate any logarithm at all (Figure 2).

The axes can be drawn in different combinations and fine-tuned in various ways. First, optional boolean arguments to the `\xaxis` and `\yaxis` macros enable or disable parts of the axis drawing, such as axis line, numerical labels, major or minor grid lines, and mirror axis at the right or top. Second, many parameters defined as dimensions can be freely specified: the line thickness, the length of both major and minor tick marks, the distance between axis and numerical labels, and the possible dash pattern used for the grid lines.

The macros actually offer a fairly general way of drawing straight lines. The fully TEX macros `\hl` and `\vl` draw horizontal and vertical lines, respectively, with length and starting points specified in drawing coordinates. They use the current value of line thickness and dash pattern (if any), and draw "projecting square caps" (matching a PostScript "linecap" value of 2), so perpendicular lines originating from the same point connect nicely. They are complemented by the hybrid TEX/PostScript macro `\rl`, drawing sloped lines in a similar way.

The macros for setting the range and those for drawing the axes are very flexible: they are largely independent, can be used several times in a given graph, and can appear before or after other drawing commands (although some commands obviously require the proper range to be set first). The range may thus be redefined to allow the correct display of data expressed in other units. Moreover, the axis need not extend to the full range. It can be drawn with several `\xaxis` or `\yaxis` commands having different arguments and boolean flags; in this way, it can show tick marks along its full length but numerical labels for only some of the tick marks (Figure 2). It can also be drawn before the data and be overprinted by them, or after the data and overprint them (Figure 3). Finally, it can be use-

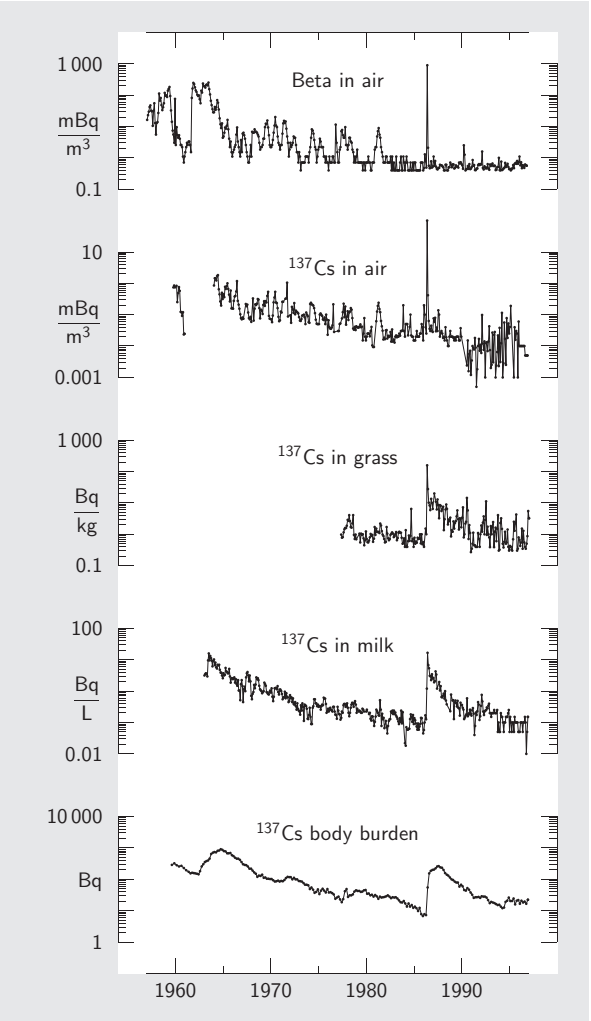


Figure 2 A more complex display, using several offset vertical scales (BNRC, 1998). With the data entered as logarithmic values, the apparently complex problem of logarithmic display boils down to setting tick marks right (with appropriate amounts of stretchable glue) and replacing the corresponding numerical labels x by 10^x (here, this default labeling was replaced by more readable annotations).

fully complemented by horizontal or vertical lines (Figures 4 and 5).

Data

Though they could have used TEX code for some, the macros consistently use PostScript code for all forms of data representation: markers, lines, boxes, error bars, areas, etc. In a rather straightforward way, constructs such as `\linedata⟨data⟩\enddata` push the data onto the PostScript stack, then invoke a PostScript routine to handle them. This routine accesses some TEX parameters, most importantly the vertical and horizontal scales and offsets.

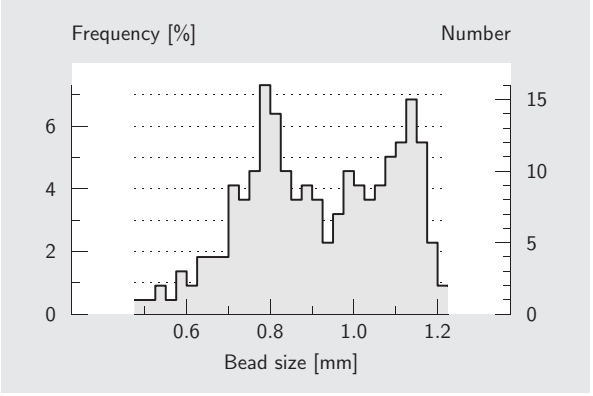


Figure 3 A histogram with different axes left and right (Vandenbroeck et al., 1996). The command sequence specifies the drawing sequence: first the y-axis with the corresponding grid lines (background), then the data, then the x-axis (foreground).

The data set is coded as a simple space-delimited list of space-separated pairs of values – or occasionally triplets, as when specifying error bars (Figure 4). Such an encoding can typically be obtained by a simple copy-and-paste operation from the tabular representation of a spreadsheet. In a prior step, the spreadsheet application may help put the data in the desired form; it may, for example, compute their logarithm or perhaps display them with a limited number of significant digits, to avoid making the subsequent TEX file unnecessarily large.

The data-graphing macros (`\linedata`, `\markerdata`, etc.) accept as optional argument a further specification of the way the data must be rendered. This argument, in the form of either direct PostScript code or TEX macros expanding to PostScript code, may thus specify the width and color of the line, and the size and shape of markers as well as the clipping area and any other PostScript parameters such as line caps, line joints, or dash patterns. Default parameters may be specified via the `\everywave` macro.

Annotations

The macros provide two forms of annotations. The numerical labels along the axes are positioned automatically by the `\xaxis` and `\yaxis` commands. All other annotations, including the labels of the axes, are positioned by the `\an` macro or variations of it.

In a way somewhat similar to LATEX's `\framebox` construct, the `\an` macro positions annotations by any corner of their enclosing box. For example:

```
\an{0.2}{35}{$r^2=0.93$}rb
```

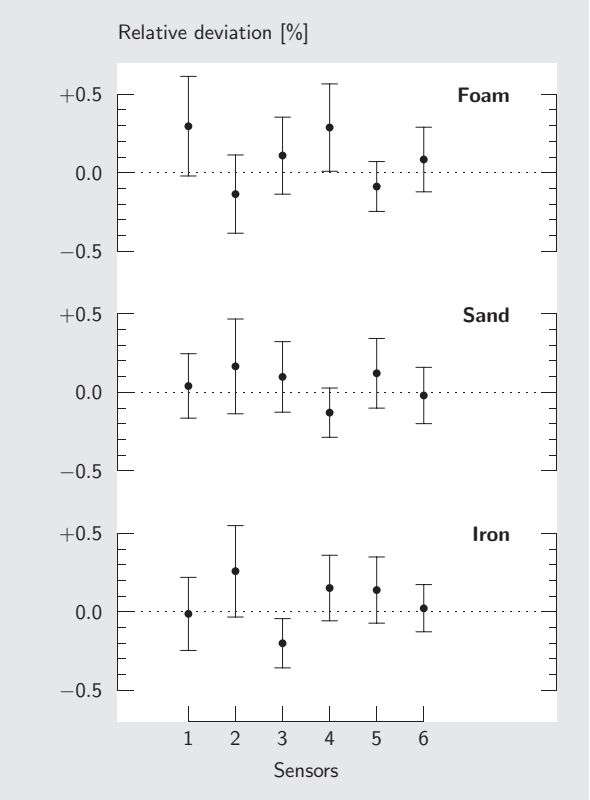


Figure 4 A graph with data and error bars (BNRC, 1998). The vertical axis is complemented by a zero line. The error bars are actually entered separately, as a list of triplets. Like other graphical elements, they may be drawn before or after the data points themselves – an issue when markers are of a different color (white-filled, for example).

places the annotation $r^2 = 0.93$ such that its right (`r`) bottom (`b`) corner is positioned at coordinates (0.2, 35). More exactly, it places the annotation so that its right bottom corner is positioned a distance `\dx` left and `\dy` up from the point of specified coordinates. The offset (`\dx`,`\dy`) allows the user to position annotations "a little away" from the object they annotate, such as a data point. Like the text itself, the specified offset does not scale with the dimensions of the graph; on the other hand, coordinates computed to be "a little away" do scale with the graph.

Although the numerical labels are usually set automatically by the `\xaxis` and `\yaxis` commands, they can also be set manually, for example, when they are non-equidistant or should otherwise be rendered differently (Figures 2 and 5). In this case, the offset (`\dx`,`\dy`) can be set equal to the offset specified for numerical labels, so `\an` sets the annotation exactly where `\xaxis` or `\yaxis` would.

As third and main argument, the `\an` macro accepts just

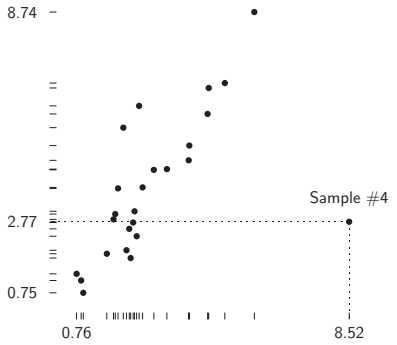


Figure 5 A dot-dash-plot à la Tufte (1983). Ranges are specified but not axes. Data are displayed as three `\markerdata...\enddata` series (dots, horizontal marks, vertical marks). One horizontal line, one vertical line, and six annotations complete this sober display.

about anything: text, of course, a framed box, a `\special` command, even another graph. Thus, it provides a very general way to position objects with respect to a coordinate system. Because it was meant for short text snippets, it considers the "bottom" of the box enclosing the annotation to be its baseline, not its real bottom (⟨*depth*⟩ below the baseline), so text with and without descenders would be positioned in the same way.

Because the coordinate system can easily be changed at any time (by calling the `\xrange` and `\yrange` macros or even by assigning new values directly to the scales and offsets), the annotations may be aligned to elements other than the graph data. For example, if the vertical scale is set so that one unit corresponds to one `\baselineskip`, annotations can easily be set to align to text elements outside the graph; such an alignment may be nice for integrating the graph in a multicolumn page designed on a strict underlying grid.

Drawings other than graphs

Although designed with graphs in mind, the macros can easily accommodate other types of illustrations based on the accurate positioning of graphical elements according to a coordinate system (Figure 6). With a macro or two for framing text, they make organizational charts possible – no PostScript required. With additional PostScript arrow heads and non-rectangular shapes (such as circles or diamonds), they can be used to create flow diagrams (Figure 7) or more complex drawings still.

Limitations

Like most macros, those presented in this paper are limited in scope: they are a TEX–PostScript hybrid, relying on

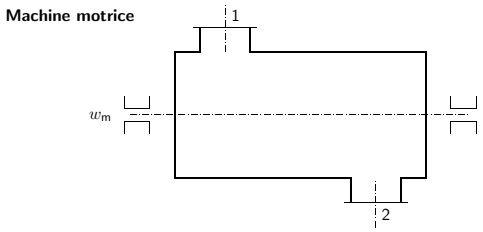


Figure 6 A simple drawing based on the graphing macros (Giot and Streydio, 1995). All elements are easily drawn with TEX commands (no PostScript).

TEX's limited arithmetic capabilities, and are intended for producing final graphs rather than playing around with the data.

First, the macros show the limitations of all TEX macros relying on PostScript: their use of `\specials` makes them implementation-dependent and their output is not readily visible in a DVI viewer. To see the graphs, one must typically convert the `.dvi` file into PostScript, then print this file or view it on-screen. This possibly slow process may render the visual optimization of a graph somewhat laborious (the legible positioning of annotations in complex graphs comes to mind). Still, the PostScript code manipulates graphical elements only; it need not manipulate text.

Second, the macros rely on TEX's limited arithmetic capabilities for calculating the scales, so they suffer round-off errors when handling large numbers, e.g., when an axis is specified to range from 0 to 10,000,000. (Of course, such large numbers read poorly and are best replaced by smaller numbers in larger units.)

Third, the macros are designed to display data sets but clearly not to explore or transform them. In this respect, they do not replace dedicated graphing or data-processing applications, thereby allowing one to select which data to represent and in which graphical form to represent them. (Similarly, TEX is meant as a tool for typesetting text, not writing it, although many of its features certainly facilitate the writing process.)

Of course, the macros may been seen as having many more limitations, although these usually reflect deliberate design choices rather than constraints: they are essentially restricted to two-dimensional representations and certainly do not favor decorative depth effects on either axes or data, they assume direct labeling of the data and thus provide no immediate way of creating legends, they foresee no mechanism for drawing pie charts or for setting text other than horizontally, etc. (Additional effects can be added, of course, sometimes quite simply, by anyone familiar with TEX and possibly PostScript.)

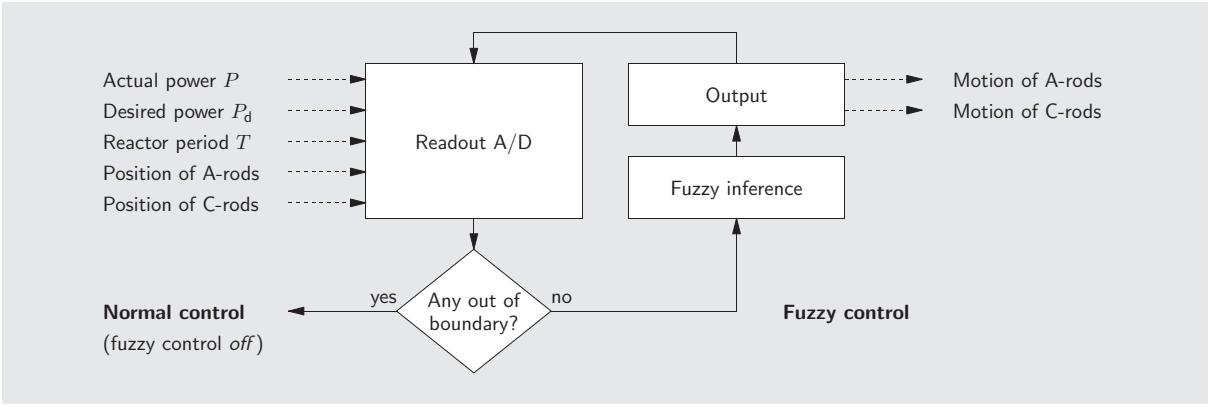
Advantages

Despite the above limitations, typical of many TEX macros, the macros presented in this paper offer indisputable advantages for integrating graphs with the rest of a document, for optimizing a collection of graphs, or for producing unusual displays.

By having access to TEX's parameters, the macros allow a harmonious integration of graphs into a document. For example, the graphing area can be specified to be exactly `\hsize` wide and `15\baselineskip` high so that it aligns well with other elements on the page and, of course, automatically rescales if the document is typeset in a different width or interline spacing. For documents based on a strict grid, elements of the graph can easily be specified to align with it: labels of the horizontal axis can line up on an adjacent baseline, text annotations can be left-aligned to a grid position, and so on.

As an important part of a smooth graph integration, the macros can typeset all annotations with the same typefaces, sizes, and mathematical beauty as the rest of the document. For example, one can easily add a formula to the graph, set

Figure 7 A flow sheet based on the graphing macros (BNRC, 1998). The only PostScript elements are the arrow heads and diamond outline.



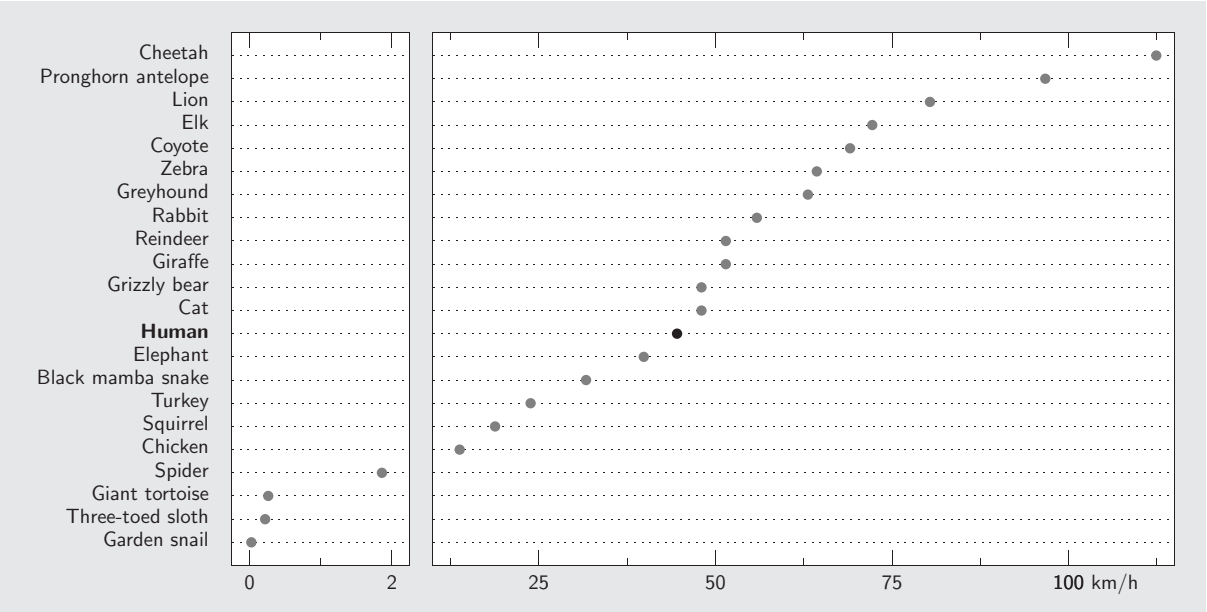


Figure 8 A dot chart à la Cleveland (1985). The left panel presents a view ten times larger than the right panel. The vertical axis is labeled with annotations, not numerical values. All elements other than the dots themselves are drawn with TEX commands.

an annotation on two lines exactly `\baselineskip` apart, or specify units in ISO notation, such as $J \cdot m^{-2} \cdot s^{-1}$ (with my macro `\[J.m-2.s-1]`; see Doumont, 1994). In contrast to imported illustrations (e.g., in encapsulated format), the graphs can scale the position and size of components independently by changing the corresponding TEX settings: an "enlarged" graph will thus not have thicker axes, bigger markers, or larger annotations.

Although TEX macros may not allow the visual optimization of a given graph as easily as a direct-manipulation application, they do allow the consistent optimization of a collection of graphs. In the declarative spirit of markup languages, the macros presented in this paper allow one to retypeset all graphs in a document at once by changing parameters (e.g., the axis thickness, tick length, or text size) at the top of the file. For this paper, for example, the length of tick marks and size of dot markers were adjusted globally after all graphs had been drawn. Direct-manipulation software, by contrast, would typically oblige the user to change these parameters for each graph separately – when they allow such parameters to be changed at all.

By giving access to all parameters in a TEX-like fashion, the macros also allow one to produce unusual graphs or combinations of graphs – sometimes admittedly at the cost of some work. Custom axes (Figures 5 and 8) are a typical example; small multiples (Figures 2 and 4) and scatterplot matrices (Figure 9) are others. With elementary PostScript programming, one can easily extend the range of available markers or visual representations in general, and even draw functions described by an analytical expression rather than a set of points.

Though it matters less and less with today's personal computers, the whole is also nicely compact. The macro package is below 32 KB and so is the file containing the nine displays of this paper. (As a comparison, the spreadsheet file containing the data for Figure 2 amounts to over 75 KB.)

Conclusion

The macros, refined progressively over the years, reached their final form about three years ago, when I co-authored a manual on communicating numerical data for Shell (Vandenbroeck et al., 1996). Since then, I have used them successfully to produce numerous real-life graphs and illustrations for my customers and for myself. Though designing an effective graph still requires careful thinking – something software will (probably) never take over – the macros help me focus on data rather than rendering details, by allowing me to specify many of these details separately and for all graphs at once. Most of all, by accessing the same dimensions and using the same positioning mechanisms as for the rest of the document, they allow me to integrate graphs and other drawings harmoniously, thus producing not merely beautiful graphs but also beautiful pages.

The macros have never been released before – so far,

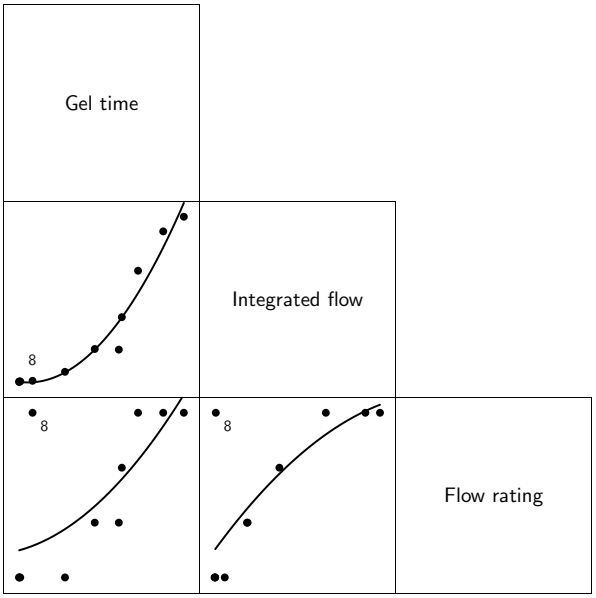


Figure 9 A scatterplot matrix, showing three views of the same data set, with an outlier (Vandenbroeck et a., 1996). The display is a combination of six graphs, three of which contain nothing but a central annotation. The three fit lines are not entered as a list of data pairs; instead, they are specified as an analytical expression, to be evaluated by the PostScript interpreter.

they have never been used by anyone but myself – but the positive feedback I have received on their output has encouraged me to do so: they will soon be available under the name `JLdraw`. I hope they can be as useful to others as they continue to be for me.

References

Bertin, Jacques. *Sémiologie graphique.* Flammarion, Paris, 1973.

BNRC. *Scientific Report 1997.* Belgian Nuclear Research Center, Mol, Belgium, 1998.

Cleveland, William S. *The Elements of Graphing Data.* Wadsworth & Brooks, Pacific Grove CA, 1985.

Doumont, Jean-luc. "Pascal pretty-printing: An example of preprocessing within TEX," TUB 15(3): 302–307 (1994).

Giot, Michel and Jean-Marie Streydio. *Chimie physique.* Université catholique de Louvain, Louvain-la-Neuve, Belgium, 1995.

Lamport, Leslie. *LATEX– A Document Preparation System.* Addison-Wesley, Reading MA, 1986.

Tufte, Edward R. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire CT, 1983.

Vandenbroeck, Philippe, Jean-luc Doumont, and Sophie Rubbers. *Communicating Numerical Data – Guidelines and Examples.* Shell Research (internal document), Louvain-la-Neuve, Belgium, 1996.

Hans Hagen & Ton Otten
(with thanks to
Karel H Wesseling
for the translation)

ConTEXt
Figures

abstract
Within the TEX community there is a widely used database for bibliographic references, BIBTEX, but not for figures. To manage figures ConTEXt now supports a figures database. The database is setup in XML and converted to an interactive PDF figure library featuring ordered displays and a search mechanism. From the library, figures can be included easily in ConTEXt documents as long as both the PDF and the XML files remain present.

Structure

Figures are described and maintained in an XML type file. Each figure is encapsulated in a `figure` element and the assembly of all figure elements is encapsulated in the main database element `figurelibrary`:

```
<figurelibrary language="en">
  ...
</figurelibrary>
```

The `figurelibrary` element has a language attribute to tell ConTEXt which language is used in the included texts. Only one language can be specified for each figure database. The default language is english and need not be specified.

The database has comment fields to describe shared characteristics of the figures. They are encapsulated in a `description` element. Several sub–elements or fields in the element are supported and each field can be omitted, left empty or filled with many lines of text. The three dots ... signify this possibility.

```
<description>
  <organization>...</organization>
  <project>...</project>
  <product>...</product>
  <comment>...</comment>
</description>
```

Each figure, as described in the figure element, has four fields of which two (`file` and `label`) are mandatory and must be completed, and two are optionally filled:

```
<figure>
  <file>filename.png</file>
  <label>symbolic name</label>
</figure>
```

If a figure is available in several formats, such as PNG and JPG, and you don't supply the file's picture format in the file extension, ConTEXt locates the most appropriate one automatically, given that the pdfTEX compiler can handle PDF, PNG and JPG formats and METAPOST. To keep the database general purpose we advise you, however, to supply the file extension together with the file name as shown.

If the figure is not yet available, a placeholder can be substituted for the future file name by replacing the `file` by a `dummy` sub–element:

```
<figure>
  <dummy width="4cm" height="3cm">to be made</dummy>
  <label>mandatory symbolic name</label>
</figure>
```

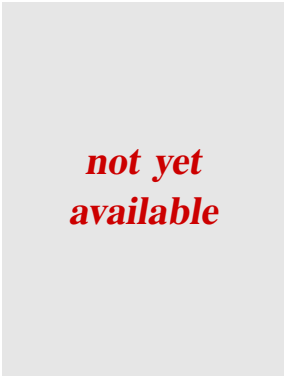
Note the positions of the <> brackets. The `label` element associates a symbolic name with the figure, to be used later to recall the figure from the database.

Usage

The figure library is a PDF file generated from the XML specification file, say `myfigs.xml`, by processing it by TEXexec (a PERL script in the ConTEXt distribution available from Pragma ADE):

```
texexec --pdf --use=fig-make myfigs.xml
```

The first option (`-pdf`) tells TEXexec to generate a PDF file. The second option (`-use`) invokes the style file `fig-make`[1] that comes with the ConTEXt distribution. TEXexec invokes ConTEXt which calls PDFTEX with this style file.
The resulting PDF file receives the name `myfigs.pdf`. It is an interactive figure library for a PDF viewer such as ACROBAT or GHOSTVIEW. Up front in the library file are the individual figures shown at their natural size. At the rear in the file is a summary with the descriptions, a table of contents and an index. Figure 1 below shows the figures of the example given later.



page 1 page 2 page 3

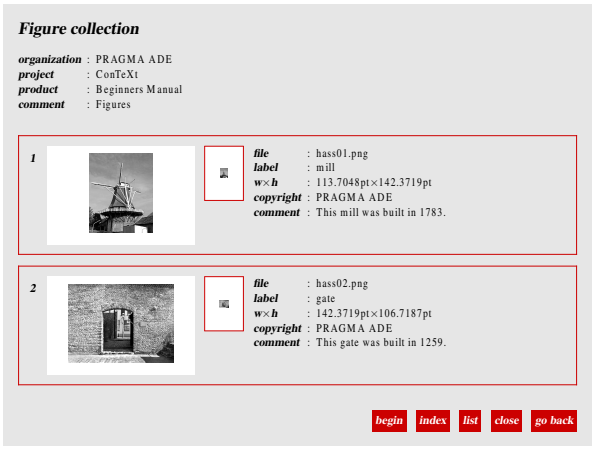
Figure 1 The first three pages of the figure library file.

Figure 2 shows the summary. In the summary the figures are shown first at a reduced height, second at their size with respect to A4 paper size, third with their descriptions.
At screen page bottom is a navigation menu to reach the table of contents or the alphabetically ordered index. Click on a figure to switch between full size and reduced size. Leave the viewer by clicking on `close`.

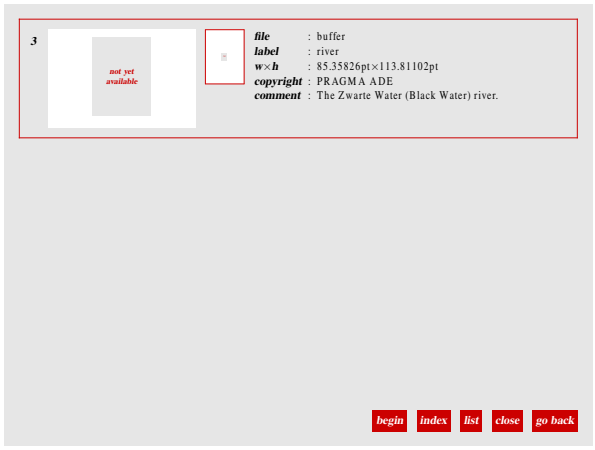
Instead of showing the figure relative to A4 paper size other sizes can be chosen with the TEXexec `-mode` command line option:

```
texexec --pdf --use=fig-make --mode=letter
```

1. `fig-make` is a synonym for `x-fig-01`

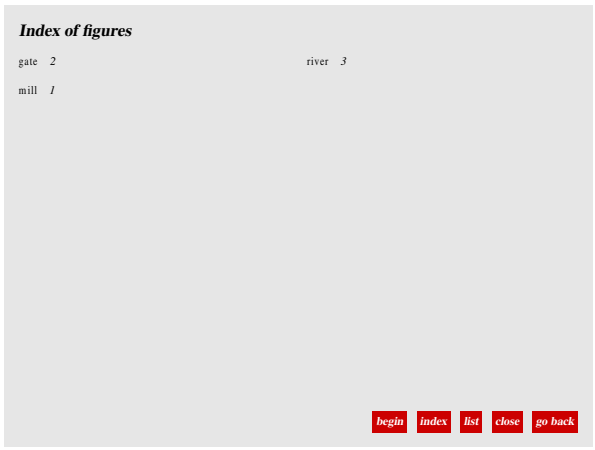


page 4



page 5

page 6



page 7

Figure 2 The overview, table of contents and index of the figures that are present in the file.

The choices for `-mode` are `A4`, `letter` and `compact`, of which `A4` is default.

`mode`	meaning
`letter`	map the preview on letter size
`compact`	use an alternative presentation

An XML example

The XML database specification file for the example shown above in figures 1 and 2 is:

```
<figurelibrary language="en">

  <description>
    <organization>PRAGMA ADE</organization>
```

```
    <project>ConTeXt</project>
    <product>Beginners Manual</product>
    <comment>Figures</comment>
  </description>

  <figure>
    <file>hass01.png</file>
    <label>mill</label>
    <copyright>PRAGMA ADE</copyright>
    <comment>This mill was built in 1783</comment>
  </figure>

  <figure>
    <file>hass02.png</file>
    <label>gate</label>
    <copyright>PRAGMA ADE</copyright>
    <comment>This gate was built in 1259</comment>
  </figure>

  <figure>
    <file>buffer</file>
    <label>river</label>
    <copyright>PRAGMA ADE</copyright>
    <comment>The Zwarte Water (Black Water) river</comment>
  </figure>

</figurelibrary>
```

A TEX example

To apply the database to a ConTEXt document the figure library processing module must be specified:

```
\usemodule[fig-base]
```

Next specify which database to use:

```
\usefigurebase[myfigs]
```

From this point, having both the XML and the PDF files available, you may place figures in your document file to be extracted from the (one or more) PDF libraries just specified:

```
\placefigure
  {A very old mill.}
  {\externalfigure[mill][width=4cm]}
```

Another example using more options of the `\placefigure` command is:

```
\placefigure
  [left] [fig:the gate] {This is an old city gate.}
  {\externalfigure[gate][height=4cm]}
```

Figure 3 This is an old city gate.

the result of which is shown in figure 3. A figure may also be included by its page number in the figure database:

```
\placefigure
  {This is an old city gate.}
  {\externalfigure[myfigs][page=2,
   width=.3\textwidth]}
```

This method is not recommended for a final document since updating the `myfigs` figure database may change the page number of a figure thus resulting in a document with errors.

Finally, it continues to be possible to load a figure directly from disk:

```
\externalfigure[myfigure][width=4cm]
```

or

```
\useexternalfigure[my not too large figure][myfigure][width=4cm]
\externalfigure[my not too large figure]
```

When the figure base module is loaded, ConTEXt will first try to locate `myfigs` in the XML database. This file, therefore, must be kept and not deleted after construction of the PDF file. When the figure cannot be found there, ConTEXt will search for a file with the name `myfigure` which may include a file name extension.

Apart from the already mentioned TEXexec `mode` option there are no other options that can be used.

Non–ConTEXt users can access the database by means of an additional file. For this, you need to run TEXexec once more:

```
texexec --pdf --use=fig-fake yourfile.xml
```

The resulting file, `yourfile.fig`, can be loaded in your document in the normal way. The figures can then be accessed with:

```
\getfigurefile{label}
\getfigurepage{label}
```

Given that you have also loaded `supp-pdf.tex`, you can now include the figure file with:

```
\includefigurefile width 10cm height 4cm {label}
```

Don't use this method in ConTEXt.

Documentation

ConTEXt has many more options to include graphics in your document, including scaling, clipping, positioning, subtitling, grouping and general manipulating. These techniques and more a found in the ConTEXt and MetaFun manuals available from our web site: `www.pragma-ade.com`.

DTP'en met LaTeX, gebruik en adviezen

Ernst van der Storm
evdstorm@mso.com

abstract
Verslag van het gebruik van LaTeX voor DTP-doeleinden, met enkele eenvoudige macro's, en het integreren van plaatjes: een beschrijving aan de hand van de praktijk.

keywords
LaTeX, 4allTeX, Postscript

Inleiding

Naar aanleiding van een email van Eric Frambach en Frans Goddijn volgt nu een reactie van een *niet*-TeX-wizard, iemand die nog steeds 4allTeX versie 4 (CD nummer 5) gebruikt, en die zelf iets in elkaar gezet heeft om een soort van *desktop publishing* te bedrijven.

Mijn gebruik van LaTeX stamt uit de tijd van de Atari-ST1040 (11 stuks 720kb-diskette's), maar inmiddels werk ik al jaren met 4allTeX. Ik heb me – voor zover ik me kan herinneren – nooit hoeven bezighouden met het maken van formats e.d. (ooit op die Atari misschien?), dus wie weet heb ik nog een keer iets aan een artikel waarin daarop ingegaan wordt.

Terzake, waarom LaTeX?

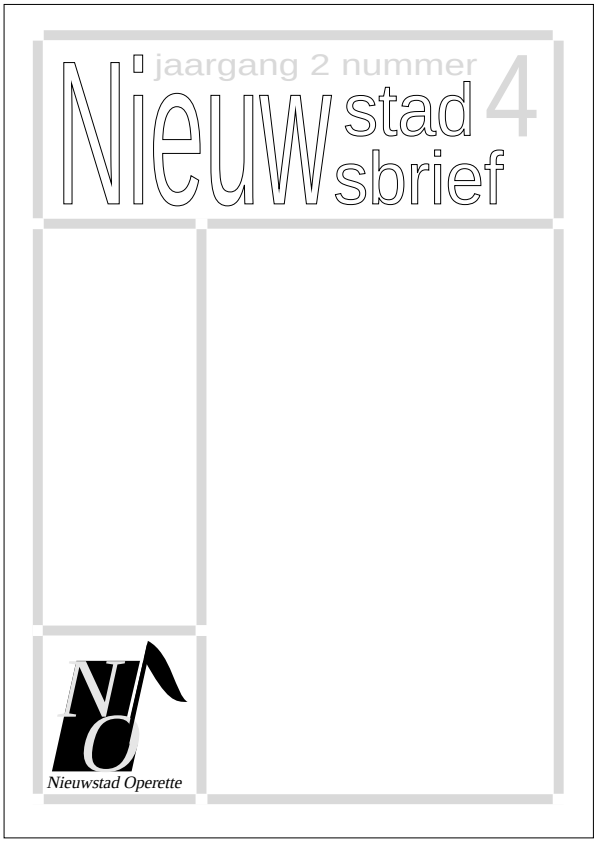
Voor de *Nieuwstad Operette* maak ik eens in het kwartaal een nieuwsbrief, en hoewel LaTeX daar niet bij uitstek voor geschikt is, ben ik vanaf nummer 0 in de weer geweest om daarmee die publicatie te realiseren.

Als je zoiets met Word aanpakt, dan heb meteen te maken met het feit dat als je teksten te groot zijn voor pagina 1, dat ze dan doorschuiven naar pagina 2.

Bij DTP wil je dat allemaal onder controle hebben. In MAPS nummer 23 staan twee artikelen die hierover gaan. Voor mijn blaadje maak ik van geen van beide oplossingen gebruik.

Vanwege (?) de hoge waarden voor `textwidth` en `textheight` heb ik doorgaans geen last van doorloop naar de volgende pagina: teveel output loopt in Ghostview 'naar beneden toe' van de pagina af. Dat is dan aanleiding voor een ingreep in de tekst.

Uitgangspunten voor de layout – hoofdzakelijk achteraf geformuleerd – zijn de volgende:



1. een artikel mag niet – tenzij het echt niet anders kan – doorlopen naar een volgende pagina; hiervoor is het nodig dat redactiewerk gedaan wordt, meestal tekst uit de aangeleverde kopij weghalen.

Een voordeel van LaTeX is in dit geval, dat je de oude tekst als commentaar tussen de definitieve kunt laten staan (met % aan het begin van de regel).

Uiteraard bieden plaatjes – die met `epsfig` eenvoudig tot een gewenst formaat zijn te vervormen – een goede mogelijkheid om pagina's geheel gevuld te krijgen.

2. de layout van de nieuwsbrief moet consequent zijn door het gehele document; bovendien moet de voorpagina een vaste, herkenbare indeling hebben; de globale layout be-

staat uit 3 kolommen, en *uitvullen* wordt door de redactie niet mooi gevonden.

Op de voorpagina is de linker kolom gereserveerd voor inhoudsopgave en colofon. Deze indeling wordt verder gedicteerd door een afbeelding in Postscript, die zoveel mogelijk de gehele voorpagina beslaat.

Hiervoor is wat kunst-en-vliegwerk nodig, met name moet hiervoor de tekstmarge aan de linkerzijde (odd- en evensided) op de juiste waarde gezet worden, en natuurlijk moeten `\texwidth` zowel als `\textheight` vergroot worden (naar resp. 210 en 297 mm) om geen *over-* of *underfull box*-meldingen te krijgen.

Veel verder naar buiten toe kan niet, want de gebruikte printer (HP Laserjet 2D) heeft aan alle kanten van een A4tje een onbedrukbare rand van zo'n 5 à 6 mm.

3. het gebruikte lettertype moet duidelijk zijn, omdat het leespubliek van het blad meestal 'een bril zal gebruiken' om te lezen.

Voor de hoofdtekst koos ik het lettertype *Palatino*, niet alleen omdat het er daarmee mooi uitziet, maar ook omdat Times New Roman wel erg gewoon is, en omdat Helvetica teveel lijkt op het in Windowsomgeving alom tegenwoordige en lelijke 'Arial'. Het enige wat je daarvoor hoeft te doen is `\usepackage{palatino}`.

4. plaatjes en foto's moeten *'alleen-maar-tekst'*-pagina's voorkómen.

5. veel meer dan 6 pagina's A4 (één A3 dubbelzijdig en een A4 idem) mag het drukwerkje niet beslaan, want het wordt met de hand in een oplage van zo'n 100 stuks gekopieerd.

6. eenvoud in het onderhoud, op z'n minst voor mezelf.

Waarom ik hiervoor LaTeX gebruik komt vooral voort uit een persoonlijke voorkeur – ben ik de enige? – en omdat het met LaTeX heel erg mooi kan worden.

Plaatjes zijn in dit soort uitgaven van groot belang om óf een speels effect te realiseren, óf om bijvoorbeeld foto's te reproduceren.

Postscript

Het uiterlijk van de voorpagina wordt bepaald door een afbeelding die ik m.b.v. Postscript zelf heb opgebouwd.

Hier moet je zin in hebben – zie eerdere artikelen o.a. van de hand van Kees van der Laan – maar dan héb je ook wat. Ter illustratie hieronder *een stukje* uit de Postscript-source (van `nieuwsb4.eps`):

```
% !PS-Adobe-3.0 EPFS-3.0
%%BoundingBox: 0 0 596 842
% beslaat dus in principe de hele pagina
% nieuwsb4.eps voorpagina nieuwsbrief nummer4
.
. code om de grijze strepen te tekenen
. en lettertype REencode
.
% invoegen logo:
gsave 0 setgray
% hiermee wordt hele logo verplaatst:
0 12 translate
%    logo komt nu links onder 0 15
0.275 0.28 scale    %    .25 .30
/_Times-Roman findfont
    305 scalefont setfont % 300
10 0 moveto

[1 0  0.225 1 0 0] concat     % schuin zetten
134 200 moveto 426 200 lineto
426 598 lineto 134 598 lineto
closepath fill
12 setlinewidth
1 setgray
371 385 moveto 374 655 lineto stroke
41 setlinewidth
1 setgray
413 300 moveto 413 655 lineto stroke
.90 setgray     % was .85
100 397 moveto (N) show
%   false charpath % 100 400
220 200 moveto (O) show
16 setlinewidth
1 setlinecap
0 setgray
387 300 moveto  % was 385
387 655 lineto  % was 685
stroke
1 setlinewidth
0 setgray
380 670 moveto  %was 655
477 625 525 475 575 450 curveto
530 450 500 450 380 605 curveto
closepath fill

/_Times-Roman findfont 62 scalefont setfont
128 140 moveto
(Nieuwstad Operette) show
grestore
% enz. enz.
```

Om de complete afbeelding goed op z'n plaats te krijgen is het volgende nodig:

```
% alleen voor het voorpaginaplaatje:
\vs{-17mm}\hspace*{-17mm}
 % naar linksboven op de pagina
\epsfig{figure=nieuwsb4.eps,%
    width=210mm,height=297mm}
\vs{-217mm}
% terug naar de verticale positie waar
% de eerste tekst begint
% (de verdere tekst moet op dezelfde pagina
% binnen het grijze kader komen)
```

Macro's

Dit levert het (maar dan voor dit artikel op 33% verkleinde) plaatje op de eerste bladzij op.

In de verwachting dat ooit iemand anders deze nieuwsbrief zou moeten kunnen produceren heb ik gezocht naar aanpassingen die het gebruik makkelijker maken, en het resultaat hiervan zijn enkele macro's: twee voor het beginnen en eindigen van de kolomindeling `\bmcolkop{3}{Deze tekst als kop boven 3 kolommen}` en `\emcol`, en de derde is een verkorte versie van `\vspace*{aantal mm}`, wat nu bijvoorbeeld gedaan wordt met `\vs{4mm}`.

Laatstgenoemde opdracht wordt doorgaans gebruikt om de balans in de kolommen enigszins te beïnvloeden, doorgaans in de rechter kolom, waar de naam van de auteur vermeld wordt.

Meestal is het op iedere pagina nodig, met bijv. `\vs{4mm}` extra witruimte in te voegen, om aan de onderzijde een uniforme ondermarge te krijgen.

Ook in *dit* artikel blijkt het soms nodig om witruimte toe te voegen, bijvoorbeeld om te voorkomen dat de vetgedrukte koppen te dicht op de voorgaande tekst staan.

Hoe je een macro moet maken vindt men bijvoorbeeld in "A guide to LaTeX"door Kopka en Daly, waar ik ook meer dan de helft van alle knowhow uit vandaan haal.

Tekst opnemen

De tekst is meestal als Word-bestand op diskette of per email aangeleverd. In dat geval bewaar ik hem via Word als *MSdos-tekst met behoud van einde regels*. Hiermee blijven de accenten in mijn editor op de juiste manier weergegeven. Dan volgt het omzetten van álle geaccentueerde letters naar `\'alle` LaTeX equivalenten. Dit is in 4TeX5.0 ongetwijfeld gemakkelijker, want ik begrijp dat de accenten zoals ze met *alt130* in een tekstfile terechtkomen ook door LaTeX begrepen worden (of is daar een stylefile voor?).

Reeds afgedrukte tekst wordt bij een kwalitatief goede afdruk door mijn scanner met OCR-software goed omgezet naar platte tekst.

Afbeeldingen in de tekst opnemen

In de praktijk heb ik te maken met drie typen afbeeldingen om in tekst op te nemen:

- afbeeldingen zelf geprogrammeerd in Postscript, zoals op de voorpagina; deze zijn met een `epsfig`-commando eenvoudig op de gewenste plaats in de tekst te zetten;

- plaatjes geïmporteerd van het internet; dit zijn doorgaans .gif- of jpeg-bestanden, die ik m.b.v. *Paint* meestal van een rand ontdoe en wegschrijf als bitmap.bmp; de omvang is meestal klein;

- afbeeldingen (meestal foto's) die met een scanner zijn ingelezen. De werkwijze is dan in principe hetzelfde, maar de omvang van de bestanden meestal groter. Voor de nabewerking is het vaak nodig de resolutie te verkleinen, en dit is het meest bewerkelijke gedeelte.

De nabewerking bestaat uit het omzetten van bitmap naar Enscapsulated PostScript. Ik gebruikt voor dit doel Graphical WorkShop van de CD.

Meestal wordt hiermee ook nog het aantal kleurniveau's verminderd. Het resultaat is een file met aan het begin een beetje en aan het eind een flink stuk binaire informatie voor intern gebruik van G.W.S. die er voor ons doel beter niet in kan zitten. File in de editor laden en deze eruit verwijderen.

Hiermee ontstaat een `.eps`-bestand dat door LaTeX zonder meer verwerkt kan worden.

Gebruik van de 4allTeX-software

Tenslotte enkele opmerkingen over de werkomgeving 4allTeX Workbench NTG-CD no. 5.

Voor het produceren van dit soort niet al te omvangrijke teksten is deze omgeving zeer geschikt.

Uiteraard kost iedere omgeving steeds weer moeite om te leren kennen, en om allerlei instellingen te krijgen zoals je ze wil hebben.

Kortgeleden heb ik de nieuwste versie (4Tex5.0) voor de tweede keer uitgeprobeerd, met een gedeeltelijke – en door diskruimtegebrek helaas nogal beperkte – installatie. Dat gebrek aan diskruimte is ongetwijfeld de oorzaak van allerlei foutmeldingen over fonts die niet gevonden worden, en waar in Ghostview*afdruk* dan ook niets van wordt weergegeven, maar ook de weergave van dvi-Win laat nogal te wensen over, vergeleken met Ghostview uit de vorige versie.

Belangrijker is dat het positioneren van afbeeldingen op de pagina op de manier zoals in dit artikel beschreven niet hetzelfde resultaat blijkt op te leveren.

Dezelfde sourcefile geeft een opvallend andere output (afbeelding hoger, tekst lager) in de nieuwe omgeving.

Aangezien mijn benadering (m.i. onvermijdelijk) veel gebruikt maakt van vaste verplaatsingen in millimeters zal dat in 4Tex5.0 allemaal opnieuw puzzelen zijn om een goed resultaat te krijgen.

Overigens is er ook een verschil in paginapositie zichtbaar tussen de de gewone vga-DVIviewer uit de oude versie en Ghostview: alle pagina's staan te ver naar links in het groene kader van de VGAviewer.

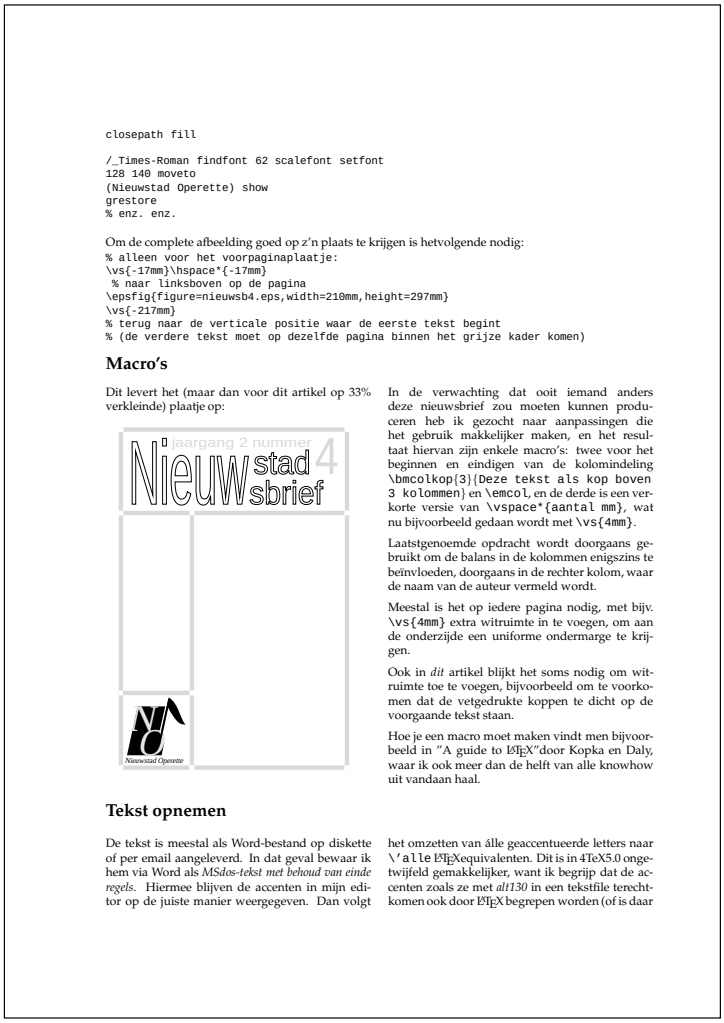
Er zijn blijkbaar toch een aantal parameters die per installatie – en per viewer – verschillen, en waarvan ik niet weet hoe of waar die vastliggen.

Wat ik in de '*praktijk van alledag*' ook waardeer aan de oude workbench is de mogelijkheid van het gebruik van mijn favoriete editor. Deze **STedi** stamt oorspronkelijk uit de Atari-tijd, en de *MSdos*-versie beschikt nog steeds over zeer handige *blockmove*- en *copy*-commando's, waarmee bijvoorbeeld voor een groep regels in één keer %-en gezet kunnen worden.

Met deze editor van de hand van *Jos Vermaseren* is ook het probleem van Edwin Drost uit de Maps van najaar 2000 op te lossen. Hij is zeer snel en krachtig – stream scripts! – maar is alleen helaas niet algemeen beschikbaar.

Het gebruik van include-file werkt ook zeer handig en dat is zo te zien in 4TeX5.0 vervallen.

Wel winst is de syntaxchecker die 'meekomt' met de MED-editor en die daarbij default beschikbaar is.



```
closepath fill

/_Times-Roman findfont 62 scalefont setfont
128 140 moveto
(Nieuwstad Operette) show
grestore
% enz. enz.
```

Om de complete afbeelding goed op z'n plaats te krijgen is hetvolgende nodig:

```
% alleen voor het voorpaginaplaatje:
\vs{-17mm}\hspace*{-17mm}
 % naar linksboven op de pagina
\epsfig{figure=nieuwsb4.eps,width=210mm,height=297mm}
\vs{-217mm}
% terug naar de verticale positie waar de eerste tekst begint
% (de verdere tekst moet op dezelfde pagina binnen het grijze kader komen)
```

Macro's

Dit levert het (maar dan voor dit artikel op 33% verkleinde) plaatje op:

jaargang 2 nummer 4
Nieuwstad sbrief
Nieuwstad Operette

In de verwachting dat ooit iemand anders deze nieuwsbrief zou moeten kunnen produceren heb ik gezocht naar aanpassingen die het gebruik makkelijker maken, en het resultaat hiervan zijn enkele macro's: twee voor het beginnen en eindigen van de kolomindeling `\bmcolkop{3}{Deze tekst als kop boven 3 kolommen}` en `\emcol`, en de derde is een verkorte versie van `\vspace*{aantal mm}`, wat nu bijvoorbeeld gedaan wordt met `\vs{4mm}`.

Laatstgenoemde opdracht wordt doorgaans gebruikt om de balans in de kolommen enigszins te beïnvloeden, doorgaans in de rechter kolom, waar de naam van de auteur vermeld wordt.

Meestal is het op iedere pagina nodig, met bijv. `\vs{4mm}` extra witruimte in te voegen, om aan de onderzijde een uniforme ondermarge te krijgen.

Ook in *dit* artikel blijkt het soms nodig om witruimte toe te voegen, bijvoorbeeld om te voorkomen dat de vetgedrukte koppen te dicht op de voorgaande tekst staan.

Hoe je een macro moet maken vindt men bijvoorbeeld in "A guide to LaTeX"door Kopka en Daly, waar ik ook meer dan de helft van alle knowhow uit vandaan haal.

Tekst opnemen

De tekst is meestal als Word-bestand op diskette of per email aangeleverd. In dat geval bewaar ik hem via Word als *MSdos-tekst met behoud van einde regels*. Hiermee blijven de accenten in mijn editor op de juiste manier weergegeven. Dan volgt het omzetten van álle geaccentueerde letters naar `\'alle` LaTeXequivalenten. Dit is in 4TeX5.0 ongetwijfeld gemakkelijker, want ik begrijp dat de accenten zoals ze met *alt130* in een tekstfile terechtkomen ook door LaTeX begrepen worden (of is daar

Hier een pagina van dit artikel in oorspronkelijke opmaak. Let op de afwisseling tussen één- en twee-koloms elementen, wat niet mogelijk is met de MAPS LaTeX stijl [Redaktie].

ConTEXt

From PC–Write to ConTEXt easy speedy beauty

Karel H Wesseling,
Gertrude L van der Sar,
Jos J Settels.
TNO TPD Biomedical Instrumentation Academic Medical Centre, Meibergdreef 9, 1105 AZ Amsterdam
email: j.j.settels@amc.uva.nl

abstract
A tale of more than 10 years of joy and struggle with TEX followed by a period of bliss, of easy to use tools, quickly obtained results, and incredible possibilities from the coming of 4TEX and ConTEXt, narrated by nongurus.

keywords
Nonguru, 4TEX, ConTEXt.

Introduction

This is about typesetting by nonexpert users. TEX sets type. TEX does the work that professional typographers used to do, still do: produce printed books from type written copy delivered to them. But TEX is not a person, TEX is software. To ease the task for TEX the copy writer inserts simple instructions with the copy text such as “switch to a larger font”, “this paragraph ends here”, “this line should go into a footnote”, such that TEX does not have to fathom from the copy text that these situations occur right there in the text. Since almost every TEX instruction is preceded by a backslash you might type instructions such as `\footnote{See bottom.}'` right in the middle of your text.
Typographers (judging from the few we met) are an independent bunch of professionals and they have individual preferences. So do publishing houses and writers. TEX has been designed to be so flexible that it can mimic an individual typographer’s preferences. This is done by making a new TEX from an as yet unbiased, pristine TEX by asking it to imbibe and digest a pack of rules of conduct (aka macros), representing the ideas of an individual typographer about proper behavior. This feature makes TEX incredibly versatile and powerful since it can, like an actor, change personality.

Whereas TEX was designed primarily—as said—to typeset beautiful books many scientists rarely write books. Their principal output is manuscripts for papers in a journal. A format that facilitates paper publishing is LATEX. It was especially designed (or so it seems to us) to ease the setting of tables, to include figures, to produce a list of references to other papers, in short, to facilitate everything needed to write papers.

Our early years

Years ago we were charmed into using TEX because we found the thought irresistable that for ordinary scientists it was possible to produce papers almost in the form of the final product: professionally typeset even on needle printers, and very good looking on a LaserJet. We could continue to use our favorite shareware editor (PC–Write). We bought PC-Tex and received a set of floppies, a printed manual, and a good luck blessing. We had no knowledge, no experience, no guru in the neighborhood, no time, and almost no

1. What we just said is possibly not entirely correct.

patience. Yet within 6 month we had a setup that did what was desired, was promised. We edited autoexec.bat and config.sys and created batch files and even got the PC-Tex menu going with yet further configuration files. Although no longer needed we still keep one PC in the lab with this system since one never knows how the need for it arises when an old text suddenly becomes relevant again. We used LaTeX exclusively and bought several copies of Lamport's book. All this for a system that we didn't really need because journals at the time (though reluctantly) even accepted handwritten, thus true, manuscripts. Plain ASCII files were entirely acceptable. When we obtained a copy of PiCTeX together with a 386 update of PC-Tex we finally began to need the TeX system since with PiCTeX it was possible to produce programmable, documented, updateable diagrams and figures of an almost bookprinter quality. Nothing convinces more than a beautiful graph.

Years later again we heard of the Dutch TeX users group, NTG, and became members to hear gurus speak of TeX, hoping for tips. Of the papers read enthousiastically at the meetings we understood little but this improved quickly, of the papers in the MAPS we understood some more, by rereading. Still, the "steep learning curve" that is often mentioned in relation to TeX continued unabated. And then, out of the thin blue air we received the 4TeX CDROM's and a small booklet explaining its installation and some of its features. We had no idea in a TeX environment about the true meaning of the word "workbench", except in carpenter's terms, but then we found out. Installation from the CD was a breeze, almost no disk space was used, and suddenly we had everything TeX available almost without knowing, thinking or reading. The learning curve was over, a thing of the past, the flatlands were reached, the possibilities expanded, and all of our favorites still available. Printing to any printer that was brought into the lab worked. Postscript output was no longer menacing. Yes, reader, you are so right, we were and continue to be TeX-dummies[2] with only one desire: to obtain beautiful output. How it is done we did't really even want to understand. We are, however, infinitely grateful to the good persons that made this possible to us.

More recently, publishing on the world wide web became popular and browsing and indexed searches of a document. Furthermore higher and higher demands were placed on the quality of (electronic) instruction manuals with the desire for animated pictures, the inclusion of voice, of abundant color. Did we have to return to medieval software? No. Just in time we learned of a new TeX system that facilitates all this and all of the above and more: ConTeXt. We learned of ConTeXt at an NTG meeting and liked its potential immediately, even though to us it sounded almost impossible to have a printable TeX document turned into browseable, clickable pages on screen with no greater effort than adding a few instructions at the start of the document. Yet it was demonstrated.

Making a User's Guide

Our lab developed a portable 24–hour continuous blood pressure recorder called Portapres (Fig. 1). An increasing number of colleagues wanted to obtain copies of that device. Soon there were so many copies made that personalized user instructions became impossible. In response, a User's Guide was created with lots of 'how–to–do–it' pictures, specification tables, lists of warnings and error messages, lists of published references, table of contents, table of key words, tutorial chapters, etc. It was created in Word. Yes. Simply, because it was there, and we wanted to build experience with WYSIWYG. The Guide took a year to produce as a part–time effort including the design of many hand crafted pictures. It was a huge file on disk. In printed form it was wellcomed

2. Dummy's Ok these days, even a marketing jewel.

and certainly well accepted by our users but a problematic aspect was that the process to create it was undocumented. The Guide itself was the (miserly) documentation of the process. Recently, in a MAPS article by Taco Hoekwater, he said that these systems are called “output only”. Such productions could well be submitted to the Journal of Irreproducible Results. And documentation demands were increasing in view of obtaining CE marking, needed later for being able to produce devices for others. What to do?

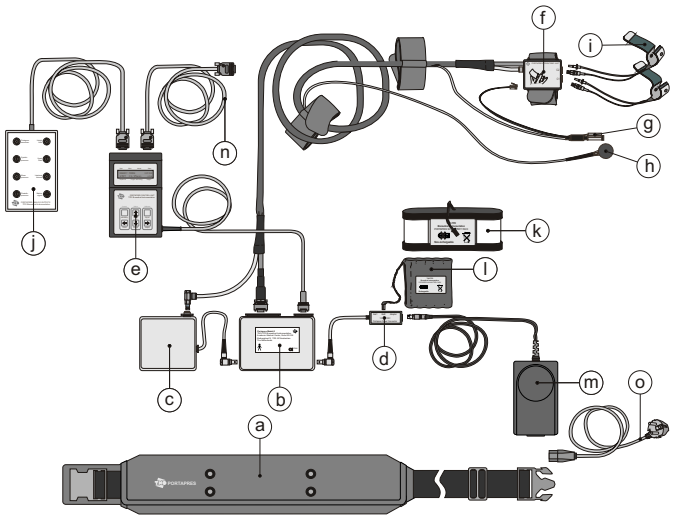


Figure 1 The component parts of a Portapres battery fed, 24–hour continuous, finger blood pressure recorder.

Instead of starting a new learning curve with a new and to us unfamiliar system immediately, we decided to delay and to call the authors of ConTEXt at Pragma in Hasselt. They agreed to a meeting and we showed our Portapres manual and asked them if it could be converted to ConTEXt with a printed and a downloadable manual and with a screen version in the bargain. After a brief pause for inspection and a few probing questions the answer was that it could. Then we asked if they could do it for us and they would. Our Word document was mostly automatically translated into ConTEXt understandable ASCII, but inclusion of the many pictures was another matter. They were line drawings in CorelDRAW and had to be converted to encapsulated Postscript (eps) files which was not too difficult. But how could we have known about the “bounding box”, the distortion of color, the proper inclusion of fonts in pictures. Once this box was understood and all the pictures bounded and boxed and otherwise done with, we received more or less final versions of the three output forms requested in portable document format (pdf), plus all the sources.

Was this a wasted effort? Not at all. Although it took more (boxing) time than we originally expected we now own and have in use three versions of the User’s Guide, one in our own paper size for the printed, bound manual, one in A4 paper size for remote printing over the Web with little space wasted and, most important of all, a screen version (Fig. 2). That version not only looks good but includes all the controls for browsing and searching that one can desire, and allows enlargements of pictures to obtain a closer look at some of the graphic details not visible on paper. In addition, by looking at the sources we learned much about programming such documents in a structured way which seems a strong feature of ConTEXt. We learned much about ConTEXt and how it is used by experts. We learned about unexpected features of ConTEXt such as the inclusion of Quicktime movies and blastable sound tracks (with a single instruction). By applying

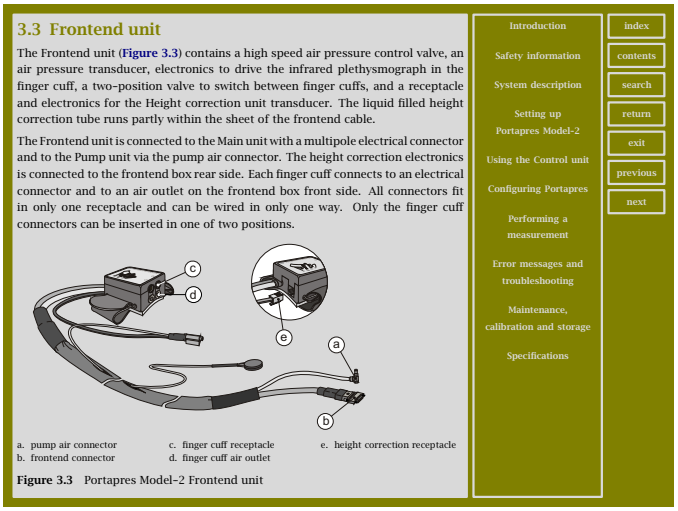


Figure 2 Page of the screen version of the Portapres User's Guide. To the right are two 'view control' columns. Blue colored words in the document are also clickable.

these principles and newly acquired techniques we learned to rely on ConTEXt more than we ever dared do on LaTeX. ConTEXt seems to have a logic built–in more similar to the way we think and everything falls in position. ConTEXt reassures us.

Not soon thereafter we had to produce yet another User's Guide for a new blood pressure recorder, in little time. So one of us (further called I), encouraged by the previous experience asked his son if he could do the ConTEXt work and the graphics while I was typing the texts. (Developing biomedical instrumentation with us often seems in a sweat shop hurry with entire families tied in.) This he refused outright. He had seen so much trouble with Windows that he objected even to 4TEX on his PC. But if he could do it in Word he was prepared and ready. When I said no he challenged me to produce the cover of the manual, he in Word, I in ConTEXt.

I accepted the challenge. Within 3 minutes he had a nice looking front page (Fig. 3) which included the already available picture as a bitmap. Within two days of trial and error and reading manuals and the magnificent 4TEX5 book and some emails to Pragma I had done the same in ConTEXt. Had my son won? I like to think not. When, days later, I asked him to do it again a slighty different page layout resulted. The Word–way was essentially undocumented. Never before did we realize the need for self–documenting methods more. What in science is the standard, on the current Microsoft PC seems the exception.

After transforming the pictures to portable network graphics (png) format for inclusion in ConTEXt documents (discovering that and details took most of the time) the remainder of the more than 100 page User's Guide took less than three weeks (19 days to be precise) full time. I did the texts and the ConTEXt instructions, my son did the "comics" (Fig. 4), the graphics and modifications. No minor task since almost half of the pages has a figure. It's not a production as advanced as Pragma had produced for us for the Portapres Guide but we met the deadline with time to spare. First user's reactions are positive and authors and readers both like the looks of the document.

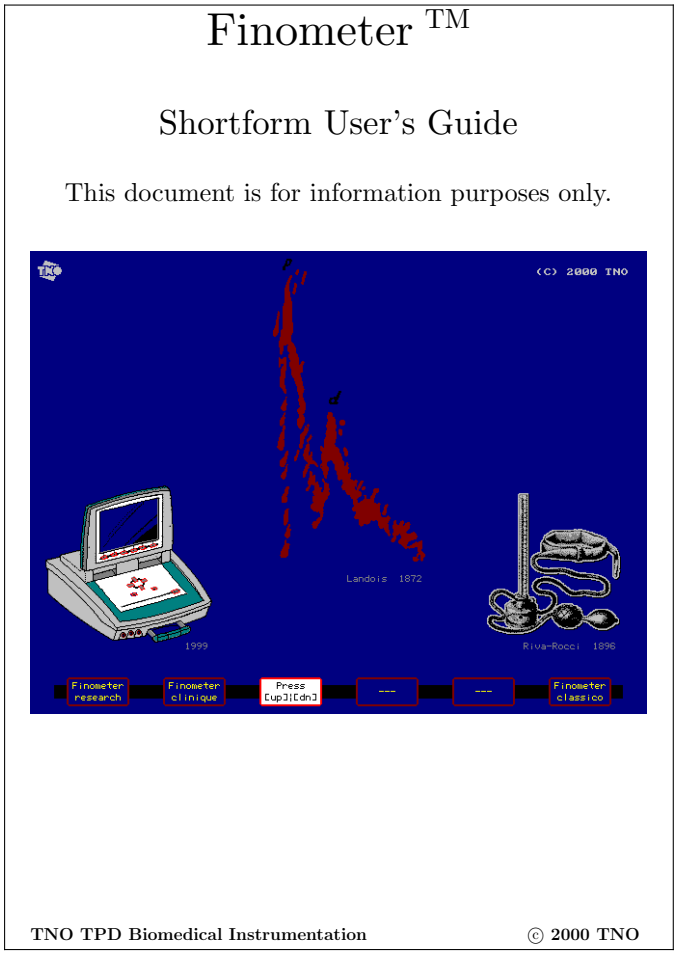


Figure 3 Front page of the Finometer User's Guide. The pulsation in the middle is probably the first blood pressure oscillations ever published (by Landois in 1872). The hollow needle of a syringe was stuck into an artery of a dog and a paper strip moved where the blood droplets hit the table.

The ConTEXt advantage

Is the one year for a first Guide in Word and the three weeks for a second Guide of similar size and complexity in ConTEXt a fair comparison? It is not. For the Guide in ConTEXt we could use some of the pictures and texts of the earlier Portapres Guide. The second Guide was written by the interface designer himself with for him little left to learn about it. For the second Guide we had the example of the first one in terms of content, (CE) demands and outline. The second Guide was essentially produced by two persons, full time, the first by one person part time. But all these factors do not explain the production time factor of 50:3 or 16 times. The use of ConTEXt, we estimate, may have saved us a factor of 2 to 4 in hours spent and weeks to the final product. For a group that is interested in TEX mainly as a tool to achieve blood pressure device goals that is a really significant factor.

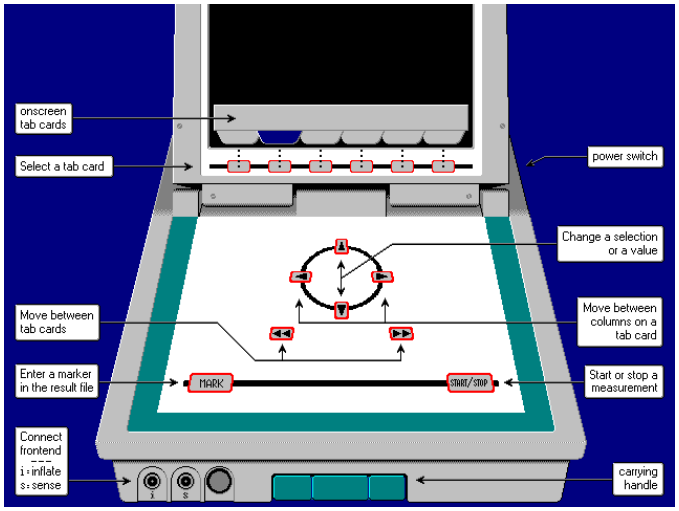


Figure 4 Layout of the Finometer front panel. This graphic is an instruction slide displayed on the device's color LCD screen and is a page in the manual.

Conclusion

One may wonder what lies beyond ConTEXt but for the present time we feel that ConTEXt offers all, perhaps more than all, we need. We developed a confidence in ConTEXt and now use it almost exclusively, also to write our other publications, except when internationally the other side doesn't know ConTEXt. Then we use LaTEX or, when there is no other option, plain ASCII, even Word. ConTEXt is nearly free of charge but that's not the reason. The reason is that it is easy to use (we have never been to three week ConTEXt courses), well documented, powerful, reliable, but above all delivers documented, repeatable beauty.

ConTEXt

Karel H Wesseling

A do–it–yourself thebibliography in CONTEXT

abstract
Moving from LaTeX to CONTEXT is not really simple but to return from CONTEXT to LaTeX would have been equally hard were it not for a publication by Berend de Boer in MAPS 24 explaining how to do LaTeX things in CONTEXT. Only one thing was missing, a do–it–yourself `thebibliography`. Hans Hagen had a solution which is described below.

keywords
non–guru, LaTeX, CONTEXT, bibliography, citation command.

An elaborate introduction

Like many, this author started writing papers in LaTeX using the documentclass 'article'. This gave great satisfaction by the pleasure of seeing things 'in print' even before the contribution was published.

So, why turn to CONTEXT?

Of course, CONTEXT is modern, and extremely powerful, and monolithic, and still TEX. On the other hand CONTEXT is different enough to require some adaptation. The fact is, I needed CONTEXT for another reason and liked it so well that going back to the earlier system seemed undesirable. This became unnecessary as a recent contribution to the MAPS by Berend de Boer[2] appeared easing scientific paper writing authors into moving from LaTeX to CONTEXT. This came as a great and wonderful help to me just in time.

When doing my first paper in CONTEXT, however, I realized that one thing was missing: a tool for the literature references. There appeared to be no obvious replacement for `\begin{thebibligraphy}{99}`. But one can be constructed, and below I demonstrate the use of the simple solution that Hans Hagen (the author of CONTEXT) suggested.

`BiBTeX` is now also available for CONTEXT thanks to Taco Hoekwater[6]. Thus, although bibliographies can be included in your publication through it, it is my personal preference to maintain my literature data base as a file of `\bibitem`'s. For each paper a relevant list of references is extracted from this file and copied into the `\begin{thebibliography}{99}` ... `\end{thebibliography}` section, and often the paper is written with several new references which are only later added to the `\bibitem` data base. In this way the full list of authors, full title and complete reference are incorporated in the text file of the paper, and there is no need to rely on another much larger file not available elsewhere. This arrangement appeared to be quite flexible when working with co–authors outside of my institute.

Thanks to the bibliography solution presented here I can continue this habit and all I have to do in the paper is change every `\bibitem{key}` to `\item[key]`. This is quite doable, even manually, when one realizes that writing an average manuscript may take from three weeks to three months whereas changing a typical 30–item bibliography takes three minutes.

A bibliography implementation in CONTEXT

The implementation is as easy as it is in LaTeX. The bibliography in the manuscript is an enumeration packed in a section. A section without a number is a subject. In `\subject[litrefs]{...}` the subject is referred to via its logical name `litrefs`. In the example below the section is also typeset in two columns with a vertical rule between the columns using the `\setup` command of CONTEXT. If you rather had two columns without a vertical rule you'd type `\setupcolumns[rule=off]`. Note that logical parameters are between square brackets, text parameters are between curly braces. Thus: `\setupcolumns[rule=on]` and `\subject{Bibliography}`. This quickly becomes second nature. The `\startitemize` has a parameter `[n]` which specifies a numbered list of items when printed. The bracketed reference just following `\item` is used in the text of the manuscript. Here is the complete specification.

```
%-------------------------------------------------
\subject[litrefs]{Bibliography}
\setupcolumns[rule=on,balance=yes]
\startcolumns

\startitemize[n]
  \item[lit:Bos] Bos WJW, van Goudoever J, van Montfrans GA,
van den Meiracker AH, Wesseling KH: Reconstruction of brachial
artery pressure from noninvasive finger pressure measurement.
Circulation 1996; {\bf 94}:1870||1875.
  \item[lit:Boer] de Boer B: \LaTeX\ in proper \CONTEXT. MAPS
2000; {\bf 24}:65||92.
  \item[lit:Dol] Dol W, Frambach E: 4\TeX\ for Windows, fifth
edition. ISBN 90||76669||01||5
  \item[lit:Gizdulich1] Gizdulich P, Imholz BPM, van den
Meiracker AH, Parati G, Wesseling KH: Finapres tracking of
systolic pressure and baroreflex sensitivity improved by waveform
filtering. J Hypertens 1996; {\bf 14}:243||250.
  \item[lit:Gizdulich2] Gizdulich P, Prentza A, Wesseling KH:
Models of brachial to finger pulse wave distortion and pressure
decrement. Cardiovasc Res 1997; {\bf 33}:698||705.
  \item[lit:Hoekwater] Hoekwater T: \CONTEXT\ Publication module,
the user documentation. Proceedings Euro\TeX 2001; 61||73.
  \item[lit:Lamport] Lamport L: \LaTeX\ : a document preparation
system. Addison||Wesley 1994, 2nd edition.
  \item[lit:Penaz] Pe\v{n}\'{a}z J:  Photoelectric measurement
of blood pressure, volume and flow in the finger. Digest 10th Int
Conf Med Biol Engng. Dresden, 1973; p 104 (abstract).
  \item[lit:Wesseling4] Wesseling KH, de Wit B, van der Hoeven
GMA, van Goudoever J, Settels JJ: Physiocal, calibrating finger
vascular physiology for Finapres. Homeostasis 1995; {\bf
36}:67||82.
  \stopitemize
\stopcolumns
%-------------------------------------------------
```

Below you can see what this looks like in print.

Bibliography

1. Bos WJW, van Goudoever J, van Montfrans GA, van den Meiracker AH, Wesseling KH: Reconstruction of brachial artery pressure from non-invasive finger pressure measurement. Circulation 1996; **94**:1870–1875.
2. de Boer B: LaTeX in proper CONTEXT. MAPS 2000; **24**:65–92.
3. Dol W, Frambach E: 4TEX for Windows, fifth edition. ISBN 90–76669–01–5
4. Gizdulich P, Imholz BPM, van den Meiracker AH, Parati G, Wesseling KH: Finapres tracking of systolic pressure and baroreflex sensitivity improved by waveform filtering. J Hypertens 1996; **14**:243–250.
5. Gizdulich P, Prentza A, Wesseling KH: Models of brachial to finger pulse wave distortion and pressure decrement. Cardiovasc Res 1997; **33**:698–705.
6. Hoekwater T: CONTEXT Publication module, the user documentation. Proceedings EuroTEX2001; 61–73.
7. Lamport L: LaTeX : a document preparation system. Addison–Wesley 1994, 2nd edition.
8. Peňáz J: Photoelectric measurement of blood pressure, volume and flow in the finger. Digest 10th Int Conf Med Biol Engng. Dresden, 1973; p 104 (abstract).
9. Wesseling KH, de Wit B, van der Hoeven GMA, van Goudoever J, Settels JJ: Physiocal, calibrating finger vascular physiology for Finapres. Homeostasis 1995; **36**:67–82.

The citations

Given the bibliography in this form it is now possible to put citations in the manuscript. In many cases a citation is made by placing a small number high up in the text line, corresponding to the bibliography entry number, like: Knuth[3]. This is achieved by defining a command near the top of the manuscript as follows:

```
\def\Lit[#1]{\unskip\high{\in[lit:#1]}}
```

This defines that a citation is made by typing `\Lit` with one parameter. Typically, you would define your personal commands to begin with a capital letter (here 'L') to avoid possible conflict with proprietary CONTEXT commands which are all lower case. Thus, if you rather used the command `\cite` then don't but use `\Cite` instead. The `\unskip` causes the number to immediately follow any preceding text. The `\high` raises the text between braces that follows. The `\in` makes the reference to the logical item between brackets. This item is `lit:#1`. Thus the internal reference in the bibliography enumeration, for example `lit:Penaz` should be used as `\Lit[Penaz]` without adding `lit:`. This is a safeguard to assure that reference is indeed made to an item in the bibliography and not to a spurious 'Penaz' elsewhere.

Below follows a piece of text from a User's Guide in which the system is used.

```
%-------------------------------------------------
Some technology developed for the Portapres ambulatory finger
blood pressure recorder is also included. For references see
\at{page}[litrefs].

\startitemize[n,packed,broad]
```

```
  \item Continuous monitoring of the finger arterial pressure
waveform with the volume||clamp method of Pe\v{n}\'{a}z
\Lit[Penaz] and the Physiocal criteria of Wesseling
\Lit[Wesseling4], as in Finapres.
  \item Reconstruction of brachial artery pressure waveform and
level from finger pressure via generalized waveform inverse
modeling \Lit[Gizdulich1]\high{,}\Lit[Gizdulich2].
  \item Automatic individual Riva||Rocci arm cuff
return||to||flow pressure level calibration. \Lit[Bos]
  \stopitemize
%-------------------------------------------------
```

In printed form this looks like:

Some technology developed for the Portapres ambulatory finger blood pressure recorder is also included. For references see page 53.

1. Continuous monitoring of the finger arterial pressure waveform with the volume–clamp method of Peňáz[8] and the Physiocal criteria of Wesseling[9], as in Finapres.
2. Reconstruction of brachial artery pressure waveform and level from finger pressure via generalized waveform inverse modeling.[4,5]
3. Automatic individual Riva–Rocci arm cuff return–to–flow pressure level calibration.[1]

But why stop here?

Once put on the right track other needs could perhaps be fullfilled. In the above you noticed the rather awkward solution for two references separated by a comma. Also, some journals like to contract a range of comma separated references like “3,4,5,6” to “3-6”. What we need in addition to `\Lit[]` is a `\Lits[][]` with the significance of Lit plus separator. It took me four trials (and three errors) to come up with a working definition:

```
\def\Lits[#1][#2]{\unskip\high{\in[lit:#1]}\high{#2}}
```

Examples of its use are:

```
Applications of \type{\Lits} are \Lits[Hoekwater][,]\Lit[Penaz] or
\Lits[Bos][-]\Lit[Hoekwater]
```

which looks as follows after processing by CONTEXT:

Applications of `\Lits` are[6,8] or[1-6]

I am nearly certain that the definition for `\Lits[][]` as given above, although working, can be programmed in better ways by someone who really understands programming in TEX and CONTEXT. Sometimes, however, it is important to have something that works when you need it.

Conclusion

I must admit that I would not have found this solution myself even though it is clearly stated in the “LATEX User’s Guide and Reference Manual”[7] (section 4.3.2 Doing It Yourself) that the `\begin{\thebibliography}` is an enumerated list. Even then, creating the `\def\Lit` definition—however simple in hindsight, would not easily have crossed the

mind of this CONTEXT user. Yet, it cannot be called difficult since the process is very clearly explained in the book "4TEX for Windows"[3] (section 16.15 Programming)[1]. The advantage is that the form of the citation is out in the open and can be adapted to your personal requirements quite easily, if need be with a little trial and error.

1. This book remains extremely useful as a compact introduction and reference to TEX even though 4TEX itself is no longer available

beginners

TEX voor thuis

Siep Kroonenberg
siep@elvenkind.com

Dit verhaal richt zich tot mensen die op hun eigen machine TeX aan de praat willen krijgen, zonder hulp van een systeembeheerder of guru-vriendje. We hadden eigenlijk gehoopt op een verhaal van iemand over zijn/haar wederwaardigheden met MikTeX, een Windows TeX distributie die behoorlijk bij de tijd is maar toch meer het accent legt op eenvoudige installatie.

Inplaats daarvan een overzichtje waar je informatie kunt krijgen en wat nadere details over MikTeX en een aantal andere populaire distributies.

Wat zijn de problemen?

TEX alleen maar aan de praat krijgen is tegenwoordig eigenlijk helemaal niet moeilijk meer: de meeste distributies hebben 'out of the box' een bruikbare configuratie.

Installatie is stap één; de installatie gebruiken stap twee. Veel gebruikers vinden een interactieve omgeving hierbij een grote hulp. Die interactieve omgeving moet dan natuurlijk ook 'out of the box' bruikbaar zijn; het is niet de bedoeling dat je eerst een week lang documentatie bij elkaar moet sprokkelen en uitpluizen.

Dat 'sprokkelen' hoort trouwens ook niet tot de Windows- of Macintosh cultuur: je bent gewend dat je wat basis-documentatie krijgt bij je software. Met TeX is dat anders. Er bestaan een aantal uitstekende boeken over LaTeX, zie verderop, maar die vertellen je niet waar je TEX vandaan moet halen, hoe je het moet installeren en hoe je het daarna moet draaien. Om het nog maar niet over aanvullende macropakketten te hebben.

Als je dingen aan je TEX-installatie wilt toevoegen of updaten ontstaan nieuwe problemen. Je moet dan soms akelig veel weten van de interne struktuur van je TeX installatie (zie kader). Daar zal ik nu echter niet op ingaan.

Implementaties en distributies

Voor het converteren van een invoerbestand naar mooi (of lelijk) gedrukte uitvoer is een hele berg programma's en hulpbestanden nodig.

TeX zelf, dat je invoer *compileert* naar een *dvifile*; een previewer om dat dvi-bestand op het scherm te bekijken; dvips om dvi te converteren naar een PostScript printbestand; bibtex voor bibliografiën, makeindex voor indices; hulpprogramma's voor fontbeheer... Als je een versie van TeX gebruikt die rechtstreeks pdf genereert kun je het zonder een previewer en zonder dvips stellen, maar ook dan blijft er genoeg over.

En dan nog de hulpbestanden: fonts, macro's, documentatie, configuratiebestanden... Kortom, je hebt een TeX *distributie* nodig.

Wellicht ten overvloede: de Computer Modern fonts zitten in alle TeX distributies. Er is meestal ook wel een mogelijkheid om Times en Helvetica met TeX te gebruiken.

Enkele overzichten van beschikbare TEX implementaties en distributies:

- `www.tug.org`, link 'Free/Nonfree'
- `www.ams.org/tex/public-domain-tex.html`
- onze eigen NTG site: `www.ntg.nl`, 'TeX, programma's en macropakketten'

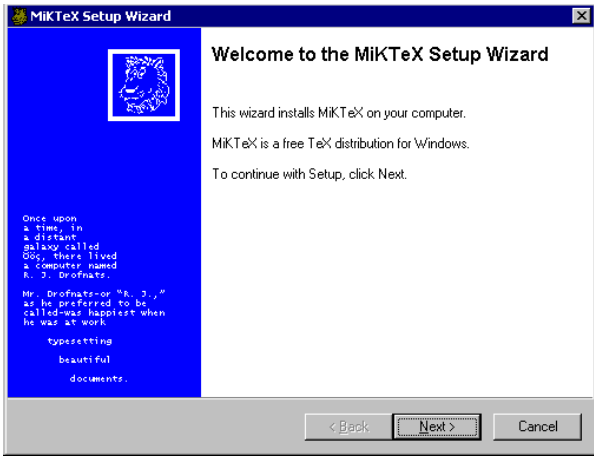
Win32 en MikTeX

Win32 (Windows 9x/NT/2000/ME/XP) gebruikers hebben de keus uit MikTeX en fpTeX, en een aantal commerciële implementaties. fpTeX is de Win32-versie van teTeX, zie kader. fpTeX is ook vertegenwoordigd op de TeX Live cd, zie verderop. Van de twee is MikTeX het kleinst en makkelijkst. 'Klein' is echter maar relatief: voor MikTeX moet je toch nog voor een minimale installatie 23MB downloaden, en na het uitpakken wordt dat 100 MB. Toch een stuk

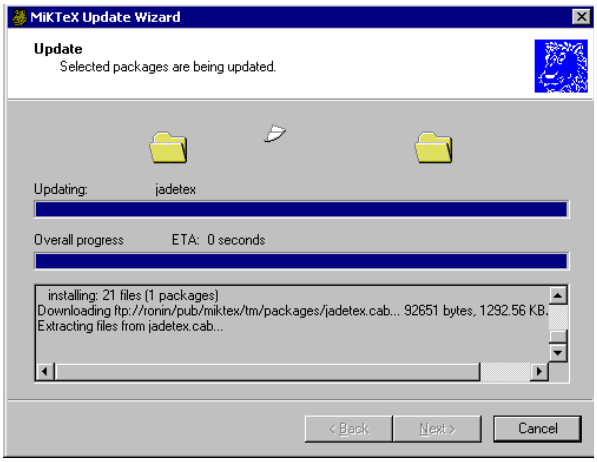
De Standaard: web2c, teTeX, TDS en texmf trees

De Unix Web2c implementatie en daarop gebaseerde teTeX en TeX Live distributies zijn momenteel *de* TEX-implementaties. teTeX is onderdeel van elke mij bekende Linux distributie.

Een onderdeel van deze standaard is de *TDS* of *TEX Directory Structure*. Databestanden (macro's, fonts en configuratiebestanden) staan in een *texmf tree*. Om het zoeken in de vele bestanden en bestandjes van een TEX installatie efficiënter te maken is een directory-structuur hiervoor afgesproken. De onderdelen van TeX kunnen een database raadplegen om een bestand te lokaliseren inplaats van de hele texmf-tree door te moeten zoeken. Op TEX-NL komen echter regelmatig posts langs van mensen die nieuwe bestanden hebben geïnstalleerd die daarna niet worden gevonden: als je met de hand dingen wilt toevoegen moet je echt weten hoe één en ander in elkaar zit.



Figuur 1 MikTeX: Een echt Windows installatie-programma



Figuur 2 MikTeX: Package management

minder dan de honderden MB's nodig om de complete TEX Live op je harde schijf te zetten.

Er is een setup wizard die download en installatie regelt. Je kunt kiezen tussen een 'kleine', grote en heel grote installatie. Installatie wordt stap voor stap uitgelegd op hun site, en met screenshots geïllustreerd.

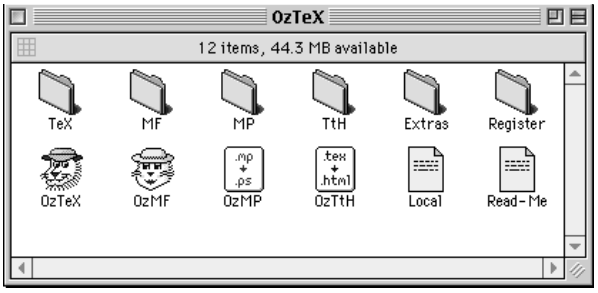
Er zijn interactieve hulpprogramma's voor onderhoudstaken zoals het updaten van packages, de filename database verversen en het genereren van format files.

Er is een manual in html formaat, met vooral informatie over installatie, configuratie en MikTeX-specifieke zaken.

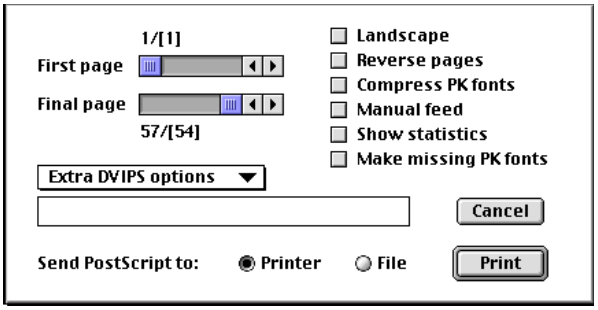
Macintosh

Een goede web pagina voor Macintosh TEX-gebruikers is `www.esm.psu.edu/mac-tex/`. Hier is informatie over de belangrijkere (alle?) Macintosh TEX implementaties. Een populaire, compacte implementatie is ozTeX. Deze implementatie is shareware, geen freeware. Installatie: maak ergens op je harde schijf een ozTeX directory aan. Download alles, of althans alles wat je nodig denkt te hebben, naar deze directory en pak het uit door het naar Stuffit Expander te slepen. Vanuit het programma ozTeX zelf kun je alles doen: compileren (TeX menu), previewen (View menu) en printen (File menu). Ook ozTeX heeft een eigen handleiding.

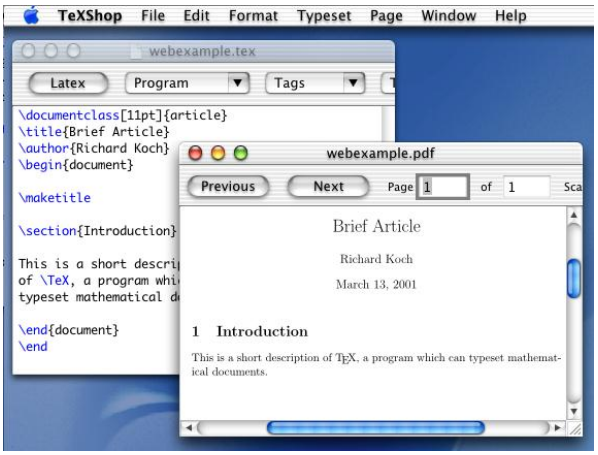
Op de zelfde site kun je CMacTeX vinden, de Mac tegenhanger van teTeX; ook shareware inplaats van gratis.



Figuur 3 ozTeX



Figuur 4 ozTeX: Een dialog box voor printen



Figuur 5 TeXShop: Een frontend voor teTeX onder MacOS X

Zowel OzTeX als CMacTeX zijn ‘carbonized’, d.w.z. dat ze zowel onder Mac OS 8/9 als Mac OS X draaien.

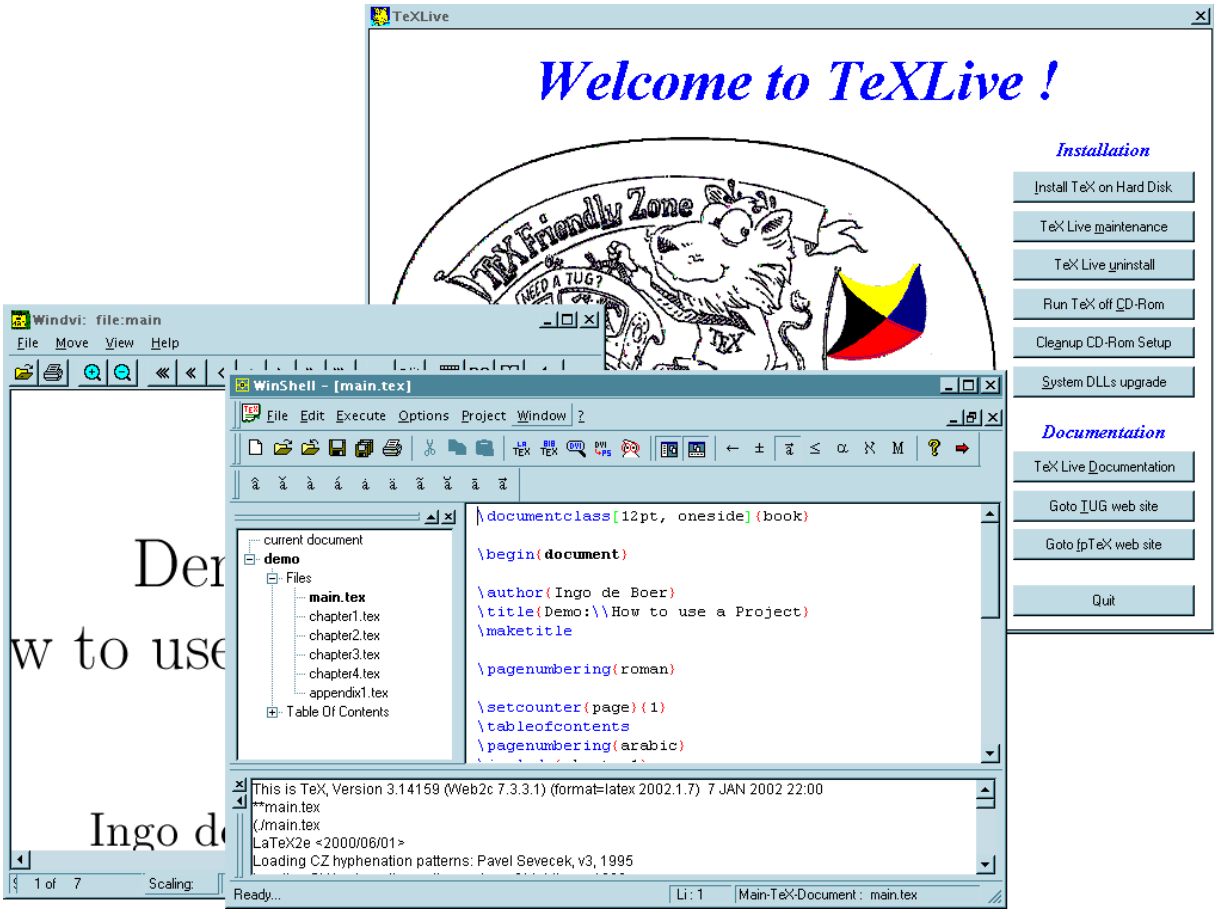
Exclusief voor MacOS X is er TeXShop, wel gratis. Elders in dit nummer meer hierover.

Linux

Voor Linux gebruikers is de keus eenvoudig: alle mij bekende Linux distributies bevatten een onmiddellijk bruikbare installatie van teTeX (dit geldt echter niet noodzakelijk voor de gratis 1-cd probeerversies die vaak gebundeld worden met Linux tijdschriften).

TeX Live

Een zeer uitgebreide web2c installatie is verkrijgbaar in de vorm van de TEX-live cd; zie de betreffende link op `www.tug.org`. Het huidige versienummer is 6. Een cd image staat op CTAN onder `systems/texlive`. Deze cd bevat binaries voor verschillende Unix varianten en voor Win32, en kan op de ondersteunde systemen vanaf cd worden gedraaid. Op Windows wordt hierbij winshell opgestart als editor en interactieve omgeving. Vanhieruit kunnen LaTeX en andere programma’s zoals bibtex en windvi opgestart worden; zie de screenshot.



Figuur 6 De TeX Live cd onder Windows. Het TeX live venster heeft menu opties voor o.a. installatie, onderhoud en TeX draaien vanaf cd. Deze laatste optie startte WinShell op. Het WinShell venster is verdeeld in deelvensters voor editen, voor een logfile en voor project management, voor als je document bestaat uit meerdere bestanden. Het derde venster, de previewer, is opgestart vanuit WinShell.

Hoe krijg ik het in huis

Er is een netwerk van ftp servers voor TEX-gerelateerd materiaal. Hier kun je complete TEX systemen ophalen maar ook aanvullende macro-pakketten, fonts, utilities en documentatie. De Nederlandse is `ftp.ntg.nl/pub/tex-archive`, een Belgische `ftp.belnet.be/packages/CTAN/`. In de root-directory vind je een bestand `FILES.byname` dat je kunt downloaden en op je eigen pc op je gemak doorbladeren om te kijken wat er zoal is. Het bestand `README.structure` geeft een overzicht. Mirror sites staan vermeld in `CTAN.sites`. Sommige CTAN servers hebben ook een zoekmachine (maar ze hebben allemaal dezelfde bestanden): `http://www.dante.de/cgi-bin/ctan-index`. Sommige CTAN servers, o.a. `ftp.dante.de`, directory `/pub/tex` bieden on-the-fly compressie: je kunt, met een command-line ftp, een hele directorystructuur dirnaam in één keer binnenhalen met de opdrachten

```
bin
get dirnaam.zip
```

Een goed assortiment aan TeX-gerelateerde boeken en cd's is te krijgen bij de Duitse Lehmanns Bookshop (`www.lob.de`). Als downloaden geen optie is kun je hier de TeX Live cd bestellen, of cd's van het bovengenoemde CTAN archief.

Documentatie

Enkele LaTeX boeken:

- Leslie Lamport, LaTeX: A Document Preparation System, 2nd Edition, Addison-Wesley, 1994. ISBN: 0-201-52983-1; de originele handleiding
- Helmut Kopka and Patrick W. Daly, A Guide to LATEX: Document Preparation for Beginners and Advanced Users, 3rd Edition, Addison-Wesley, 1999. ISBN: 0-201-39825-7
- Tobias Oetiker's, The Not So Short Introduction to LaTeX2e; beschikbaar op CTAN; zoek naar 'lshort'.

TEX-NL. Vooral als je geen experts bij de hand hebt moet je je beslist op TEX-NL abonneren; instructies vind je op `www.ntg.nl/mail.html`. Hier kun je terecht met vragen over TEX, LaTeX, TEX installaties en allerlei verwante zaken. Als je een probleem hebt dan is de kans heel groot dat iemand op TEX-NL een oplossing aandraagt, niet zelden binnen een uur. Natuurlijk moet je wel eerst proberen het zelf op te lossen – naar de bekende weg vragen leidt alleen maar tot irritatie.

Op je eigen pc. Kijk na installatie vooral op je eigen pc wat er allemaal is geïnstalleerd aan documentatie, hulpprogramma's, fonts en macro's. Bij teTeX/fpTeX is heel veel documentatie te vinden onder `texmf/doc`, o.a. in html-formaat. Wellicht kun je je browser configureren voor dvi, .ps en .pdf files en kun je dit alles met je browser doorbladeren.

macintosh

Mac OS X als TeX platform

Siep Kroonenberg
siepo@cybercomm.nl

abstract

Nu het Macintosh platform tot Unix is bekeerd en ingebouwde ondersteuning heeft voor pdf, bezit de Mac hele goede papieren als een platform voor TeX. Het programma TeXShop is hiervan het bewijs.

Achtergrond

In maart 2001 kwam de langverwachte Mac OS X uit, de nieuwe versie van het verouderde Macintosh besturingssysteem.

De kern van Mac OS X is Darwin, een op FreeBSD gebaseerde versie van Unix. Dit gedeelte is open source en is ook voor het Intel platform beschikbaar. Het grote verschil tussen Darwin en klassieke Unices is systeemconfiguratie met behulp van NetInfo. Wie meer wil weten over Darwin en NetInfo kan zijn zoektocht beginnen op de Apple site, in het bijzonder op `www.opensource.apple.com`.

OS X bouwt ook voort op Next, een Unix-achtig grafisch besturingssysteem dat Steve Jobs mee heeft genomen naar Apple toen hij daar terugkwam. Next kreeg een uitstekende pers, maar heeft het niet gehaald. Eén van de pronkstukken van Next was Display PostScript, schermaansturing gebaseerd op PostScript.

OS X gebruikt voor zijn schermaansturing geen PostScript maar (o.a.) pdf, dat voor veel situaties de opvolger is van PostScript. Eén van de meegeleverde hulpprogramma's is dan ook een pdf viewer, hoewel deze lang niet de functionaliteit heeft van Acrobat Reader.

De onderdelen voor grafische aansturing en de gebruikersinterface worden losjes samengevat onder de term Aqua.

De gebruikersinterface voelt als een Mac, ondanks de totaal andere basis. Apple heeft hard gewerkt om een Unix versie te maken die geschikt is voor eindgebruikers, en het algemene oordeel is dat dat prima is gelukt. Met het verschijnen van versie 10.1 in september zijn ook de ergste kinderziekten verholpen.

TeXShop: teTeX met een grafische schil

Het programma TeXShop (`darkwing.uoregon.edu/~koch/texshop/texshop.html`) maakt efficïent gebruik van de mogelijkheden van OS X: het is een TeX implementatie met de standaard Unix teTeX onder de motorkap, en met een pdf previewer die het eigenlijke werk laat doen door het besturingssysteem. Het omvat ook een editor met TeX syntax highlighting. Naast LaTeX ondersteunt TeXShop ook Context. Er is een 'point-and-click' installatieprogramma dat zowel TeXShop zelf als teTeX en Ghostscript voor zijn rekening neemt. Als viewer is Ghostscript misschien niet meer zo relevant, maar voor allerlei conversies blijft Ghostscript onmisbaar.

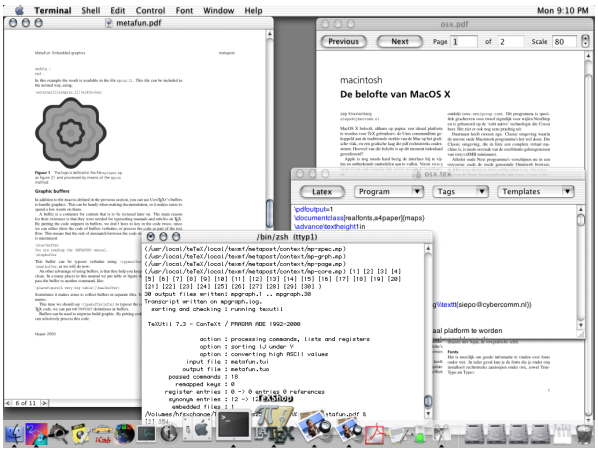
De eerste screenshot heb ik alweer een half jaar geleden gemaakt voor de NTG bijeenkomst in Gent. Het laat de TeXShop editor en -previewer zien, en ook de previewer van Mac OS X zelf, en een terminal venster waarin zojuist een Context run heeft gedraaid. Maar, zoals gezegd, kan tegenwoordig Context ook vanuit TeXShop aangestuurd worden.

Aanbevolen: de Mac TeX/LaTeX Web Site; `www.esm.psu.edu/mac-tex/`. Hier is ook een link naar een 'TeX on Mac OS X' mailing list.

Andere Mac OS X software

Natuurlijk willen de meeste mensen meer dan alleen maar TeX-en met hun computer. Gebruikers van OS X kunnen kiezen uit software van zeer gevarieerde herkomst:

- De belangrijkste Macintosh programma's zijn nu geschikt gemaakt voor OS X. Dit geldt ook voor professionele grafische software: Illustrator, FreeHand, CorelDRAW, InDesign en nu ook PhotoShop.



□ Wat Unix software betreft: Apple levert zelf al veel mee, waaronder de GNU C compiler – onder de naam cc, een actuele Perl, en console versies van de klassieke Unix editors vi en emacs. Er zijn een aantal projecten die grootschalig Unix en X software porten naar Mac OS X. Ik noem gnu-darwin (`gnu-darwin/sourceforge/net`), fink (`fink.sourceforge.net`) en osxgnu (`ww1.osxgnu.org`). De eerste twee zijn vooral bedoeld voor mensen die vertrouwd zijn met Unix en de commandline.

Echte OS X ports, die grafische programma's onder Aqua laten draaien, hebben veel meer voeten in aarde dan wat deze port projecten doen. Het is daarom handig dat X en Aqua zij aan zij kunnen draaien voor programma's waar wel een X- maar geen Aqua versie van bestaat. Vim (`homepage.mac.com/fisherbb`) en emacs (`emacs-on-aqua.sourceforge.net`) zijn echter wel beschikbaar met een Aqua interface.

□ Een aantal Next programma's zijn nu uitgebracht voor OS X. Vooral interessant zijn de OmniWeb web browser en het OmniGraffle pakket voor diagrammen tekenen van Omni Group (`www.omnigroup.com`) en de Stone Studio grafische suite van Stone Design(`www.stone.com`).

□ Oude Macintosh programma's kunnen in de zgn. 'Classic' omgeving gedraaid worden. De Classic omgeving is een complete virtuele machine. Dit werkt behoorlijk goed.

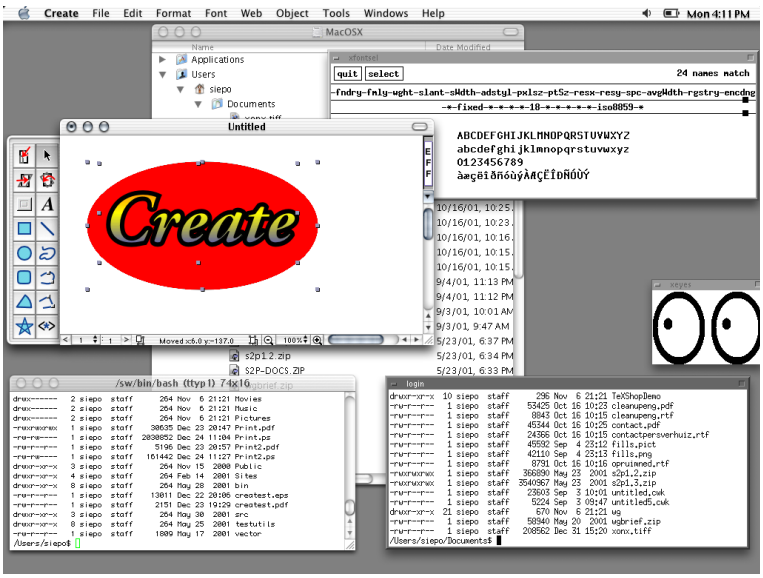
□ Je kunt zelfs de meeste Windows programma's probleemloos draaien onder Virtual PC. Dit programma creëert virtuele machines waarop een Windows naar keuze kan worden geïnstalleerd of al is voorgeïnstalleerd.

De tweede screenshot laat een aantal Aqua- en X programma's zien. De Aqua vensters zijn te herkennen aan de horizontale strepen en de glazen knopjes: op de achtergrond een Finder (filemanager) venster en daaroverheen Create, de tekenmodule uit de reeds genoemde Stone Studio. De Unix X vensters zijn te herkennen aan een soberder omlijsting. Rechtsboven xfontsel, dat je helpt met het identificeren van fonts onder X, en daaronder xeyes: twee ogen die je cursor volgen. Links- en rechtsonder een Aqua- en een X versie van een terminal-venster.

Slotopmerkingen

Mac OS X is ontworpen voor eindgebruikers. Dat maakt het voor iedereen mogelijk er snel in wegwijs te raken. Eindgebruikers zullen liever de kale Unix commandline met rust laten, maar programmeurs slagen er in Unix functionaliteit van OS X toegankelijk te maken door er een grafische schil omheen te bouwen. TeXShop is daar een goed voorbeeld van.

BeOS, Atari, Next en OS2 waren uitstekende systemen die het niet hebben gehaald. De belangrijkste factor in deze mislukkingen was het ontbreken van voldoende software. Maar Mac OS X slaat kennelijk aan en lijkt een zonnige toekomst tegemoet te gaan.



Siep Kroonenberg
siep@elvenkind.com

configuration

Juggling texmf trees

abstract
Texmf trees can make a TEX installation more maintainable. With creative use of environment variables, it is possible to run different versions and different configurations in different xterm- or console windows.

keywords
texmf trees, configuration, environment variables, file searching

Texmf trees are being cursed for making life unnecessarily complicated. There may be some truth to this, but they also let you break your TEX installation in pieces which can be used and maintained independently:

- Keep your own enhancements in a separate tree, which will be untouched when you upgrade the main tree.
- Give a fast-changing package such as Context a tree of its own. If you want to, you can run two Context versions side-by-side.
- Keep your own installation compact but hook up the TEX Live cd when you need some esoteric feature.
- Unhook your local tree(s) to check whether your journal submission compiles on a vanilla TEX system.

Configuration with TEXMFCNF and `texmf.cnf`

The core components of a TEX system all find their files by the same method: they read one or more `texmf.cnf` configuration files, which, among other things, tell them where to find their files.

TEX data files (macros, format files, fonts etc.) are stored in one or more `texmf` directory trees, which have a more or less fixed internal organization. TEX is configured to always search the current directory for your own files.

teTeX and TEX Live are designed to run 'out of the box'. All the programs have a compiled-in default location for looking for configuration files *relative to their own location*. If you just add the appropriate binaries directory to your path then you will have a runnable system without further manual configuration. Let us see how this magic is accomplished:

This is the relevant fragment of the `texmf.cnf` of the current (january 25 2002) teTeX beta; the version for the TEX Live 6 cd is very similar:

```
% The main tree, which must be mentioned in $TEXMF, below:
TEXMFMAIN = $SELFAUTOPARENT/texmf
% A place for local additions to a "standard" texmf tree.
TEXMFLOCAL = $SELFAUTOPARENT/texmf-local

% User texmf trees can be catered for like this...
HOMETEXMF=$HOME/texmf
```

```
% A place where texconfig stores modifications (instead of the
% TEXMFMAIN tree). texconfig relies on the name, so don't change it.
VARTEXMF = $SELFAUTOPARENT/texmf-var

% Now, list all the texmf trees. If you have multiple trees,
% use shell brace notation, like this:
TEXMF = {$HOMETEXMF,!!$VARTEXMF,$TEXMFLOCAL,!!$TEXMFMAIN}

% Where to look for ls-R files.  There need not be an ls-R in the
% directories in this path, but if there is one, Kpathsea will use it.
TEXMFDBS = $TEXMF
```

That is, there are four `texmf` trees configured: the main tree, `TEXMFMAIN`; two trees for local and user files, and one tree for generated files such as format files or font bitmaps.

The division in 'local' and 'user' makes sense for a shared TEX installation but if you have a TEX installation just for yourself you may want to dispense with one or the other, or reserve one of them for a special purpose.

The order in which `TEXMF` enumerates the trees determines the order in which trees are being searched. If you don't like a version of a file in the main tree then you can put another version in the local or home tree, which will be found before the original version.

The '!!' in front of two of the four tree names indicates that these directories should be searched *only* via the corresponding filename database, which bears the name `ls-R` and resides at the root of the tree. For the other trees, which presumably aren't very large, ordinary file searching is allowed[1].

The variable `SELFAUTOPARENT` gets its value dynamically: if the binaries are in `/usr/local/teTeX/bin/linux`, then `SELFAUTOPARENT` will be the grandparent directory, i.e. `/usr/local/teTeX`. As long as the texmf trees and the binaries are in the same relative position, TEX will be able to find its files.

`texmf.cnf` is thoroughly commented. I recommend that you study it closely. You can find additional information in the kpathsea documentation, which should be present on your system in pdf- and other formats. Also study the structure of the (main) texmf tree. Your local tree doesn't have to be this elaborate, though.

Using environment variables for versioning

For a harddisk installation, you may simply accept the directory structure as it is configured. For a cd installation however, the default value for `VARTEXMF` and `TEXMFLOCAL`, `$SELFAUTOPARENT/texmf-var` and `$SELFAUTOPARENT/texmf-local`, point to directories on the cd so you'll probably want to change them. You can simply overrule them by setting environment variables:

```
VARTEXMF=/var/tmp/texmf-var
export VARTEXMF
```

for Unix (Bourne-compatible shell), or

```
set VARTEXMF=/var/tmp/texmf-var
```

1. The reason not to search on disk is efficiency, but if you disallow searching on disk then you *must* maintain a filename database. The command for this is `mktexlsr`. Without parameters, the command will try to (re)generate the `ls-R` file for all trees listed in TEXMFDBS. You can also specify a texmf tree as parameter.

for DOS/Windows.

But you may have all sorts of reasons to want the trees in other locations than the preconfigured ones.

Context users may appreciate the possibility to reserve an entire tree, say the local tree, for Context. This greatly simplifies upgrading. It also makes it easy to have different versions around:

```
set TEXMFLOCAL=c:/cnstable
path ...;c:\cnstable\context\perltk;...
```

activates the stable version, and

```
set TEXMFLOCAL=c:/cnbeta
path ...;c:\cnbeta\context\perltk;...
```

activates the latest beta. You may not even need to add `context/perltk` to your path, depending on your TEX version.

One thing to watch out for: create subdirectories `cnstable/web2c` and `cnbeta/web2c`, and immediately move or copy your freshly generated Context format files there, otherwise they will overwrite each other.

So now you can switch back and forth between Context versions simply by changing an environment variable.

Hooking and unhooking trees

TEX installations tend to be unpleasantly big. You may not be using a lot of fancy features yourself but you may have colleagues who like to use every package they can get their hands on. A solution: add a cd-based texmf tree, from e.g. the TEX Live- or the 4TeX5 cd, to compile their papers.

In this example I use a second `texmf.cnf` file consisting of just the lines

```
TEXMFCD=d:/texmf
TEXMF = {$HOMETEXMF,!!$VARTEXMF,$TEXMFLOCAL,!!$TEXMFMAIN,!!$TEXMFCD}
```

So the cd tree is added last, and '!!' ensures that the filename database is used. The first line could have been replaced with an environment variable. In fact, if `TEXMFCD` is defined as an environment variable then the setting in this file won't take effect.

To make sure that TEX reads the new `texmf.cnf`, add its directory to the `TEXMFCNF` environment variable, e.g.:

```
set TEXMFCNF=<directory of the new texmf.cnf>;c:/texmf/web2c
```

This works! This is what happens: TEX will first read the environment, then the new `texmf.cnf` and finally the original `texmf.cnf`. *Once a variable is set, subsequent settings won't overrule it.* So the setting for `TEXMF` in the new `texmf.cnf` will remain in effect but we let the original `texmf.cnf` worry about all the other settings.

Unhooking trees is exactly analogous: create a `texmf.cnf` file with a line

```
TEXMF = {!!$VARTEXMF,!!$TEXMFMAIN}
```

and make sure, as above, that the new `texmf.cnf` is read before the main one.

As indicated at the beginning, when you are about to send your paper elsewhere then this is a good way to check whether your paper compiles on a vanilla system. If it doesn't, copy stuff from the local tree(s) to your working directory until it compiles cleanly.

Testing with kpsewhich

Be sure to test whether the right versions will be picked up from the right trees. Type e.g.

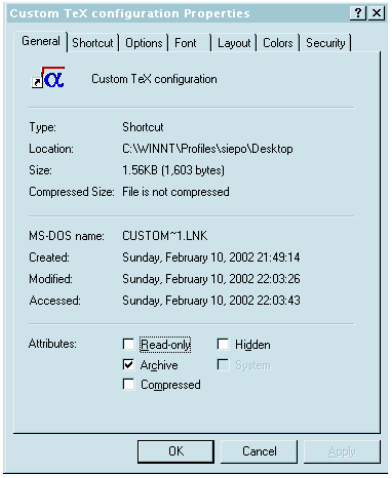
```
kpsewhich cont-en.efmt
```

and you'll see which version of the Context format file is actually going to be used.

Using scripts

Of course, you'll want to put these environment settings into scripts or batchfiles.

Under DOS and at least some versions of Windows, you can set environment variables in a batchfile, and they will stay set after the batchfile has run its course. Under more recent versions of Windows, this may no longer be the case. Anyhow, you can still specify an initialization batchfile for a custom console window:



The icon and the name will show up in the titlebar of the console window.

Under Unix, environment settings defined in a shell script will certainly get lost, unless you 'source' your script, which we shall name `texcd`:

```
source texcd
```

This lets the script run in the current process instead of in a child process. With Bash, you can write '.' instead of 'source':

```
. texcd
```

If you still don't like the extra syntax, you can define an alias:

```
alias tcd='source texcd'
```

All terminal emulators that I know of also let you specify a title to be displayed on the titlebar. If you use bash, try the following:

```
PROMPT_COMMAND="echo -e -n \"\033]0;cdtex \$PWD \07\""
```

to get the tag 'cdtex' and the current directory on your titlebar.

Hans Hagen
Pragma ADE
Hasselt
pragma@wxs.nl

ConTEXt MathML

Introduction

It is a well known fact that TEX can do a pretty good job on typesetting math. This is one reason why many scientific articles, papers and books are typeset using TEX. However, in these days of triumphing angle brackets, coding in TEX looks more and more out of place.

From the point of view of an author, coding in TEX is quite natural, given that some time is spent on reading the manuals. This is (I'm told) because not only the natural flow of the definition suits the way mathematicians think, but also because the author has quite some control over the way his thoughts end up on paper. It will be no surprise that switching to a more restricted way of coding, which also demands more keystrokes, is not on forehand considered to be better.

There are however circumstances that one wants to share formulas (or formula–like specifications) between several applications, one of which is a typesetting engine. In that case, a bit more work now, later saves you some headaches due to keeping the different source documents in sync.

As soon as coding math in angle brackets is discussed, those in favour stress that coding can be eased by using appropriate editors. Here we encounter a dilemma. For optimal usage, one should code in terms of content, that is, the principles that are expressed in a formula. Editors are not that strong in this area, and if they would be, editing would be not that much different from traditionally editing formulas: just keying in ideas using code that at first sight looks obscure. A more graphical oriented editor can help authors to compose formulas, but the underlaying coding will mainly be in terms of placing glyphs and boxes, and as a result the code will hardly be usable in other applications.

So either we code in terms of concepts, which permits sharing code among applications, and poses strong limitations on the influence of authors on the visual appearance. Or we use an interactive editor to fine tune the appearance of a formula and take for granted that reuse will be minimal or suboptimal.

In the following chapters we will discuss the mathematical language MATHML in the perspective of typography. As a typesetting vehicle, we have used CONTEXT. However, the principles introduced here and the examples that we provide are independent of CONTEXT. For a more formal exploration we recommend the MATHML specification.

This document is dedicated to all those CONTEXT users who like typesetting math. I'm sure that my father, who was a math teacher, would have liked proofreading this document. His absence was compensated by Tobias Burnus, Wang Lei, Ton Otten, and members of the CONTEXT mailing list who carefully read the text, corrected the errors in my math, tested the functionality, and made suggestions. Any remaining errors are mine.

What is MATHML

Backgrounds

MATHML showed up in the evolving vacuum between structural SGML markup and presentational HTML. Both SGML and HTML can be recognized by angle brackets. The disadvantage of SGML was that is was so open ended, that general tools could hardly be

developed. HTML on the other hand was easy to use and became extremely popular and users as well as software vendors quickly spoiled the original ideas and created a mess. SGML never became really popular, but thanks to HTML people became accustomed to that kind of notation. So, when XML came around as a more restricted cousin of SGML, the world was kind of ready for it. It cannot be denied that by some clever marketing many of today's users think that they use something new and modern, while we are actually dealing with something from the early days of computing. A main benefit of XML is that it brought the ideas behind SGML (and medium neutral coding in general) to the users and at the same time made a major cleanup of HTML possible.

About the same time, MATHML was defined, both to bring math to the WWW, and to provide a way of coding math that will stimulate sharing the same code between different applications. At the end of 2000, the MATHML version 2 draft became a recommendation.

Now, imagine that we want to present a document on the internet using a format like HTML, either for viewing or for aural reproduction. Converting text and graphics is, given proper source coding, seldom a problem, but converting formulas into some angle bracket representation is more tricky. A way out of this is MATHML's presentational markup.

$$a + b = c$$

This simple formula, when coded in TEX, looks like:

```
$$ a + b = c $$
```

In presentational MATHML we get:

```
<math>
  <mrow>
    <mi> a </mi>
    <mo> + </mo>
    <mi> b </mi>
    <mo> = </mo>
    <mi> c </mi>
  </mrow>
</math>
```

In presentational MATHML, we use mostly begintags (`<mi>`) and end tags (`</mi>`). The *row* element is the basic building block of a formula. The *mi* element specifies a math identifier and *mo* is used for operators. In the process of typesetting, both are subjected to interpretation in order to get the best visualization.

Converting TEX code directly or indirectly, using the DVI output or even in–memory produced math lists, has been one of the driving forces behind presentational MATHML and other math related DTD's like EUROMATH. One may wonder if there are sound and valid reasons for going the opposite way. You can imagine that a converter from TEX to MATHML produces *menclose*, *mspace*, *mstyle* and other elements that can have many spacing related attributes, but I wonder if any author is willing to think in those quantities. Visual editors of course are good candidates for producing presentational MATHML.

But wouldn't it be more efficient if we could express ideas and concepts in such a way that they could be handled by a broad range of applications, including a typesetting engine? This is why, in addition to presentational MATHML, there is also content MATHML. The previous formula, when coded in such a way, looks like:

```
<math>
  <apply> <eq/>
    <apply> <plus/>
      <ci> a </ci>
```

```
      <ci> b </ci>
    </apply>
    <ci> c </ci>
  </apply>
</math>
```

This way of defining a formula resembles the so called polish (or stackwise) notation. Opposite to presentational markup, here a typesetting engine has to find out in what order and what way the content has to be presented. This may seem a disadvantage, but in practice implementing content markup is not that complicated. The big advantage is that, once we know how to typeset a concept, TEX can do a good job, while in presentational markup much hard coded spacing can spoil everything. One can of course ignore specific elements, but it is more safe to start from less and enhance, than to leave away something with unknown quantities.

Instead of using hard coded operators as in presentational MATHML, content markup uses empty elements like `<plus/>`. Many operators and functions are predefined but one can also define his own, which is not entirely en par with the concept.

Of course the main question to be answered now is to what extent the author can influence the appearance of a formula defined in content markup. Content markup has the advantage that the results can be more consistent, but taking away all control is counterproductive. The MATHML level 2 draft mentions that this level covers most of the pre university math. If so, that is a proper starting point, but especially educational math often has to be typeset in such ways that it serves its purpose. Also, (re)using the formulas in other applications (simulators and alike) is useful in an educational setting, so content markup is quite suitable.

How do we combine the advantages of content markup with the wish of an author to control the visual output and at the same time get an as high as possible typeset result. There are several ways to accomplish this. One is to include in the document source both the content markup and the TEX specific code.

```
<math>
  <semantics>
    <apply> <eq/>
      <apply> <plus/>
        <ci> a </ci>
        <ci> b </ci>
      </apply>
    </apply>
    <ci> c </ci>
    <annotation encoding="TeX">a+b=c</annotation>
  </semantics>
</math>
```

The *annotation* element is one of the few that is permitted inside the *math* element. In this example, we embed pure TEX code, which, when enabled is typeset in math mode. It will be clear that for a simple formula like this one, such redundant coding is not needed, but one can imagine more complicated formulas. Because we want to limit the amount of work, we prefer just content markup.

Two methods

The best way to learn MATHML is to key in formulas, so that is what we did as soon as we started adding MATHML support to CONTEXT. In some areas, MATHML provides much detail (many functions are represented by elements) while in other areas one has to fall back on the more generic function element or a full description. Compare the following definitions:

```
<math> <apply> <sin/> <ci> x </ci> </apply> </math>
<math> <mrow> <mo> sin </mo> <mi> x </mi> </mrow> </math>
```

We prefer the first definition because it is more structured and gives more control over the result. There is only one ‘unknown’ quantity, x, and from the encapsulating element *ci* we know that it is an identifier.

$$\sin x$$

$$\sin x$$

In the content example, from the *apply sin* we can deduce that the following argument is an operand, either an *apply*, or an *ci* or *cn*. In the presentational alternative, the following elements can be braces, a math identifier, a row, a sequence of identifiers and operators, etc. There, the look and feel is hard coded.

```
<?context-mathml-directive function reduction no ?>
```

This directive, either issued in the XML file, or set in the style file, changes the appearance of the function, but only in content markup. It is because of this feature, that we favour content markup.

$$\sin(x)$$

$$\sin x$$

Does this mean that we can cover everything with content markup? The answer to this is still unclear. Consider the following definition.

Here we combine several cases in one formula by using $\pm$ and $\mp$ symbols. Because we only have *plus* and *minus* elements, we have to revert to the generic function element *fn*. We show the complete definition of this formula.

```
[file pc-i-380.xml does not exist]
```

The MATHML parser and typesetting engine have to know how to handle these special cases, because the visualization depends on the function (or operator). Here both composed signs are treated like the plus and minus signs, but in other cases an embraced argument may be needed. Each special case needs a specific handler.

Presentational markup

If a document contains presentational MATHML, there is a good chance that the code is output by an editor. Here we will discuss the presentation elements that make sense for users when they want to manually code presentational MATHML. In this chapter we show the default rendering, later we will discuss options.

Although much is permitted, we advise to keep the code as simple as possible, because then TEX can do a rather good job on interpreting and typesetting it. Just let TEX take care of the spacing.

mi, mn, mo

Presentational markup comes down to pasting boxes together in math specific ways. The basic building blocks are these three character elements.

$$x = 5$$

```
<math>
  <mrow>
    <mi> x </mi> <mo> = </mo> <mn> 5 </mn>
  </mrow>
</math>
```

mi	identifier	normally typeset in an italic font
mn	number	normally typeset in a normal font
mo	operator	surrounded by specific spacing

mrow

The previous example demonstrated the use of *mrow*, the element that is used to communicate the larger building blocks. Although this element from the perspective of typesetting is not always needed, by using it, the structure of the formula in the document source is more clear.

msup, msub, msubsup

Where in content markup super and subscript are absent and derived from the context, in presentational markup they are quite present.

$$x_1^2$$

```
<math>
  <msup>
    <msub> <mi> x </mi> <mn> 1 </mn> </msub>
    <mn> 2 </mn>
  </msup>
</math>
```

$$x_1^2$$

```
<math>
  <msubsup>
    <mi> x </mi>
    <mn> 1 </mn>
    <mn> 2 </mn>
  </msubsup>
</math>
```

Watch the difference between both definitions and appearances. You can influence the default behaviour with processing instructions.

mfrac

Addition, subtraction and multiplication is hard coded using the *mo* element with $+$, $-$, and $\times$ (or nothing). You can use $/$ for division, but for more complicated formulas you have to fall back on fraction building. This is why MATHML provides the *mfrac*.

$$\frac{x+1}{y+1}$$

```
<math>
  <mfrac>
    <mrow> <mi> x </mi> <mo> + </mo> <mn> 1 </mn> </mrow>
    <mrow> <mi> y </mi> <mo> + </mo> <mn> 1 </mn> </mrow>
  </mfrac>
</math>
```

You can change the width of the rule, but this is generally a bad idea. For special purposes you can set the line thickness to zero.

$$x \geq 2$$
$$y \leq 4$$

```
<math>
  <mfrac linethickness="0">
    <mrow> <mi> x </mi> <mo> &geq; </mo> <mn> 2 </mn> </mrow>
    <mrow> <mi> y </mi> <mo> &leq; </mo> <mn> 4 </mn> </mrow>
  </mfrac>
</math>
```

mfenced

Braces are used to visually group sub–expressions. In presentational MATHML you can either hard code braces, or use the *mfenced* element to generate delimiters automatically.

```
<mo>(</mo> <mi> x </mi> <mo> + </mo> <mn> 1 </mn> <mo>)</mo>
<mfenced>  <mi> x </mi> <mo> + </mo> <mn> 1 </mn> </mfenced>
```

In CONTEXT, as much as possible, the operators and identifiers are interpreted, and when recognized treated according to their nature.

$$\frac{(x+1)\,(x-1)}{(y+1)\,(y-1)}$$

```
<math>
  <mfrac>
    <mrow>
      <mfenced> <mi> x </mi> <mo> + </mo> <mn> 1 </mn> </mfenced>
      <mfenced> <mi> x </mi> <mo> - </mo> <mn> 1 </mn> </mfenced>
    </mrow>
    <mrow>
      <mfenced> <mi> y </mi> <mo> + </mo> <mn> 1 </mn> </mfenced>
      <mfenced> <mi> y </mi> <mo> - </mo> <mn> 1 </mn> </mfenced>
    </mrow>
  </mfrac>
</math>
```

The braces adapt their size to the content. Their dimensions also depend on the way math fonts are defined. The standard TEX fonts will give the same height of braces around x and y, but in other fonts the y may invoke slightly larger ones.

$$(x,y,z)$$

```
<math>
  <mfenced open="(" close=")" separators=",">
    <mi> x </mi> <mi> y </mi> <mi> z </mi>
  </mfenced>
</math>
```

The separators will adapt their size to the fenced content.

$$\left[\frac{1}{x}\middle|\frac{1}{y}\middle|\frac{1}{z}\right]$$

```
<math>
  <mfenced open="[" close="]" separators="|">
    <mfrac> <mn> 1 </mn> <mi> x </mi> </mfrac>
    <mfrac> <mn> 1 </mn> <mi> y </mi> </mfrac>
    <mfrac> <mn> 1 </mn> <mi> z </mi> </mfrac>
  </mfenced>
</math>
```

msqrt, mroot

The shape and size of roots, integrals, sums and products can depend on the size of the content.

$$\sqrt{b}$$

```
<math>
  <msqrt>
    <mi> b </mi>
  </msqrt>
</math>
```

$$\sqrt[2]{b}$$

```
<math>
  <mroot>
    <mi> b </mi>
    <mn> 2 </mn>
  </mroot>
</math>
```

$$\sqrt[2]{\frac{1}{b}}$$

```
<math>
  <mroot>
    <mfrac> <mn> 1 </mn> <mi> b </mi> </mfrac>
    <mn> 2 </mn>
  </mroot>
</math>
```

$$\sqrt[3]{\frac{1}{a+b}}$$

```
<math>
  <mroot>
    <mfrac>
      <mn> 1 </mn>
      <mrow> <mi> a </mi> <mo> + </mo> <mi> b </mi> </mrow>
    </mfrac>
    <mn> 3 </mn>
  </mroot>
</math>
```

mtext

If you put text in a *mi* element, it will come out rather ugly. This is due to the fact that identifiers are (at least in TEX) not subjected to the kerning that is normally used in text. Therefore, when you want to add some text to a formula, you should use the *mtext* element.

$$\frac{SomeText}{\text{Some Text}}$$

```
<math>
  <mfrac>
    <mi> Some Text </mi>
    <mtext> Some Text </mtext>
  </mfrac>
</math>
```

mover, munder, munderover

Not all formulas are math and spacing and font rules may differ per discipline. The following formula reflects a chemical reaction.

$$2H_2 + O_2 \xrightarrow[\text{explosion}]{} 2H_2O$$

```
<math>
  <mrow>
    <mrow>
      <mn> 2 </mn>
      <msub> <mtext> H </mtext> <mn> 2 </mn> </msub>
    </mrow>
    <mo> + </mo>
    <msub> <mtext> O </mtext> <mn> 2 </mn> </msub>
    <munder>
      <mo> &RightArrow; </mo>
      <mtext> explosion </mtext>
    </munder>
    <mrow>
      <mn> 2 </mn>
      <msub> <mtext> H </mtext> <mn> 2 </mn> </msub>
      <mtext> O </mtext>
    </mrow>
  </mrow>
</math>
```

The *munder*, *mover* and *munderover* elements can be used to put symbols and text or formulas on top of each other. When applicable, the symbols will stretch themselves to span the natural size of the text or formula.

ms

This is a bit weird element. It behaves like *mtext* but puts quotes around the text.

$$\frac{\text{"Some Text"}}{\text{Some Text}}$$

```
<math>
  <mfrac>
    <ms> Some Text </ms>
    <mtext> Some Text </mtext>
  </mfrac>
</math>
```

You can specify the left and right boundary characters, either directly or (preferably) using entities like *quot*.

+A Famous Quotation+

```
<math>
  <ms lquote="+" rquote="+"> A Famous Quotation </ms>
</math>
```

menclose

This element is implemented but it is such a weird element that we don't yet describe it here.

merror

There is not much chance that this element will end up in a math textbook, unless the typeset output of programs is part of the story.

$$\text{Are you kidding?}\quad \frac{1+x}{0}$$

```
<math>
  <merror>
    <mtext> Are you kidding?   </mtext>
    <mfrac>
      <mrow> <mn> 1 </mn> <mo> + </mo> <mi> x </mi> </mrow>
      <mn> 0 </mn>
    </mfrac>
  </merror>
</math>
```

mmultiscripts, mprescripts

This element is one of the less obvious ones. The next two examples are taken from the specification. The *multiscripts* element takes an odd number of arguments. The second and successive child elements alternate between sub- and superscript. The empty element *none* —a dedicated element *mnone* would have been a better choice— serves as a placeholder.

$$R_i{}^j{}_{kl}$$

```
<math>
  <mmultiscripts>
    <mi> R </mi>
    <mi> i </mi>
    <none/>
    <none/>
    <mi> j </mi>
    <mi> k </mi>
    <none/>
    <mi> l </mi>
    <none/>
  </mmultiscripts>
</math>
```

The *mmultiscripts* element can also be used to attach prescripts to a symbol. The next example demonstrates this. The empty *prescripts* element signals the start of the prescripts section.

$${}^{427}Qb_4$$

```
<math>
  <mmultiscripts>
    <mi> Qb </mi>
    <mn> 4 </mn>
    <none/>
    <mprescripts/>
    <mn> 427 </mn>
    <none/>
  </mmultiscripts>
</math>
```

mspace

Currently not all functionality of the *mspace* element is implemented. Over time we will see what support is needed and makes sense, especially since this command can spoil things. We only support the units that make sense, so units in terms of pixels —a rather persistent oversight in drafts— are kindly ignored.

use me with care

```
<math>
  <mrow>
    <mtext> use  </mtext> <mspace width="1em" />
    <mtext> me   </mtext> <mspace width="1ex" />
    <mtext> with </mtext> <mspace width="10pt"/>
    <mtext> care </mtext>
  </mrow>
</math>
```

mphantom

A phantom element hides its content but still takes its space. A phantom element can contain other elements.

who is afraid of elements

```
<math>
  <mrow>
    <mtext>    who is afraid of </mtext>    <mspace width=".5em" />
    <mphantom> phantom          </mphantom> <mspace width=".5em" />
    <mtext>    elements         </mtext>
  </mrow>
</math>
```

mpadded

As with a few other elements, I first have to see some practical usage for this, before I implement the functionality needed.

mtable, mtr, mtd, mlabeledtr

As soon as you want to represent a matrix or other more complicated composed constructs, you end up with spacing problems. This is when tables come into view. Because presentational elements have no deep knowledge about their content, tables made with presentational MATHML will in most cases look worse than those that result from content markup.

We have implemented tables on top of the normal XML (HTML) based table support in CONTEXT, also known as natural tables. Depending on the needs, support for the *mtable* element will be extended.

The *mtable* element takes a lot of attributes. When no attributes are given, we assume that a matrix is wanted, and typeset the content accordingly.

$$\begin{pmatrix} x_{1,1} & 1 & 0 \\ 0 & x_{2,2} & 1 \\ 0 & 1 & x_{3,3} \end{pmatrix}$$

```
<math>
  <mrow>
    <mo> ( </mo>
    <mtable>
      <mtr>
        <mtd> <msub> <mi> x </mi> <mn> 1,1 </mn> </msub> </mtd>
```

```
        <mtd> <mn> 1 </mn> </mtd>
        <mtd> <mn> 0 </mn> </mtd>
      </mtr>
      <mtr>
        <mtd> <mn> 0 </mn> </mtd>
        <mtd> <msub> <mi> x </mi> <mn> 2,2 </mn> </msub> </mtd>
        <mtd> <mn> 1 </mn> </mtd>
      </mtr>
      <mtr>
        <mtd> <mn> 0 </mn> </mtd>
        <mtd> <mn> 1 </mn> </mtd>
        <mtd> <msub> <mi> x </mi> <mn> 3,3 </mn> </msub> </mtd>
      </mtr>
    </mtable>
    <mo> ) </mo>
  </mrow>
</math>
```

$$\begin{vmatrix} 100 & 100 & 100 \\ 10 & 10 & 10 \\ 1 & 1 & 1 \end{vmatrix}$$

```
<math>
  <mtable columnalign="left center right" color="red blue black">
    <mtr>
      <mtd frame="on"> <mn> 100 </mn> </mtd>
      <mtd           > <mn> 100 </mn> </mtd>
      <mtd           > <mn> 100 </mn> </mtd>
    </mtr>
    <mtr>
      <mtd           > <mn> 10  </mn> </mtd>
      <mtd frame="on"> <mn> 10  </mn> </mtd>
      <mtd           > <mn> 10  </mn> </mtd>
    </mtr>
    <mtr>
      <mtd           > <mn> 1   </mn> </mtd>
      <mtd           > <mn> 1   </mn> </mtd>
      <mtd frame="on"> <mn> 1   </mn> </mtd>
    </mtr>
  </mtable>
</math>
```

Although the underlying table mechanism can provide all the support needed (and even more), not all attributes are yet implemented. We will make a useful selection.

columnalign	keyword: left center (middle) right
columnspacing	a meaningful dimension
rowspacing	a meaningful dimension
frame	keyword: none (off) solid (on)
color	a named color identifier
background	a named color identifier

We only support properly named colors as back- and foreground colors. The normal ConTEXt color mapping mechanism can be used to remap colors. This permits (read:

forces) a consistent usage of colors. If you use named backgrounds . . . the sky is the limit.

malignmark
This element is used in tables and is not yet implemented, first because I still have to unravel its exact usage, but second, because it is about the ugliest piece of MATHML markup you will encounter.

mglyph
This element is for those who want to violate the ideas of general markup by popping in his or hers own glyphs. Of course one should use entities, even if they have to be defined.

▶ + ▶ = ▶▶

```
<math>
  <mrow>
    <mi> <mglyph fontfamily="navifont" index="2" alt="right"/>    </mi>
    <mo> + </mo>
    <mi> <mglyph fontfamily="navifont" index="2" alt="right"/>    </mi>
    <mo> = </mo>
    <mi> <mglyph fontfamily="navifont" index="6" alt="veryright"/></mi>
  </mrow>
</math>
```

mstyle
This element is implemented but not yet discussed since I want more control over its misuse.

afterword
You may have noticed that we prefer content MATHML over presentational MATHML. So, unless you're already sick of any math coded in angle brackets, we invite you to read the next chapter too.

Content markup

In this chapter we will discuss the MATHML elements from the point of view of typesetting. We will not pay attention to other rendering techniques, like speech generation. Some elements take attributes and those often make more sense for other applications than for a typesetting engine like TEX, which has a strong math engine that knows how to handle math.

apply
If you are dealing with rather ordinary math, you will only need a subset of content MATHML. For this reason we will start with the most common elements. When you key in XML directly, you will encounter the *apply* element quite often, even in a relatively short formula like the following.

```
<math> <apply> <minus/> <cn> 1 </cn> </apply> </math>
```

In most cases the *apply* element is followed by a specification disguised as an empty element.

ci, cn, sep
These elements are used to specify identifiers and numbers. Both elements can be made more explicit by using attributes.

`type`	`set`	use a representation appropriate for sets
	`vector`	mark this element as vector

	function	consider this element to be a function
	fn	idem

When *set* is specified, a blackboard symbol is used when available.

$$x \in \mathbb{N}$$

```
<math>
  <apply> <in/>
    <ci> x </ci>
    <ci type="set"> N </ci>
  </apply>
</math>
```

The *function* specification makes sense when the *ci* element is used in for instance a differential equation.

type	integer	a whole number with an optional base
	logical	a boolean constant
	rational	a real number
	complex-cartesian	a complex number in $x + iy$ notation
	complex	idem
	complex-polar	a complex number in polar notation ...

You're lucky when your document uses decimal notation, otherwise you will end up with long specs if you want to be clear in what numbers are used.

$$1A2C_{16} + 0101_{16} = 1B2D_{16}$$

```
<math>
  <apply> <eq/>
    <apply> <plus/>
      <cn type="integer" base="16"> 1A2C </cn>
      <cn type="integer" base="16"> 0101 </cn>
    </apply>
    <cn type="integer" base="16"> 1B2D </cn>
  </apply>
</math>
```

Complex numbers have two components. These are separated by the *sep* element. In the following example we see that instead of using a *ci* with set specifier, the empty element *complexes* can be used. We will see some more of those later.

$$2 + 5i \in \mathbb{C}$$

```
<math>
  <apply> <in/>
    <cn type="complex"> 2 <sep/> 5 </cn>
    <complexes/>
  </apply>
</math>
```

eq, neq, gt, lt, geq, leq

Expressions, and especially those with *eq* are typical for math. Because such expressions can be quite large, there are provisions for proper alignment.

lt	$a < b$	leq	$a \leq b$
eq	$a = b$	neq	$a \neq b$
gt	$a > b$	geq	$a \geq b$

$$a \leq b \leq c$$

```
<math>
  <apply> <leq/>
    <ci> a </ci>
    <ci> b </ci>
    <ci> c </ci>
  </apply>
</math>
```

equivalent, approx, implies

Equivalence, approximations, and implications are handled like *eq* and alike and have their own symbols.

$$a + b \equiv b + a$$

```
<math>
  <apply> <equivalent/>
    <apply> <plus/> <ci> a </ci> <ci> b </ci> </apply>
    <apply> <plus/> <ci> b </ci> <ci> a </ci> </apply>
  </apply>
</math>
```

This document is typeset with PDFTEX version 3.14159, and given that TEX is written by a mathematician, it will be no surprise that:

$$3.14159 \approx \pi$$

```
<math>
  <apply> <approx/>
    <cn> 3.14159 </cn>
    <pi/>
  </apply>
</math>
```

$$x + 4 = 9 \Rightarrow x = 5$$

```
<math>
  <apply> <implies/>
    <apply> <eq/>
      <apply> <plus/>
        <ci> x </ci>
        <cn> 4 </cn>
      </apply>
      <cn> 9 </cn>
    </apply>
    <apply> <eq/>
      <ci> x </ci>
      <cn> 5 </cn>
    </apply>
  </apply>
</math>
```

minus, plus

Addition and subtraction are main building blocks of math so you will meet them often.

$$37 - x$$

```
<math>
  <apply> <minus/>
    <cn> 37 </cn>
    <ci> x </ci>
  </apply>
</math>
```

In most cases there will be more than one argument to take care of, but especially *minus* will be used with one argument too. Although `<cn> -37 </cn>` is valid, using *minus* is sometimes more clear.

$$-37$$

```
<math>
  <apply> <minus/>
    <cn> 37 </cn>
  </apply>
</math>
```

You should pay attention to combinations of *plus* and *minus*. Opposite to presentational MATHML, in content markup you don't think and code sequential.

$$-x + 37$$

```
<math>
  <apply> <plus/>
    <apply> <minus/>
      <ci> x </ci>
    </apply>
    <cn> 37 </cn>
  </apply>
</math>
```

times

Multiplication is another top ten element. Although `3p` as content of the *ci* element would have rendered the next example as well, you really should split off the number and mark it as *cn*. When this is done consistently, we can comfortably change the font of numbers independent of the font used for displaying identifiers.

$$3p$$

```
<math>
  <apply> <times/>
    <cn> 3 </cn>
    <ci> p </ci>
  </apply>
</math>
```

In a following chapter we will see how we can add multiplication signs between variables and constants.

divide

When typeset, a division is characterized by a horizontal rule. Some elements, like the differential element *diff*, generate their own division.

This example also demonstrates how to mix *plus* and *minus*.

```
[file pc-s-001.xml does not exist]
```

power

In presentational MATHML you think in super- and subscripts, but in content MATHML these elements are not available. There you need to think in terms of *power*.

$$x^2 + \sin^2 x$$

```
<math>
  <apply> <plus/>
    <apply> <power/>
      <ci> x </ci>
      <cn> 2 </cn>
    </apply>
    <apply> <power/>
      <apply> <sin/>
        <ci> x </ci>
      </apply>
      <cn> 2 </cn>
    </apply>
  </apply>
</math>
```

The *power* element is clever enough to determine where the superscript should go. In the case of the sinus function, by default it will go after the function identifier.

root, degree

If you study math related DTD's —this are the formal descriptions for SGML or XML element collections— you will notice that there are not that many elements that demand a special kind of typography: differential equations, limits, integrals and roots are the most distinctive ones.

$$\sqrt[3]{64} = 4$$

```
<math>
  <apply> <eq/>
    <apply> <root/>
      <degree> 3 </degree>
      <ci> 64 </ci>
    </apply>
    <cn> 4 </cn>
  </apply>
</math>
```

Contrary to *power*, the *root* element uses a specialized child element to denote the degree. The positive consequence of this is that there cannot be a misunderstanding about what role the child element plays, while in for instance *power* you need to know that the second child element denotes the degree.

sin, cos, tan, cot, scs, sec, . . .

All members of the family of goniometric functions are available as empty element. When needed, their argument is surrounded by braces. They all behave the same.

sin	arcsin	sinh	arcsinh
cos	arccos	cosh	arccosh
tan	arctan	tanh	arctanh
cot	arccot	coth	arccoth
csc	arccsc	csch	arccsch
sec	arcsec	sech	arcsech

These functions are normally typeset in a non italic (often roman) font shape.

By default the typesetting engine will minimize the number of braces that surrounds the argument of a function.

```
[file wh-g-001.xml does not exist]
```

You can specify π as an entity *pi* (coded as `&pi;`) or as empty element *pi*. In many cases it is up to your taste which one you use. There are many symbols that are only available as entity, so in some respect there is no real reason to treat π different.

$$\cos \pi = -1$$

```
<math>
  <apply> <eq/>
    <apply> <cos/>
      <pi/>
    </apply>
    <apply> <minus/>
      <cn> 1 </cn>
    </apply>
  </apply>
</math>
```

log, ln, exp

The *log* and *ln* are typeset similar to the previously discussed goniometric functions. The *exp* element is a special case of *power*. The constant e can be specified with *exponentiale*.

$$\ln (e + 2) \approx 1.55$$

```
<math>
  <apply> <approx/>
    <apply> <ln/>
      <apply> <plus/>
        <exponentiale/>
        <cn> 2 </cn>
      </apply>
    </apply>
    <cn> 1.55 </cn>
  </apply>
</math>
```

$$e^2 = 7.3890560989307$$

```
<math>
  <apply> <eq/>
    <apply> <exp/>
      <cn> 2 </cn>
    </apply>
    <cn> 7.3890560989307 </cn>
  </apply>
</math>
```

quotient, rem

The result of a division can be a rational number, so $\frac{5}{4}$ is equivalent to 1.25 and 1.25×4 gives 5. An integer division will give 1 with a remainder 2. Many computer languages provide a `div` and `mod` function, and since MATHML is also meant for computation, it provides similar concepts, represented by the elements *quotient* and *rem*. The representation of *quotient* is rather undefined, but the next one is among the recommended alternatives.

$$\lfloor a/b \rfloor$$

```
<math>
  <apply> <quotient/>
    <ci> a </ci>
    <ci> b </ci>
  </apply>
</math>
```

factorial

Showing the representation of a factorial is rather dull, so we will use a few more elements as well as a processing instruction to illustrate the usage of *factorial*.

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

```
<math>
  <?context-mathml-directive times symbol yes ?>
  <apply> <eq/>
    <apply> <factorial/>
      <ci> n </ci>
    </apply>
    <apply> <times/>
      <ci> n </ci>
      <apply> <minus/> <ci> n </ci> <cn> 1 </cn> </apply>
      <apply> <minus/> <ci> n </ci> <cn> 2 </cn> </apply>
      <ci> &cdots; </ci>
      <cn> 1 </cn>
    </apply>
  </apply>
</math>
```

The processing instruction is responsible for the placement of the $\times$ symbols.

min, max, gcd, lcm

These functions can handle more than two arguments. When typeset, these are separated by comma's.

$$z = \min\left\{(x+y), 2x, \frac{1}{y}\right\}$$

```
<math>
  <apply> <eq/>
    <ci> z </ci>
    <apply> <min/>
      <apply> <plus/>   <ci> x </ci> <ci> y </ci> </apply>
      <apply> <times/>  <cn> 2 </cn> <ci> x </ci> </apply>
      <apply> <divide/> <cn> 1 </cn> <ci> y </ci> </apply>
    </apply>
  </apply>
</math>
```

and, or, xor, not

Logical expressions can be defined using these elements. The operations are represented by symbols and braces are applied when needed.

$$1001_2 \wedge 0101_2 = 0001_2$$

```
<math>
  <apply> <eq/>
    <apply> <and/>
      <cn type="integer" base="2"> 1001 </cn>
      <cn type="integer" base="2"> 0101 </cn>
    </apply>
    <cn type="integer" base="2"> 0001 </cn>
  </apply>
</math>
```

set, bvar

The appearance of a *set* depends on the presence of the child element *bvar*. In its simplest form, a set is represented as a list.

$$\{1,4,8\} \neq \emptyset$$

```
<math>
  <apply> <neq/>
    <set>
      <cn> 1 </cn>
      <cn> 4 </cn>
      <cn> 8 </cn>
    </set>
    <emptyset/>
  </apply>
</math>
```

A set can be distinguished from a vector by its curly braces. The simplest case is just a comma separated list. The next example demonstrates the declarative case. Without doubt, there will be other alternatives.

$$\{x \mid 2 < x < 8\}$$

```
<math>
  <set>
    <bvar><ci> x </ci></bvar>
    <condition>
      <apply> <lt/>
        <cn> 2 </cn>
        <ci> x </ci>
        <cn> 8 </cn>
      </apply>
    </condition>
  </set>
</math>
```

list

This element is used in different contexts. When used as a top level element, a list is typeset as follows.

$$[1,1,3]$$

```
<math>
  <list>
    <cn> 1 </cn>
    <cn> 1 </cn>
    <cn> 3 </cn>
```

```
  </list>
</math>
```

When used in a context like *partialdiff*, the list specification becomes a subscript.

$$D_{1,1,3}f$$

```
<math>
  <apply> <partialdiff/>
    <list>
      <cn> 1 </cn>
      <cn> 1 </cn>
      <cn> 3 </cn>
    </list>
    <ci type="fn"> f </ci>
  </apply>
</math>
```

The function specification in this formula (which is taken from the specs) can also be specified as `<fn> <ci> f </ci> </fn>` (which is more clear).

union, intersect, . . .

There is a large number of set operators, each represented by a distinctive symbol.

union	$U \cup V$		
intersect	$U \cap V$		
in	$U \in V$	notin	$U \notin V$
subset	$U \subset V$	notsubset	$U \not\subset V$
prsubset	$U \subseteq V$	notprsubset	$U \not\subseteq V$
setdiff	$U \setminus V$		

These operators are applied as follows:

$$U \cup V$$

```
<math>
  <apply> <union/>
    <ci> U </ci>
    <ci> V </ci>
  </apply>
</math>
```

conjugate, arg, real, imaginary

The visual representation of *conjugate* is a horizontal bar with a width matching the width of the expression.

$$\overline{x + \mathrm{i}y}$$

```
<math>
  <apply> <conjugate/>
    <apply> <plus/>
      <ci> x </ci>
      <apply> <times/>
        <cn> &ImaginaryI; </cn>
        <ci> y </ci>
      </apply>
    </apply>
  </apply>
</math>
```

The *arg*, *real* and *imaginary* elements trigger the following appearance.

$$\arg(x + \mathrm{i}y)$$

$$\Re(x + \mathrm{i}y)$$

$$\Im(x + \mathrm{i}y)$$

abs, floor, ceiling

There are a couple of functions that turn numbers into positive or rounded ones. In computer languages names are used, but in math we use special boundary characters.

$$|-5| = 5$$

$$\lfloor 5.5 \rfloor = 5$$

$$\lceil 5.5 \rceil = 6$$

```
<math>
  <apply> <eq/>
    <apply> <abs/> <cn> -5 </cn> </apply>
    <cn> 5 </cn>
  </apply>
</math>
<math>
  <apply> <eq/>
    <apply> <floor/> <cn> 5.5 </cn> </apply>
    <cn> 5 </cn>
  </apply>
</math>
<math>
  <apply> <eq/>
    <apply> <ceiling/> <cn> 5.5 </cn> </apply>
    <cn> 6 </cn>
  </apply>
</math>
```

interval

An interval is visualized as: $[1, 10]$. The *interval* element is a container element and has a begin and endtag. You can specify the closure as attribute:

$$(a, b]$$

```
<math>
  <interval closure="open-closed">
    <ci> a </ci>
    <ci> b </ci>
  </interval>
</math>
```

The following closures are supported:

open	(a, b)
closed	$[a, b]$
open-closed	$(a, b]$
closed-open	$[a, b)$

inverse
This operator is applied to a function. The following example demonstrates that this is one of the few cases (if not the only one) where the first element following an *apply* begintag is an *apply* itself.

$$\sin^{-1} x$$

```
<math>
  <apply>
    <apply> <inverse/> <sin/> </apply>
    <ci> x </ci>
  </apply>
</math>
```

reln
This element is a left–over from the first MATHML specification and its usage is no longer advocated. Its current functionality matches the functionality of *apply*.

cartesianproduct, vectorproduct, scalarproduct, outerproduct
Often the context of the formula will provide information of what kind of multiplication is meant, but using different symbols to represent the kind of product certainly helps.

$$a \times b$$

```
<math>
  <apply> <cartesianproduct/>
    <ci> a </ci>
    <ci> b </ci>
  </apply>
</math>
```

We have:

cartesian	$a \times b$
vector	$a \times b$
scalar	$a \cdot b$
outer	$a \otimes b$

sum, product, limit, lowlimit, uplimit, bvar
Sums, products and limits have a distinctive look, especially when they have upper and lower limits attached. Unfortunately there is no way to specify the x_i in content MATHML. In the next chapter we will see how we can handle that.

$$\sum_{i=1}^{n} \frac{1}{x}$$

```
<math>
  <apply> <sum/>
    <bvar> <ci> i </ci> </bvar>
    <lowlimit> <cn> 1 </cn> </lowlimit>
    <uplimit> <ci> n </ci> </uplimit>
    <apply> <divide/>
      <cn> 1 </cn>
      <ci> x </ci>
    </apply>
  </apply>
</math>
```

When we omit the limits, the *bvar* is still typeset.

$$\prod_{i} \frac{1}{x}$$

```
<math>
  <apply> <product/>
    <bvar>
      <ci> i </ci>
    </bvar>
    <apply> <divide/>
      <cn> 1 </cn>
      <ci> x </ci>
    </apply>
  </apply>
</math>
```

You can specify the condition under which the function is applied.

$$\prod_{x \in \mathbb{R}} f(x)$$

```
<math>
  <apply> <product/>
    <bvar>
      <ci> x </ci>
    </bvar>
    <condition>
      <apply> <in/>
        <ci> x </ci>
        <ci type="set"> R </ci>
      </apply>
    </condition>
    <apply> <ci type="fn"> f </ci>
      <ci> x </ci>
    </apply>
  </apply>
</math>
```

$$\lim_{x \to 0} \sin x$$

```
<math>
  <apply> <limit/>
    <bvar>
      <ci> x </ci>
    </bvar>
    <lowlimit>
      <cn> 0 </cn>
    </lowlimit>
    <apply> <sin/>
      <ci> x </ci>
    </apply>
  </apply>
</math>
```

int, diff, partialdiff, bvar, degree

These elements reach a high level of abstraction. The best way to learn how to use them is to carefully study some examples.

$$\frac{\mathrm{d}\int_{p}^{q} f(x,a)\ \mathrm{d}x}{\mathrm{d}a}$$

```
<math>
  <apply> <diff/>
    <bvar> <ci> a </ci> </bvar>
    <apply> <int/>
      <lowlimit> <ci> p </ci> </lowlimit>
      <uplimit>  <ci> q </ci> </uplimit>
      <bvar> <ci> x </ci> </bvar>
      <apply>
        <fn> <ci> f </ci> </fn>
        <ci> x </ci>
        <ci> a </ci>
      </apply>
    </apply>
  </apply>
</math>
```

The *bvar* element is essential, since it is used to automatically generate some of the components that make up the visual appearance of the formula. If you look at the formal specification of these elements, you will notice that the appearance may depend on your definition. How the formula shows up, depends not only on the *bvar* element, but also on the optional *degree* element within.

$$f'$$

```
<math>
  <apply> <diff/>
    <ci> f </ci>
  </apply>
</math>
```

$$\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}$$

```
<math>
  <apply> <diff/>
    <bvar>
      <ci> x </ci>
      <degree> <cn> 2 </cn> </degree>
    </bvar>
    <apply> <fn> <ci> f </ci> </fn>
      <ci> x </ci>
    </apply>
  </apply>
</math>
```

$$\frac{\partial^4 f}{\partial x^2 \partial y \partial x}$$

```
<math>
  <apply> <partialdiff/>
    <bvar>
      <degree> <cn> 2 </cn> </degree>
      <ci> x </ci>
    </bvar>
    <bvar> <ci> y </ci> </bvar>
    <bvar> <ci> x </ci> </bvar>
    <degree> <cn> 4 </cn> </degree>
    <ci type="fn"> f </ci>
  </apply>
</math>
```

$$\frac{\partial^k f(x,y)}{\partial x^m \partial y^n}$$

```
<math>
  <apply> <partialdiff/>
    <bvar>
      <ci> x </ci> <degree> <ci> m </ci> </degree>
    </bvar>
    <bvar>
      <ci> y </ci> <degree> <ci> n </ci> </degree>
    </bvar>
    <degree> <ci> k </ci> </degree>
    <apply> <ci type="fn"> f </ci>
      <ci> x </ci>
      <ci> y </ci>
    </apply>
  </apply>
</math>
```

$$\frac{\partial^{m+n} f(x,y)}{\partial x^m \partial y^n}$$

```
<math>
  <apply> <partialdiff/>
    <bvar>
      <ci> x </ci> <degree> <ci> m </ci> </degree>
    </bvar>
    <bvar>
      <ci> y </ci> <degree> <ci> n </ci> </degree>
    </bvar>
    <apply> <ci type="fn"> f </ci>
      <ci> x </ci>
      <ci> y </ci>
    </apply>
  </apply>
</math>
```

When a degree is not specified, it is deduced from the context, but since this is not 100% watertight, you can best be complete in your specification.

These examples are taken from the MathML specification. In the example document that comes with this manual you can find a couple of more.

fn

There are a lot of predefined functions and operators. If you want to introduce a new one, the *fn* element can be used. In the following example we have turned the $\pm$ and $\mp$ symbols into (coupled) operators.

$$x \pm 1 \, x \mp 1 = x^2 - 1$$

```
<math>
  <apply> <eq/>
    <apply> <times/>
      <apply> <fn> <ci> &plusminus; </ci> </fn>
        <ci> x </ci>
        <cn> 1 </cn>
      </apply>
      <apply> <fn> <ci> &minusplus; </ci> </fn>
        <ci> x </ci>
        <cn> 1 </cn>
      </apply>
    </apply>
    <apply> <minus/>
      <apply> <power/>
        <ci> x </ci>
        <cn> 2 </cn>
      </apply>
      <cn> 1 </cn>
    </apply>
  </apply>
</math>
```

The typeset result depends on the presence of a handler, which in this case happens to be true.

matrix, matrixrow

Matrices are one of the building blocks of linear algebra and therefore both presentational and content MATHML have dedicated elements for defining them.

$$\begin{pmatrix} 23 & 87 & c \\ 41 & b & 33 \\ a & 65 & 16 \end{pmatrix}$$

```
<math>
  <matrix>
    <matrixrow> <cn> 23 </cn> <cn> 87 </cn> <ci>  c </ci> </matrixrow>
    <matrixrow> <cn> 41 </cn> <ci>  b </ci> <cn> 33 </cn> </matrixrow>
    <matrixrow> <ci>  a </ci> <cn> 65 </cn> <cn> 16 </cn> </matrixrow>
  </matrix>
</math>
```

vector

We make a difference between a vector specification and a vector variable. A specification is presented as a list:

$$(x,y)$$

```
<math>
  <vector>
    <ci> x </ci>
    <ci> y </ci>
```

```
  </vector>
</math>
```

When the *vector* element has one child element, we use a right arrow to identify the variable as vector.

$$\overrightarrow{A} \times \overrightarrow{B}$$

```
<math>
  <apply> <vectorproduct/>
    <vector> <ci> A </ci> </apply>
    <vector> <ci> B </ci> </apply>
  </vector>
</math>
```

grad, curl, ident, divergence

These elements expand into named functions, but we can imagine that in the future a more appropriate visualization will be provided as an option.

$$\operatorname{grad} A \neq \operatorname{curl} B \neq \operatorname{id} C \neq \operatorname{div} D$$

```
<math>
  <apply> <neq/>
    <apply> <grad/>       <ci> A </ci> </apply>
    <apply> <curl/>       <ci> B </ci> </apply>
    <apply> <ident/>      <ci> C </ci> </apply>
    <apply> <divergence/> <ci> D </ci> </apply>
  </apply>
</math>
```

lambda, bvar

The lambda specification of a function needs a *bvar* element. The visualization can be influenced with processing instructions as described in a later chapter.

$$x \mapsto \sin\left(x - \frac{x}{2}\right)$$

```
<math>
  <lambda>
    <bvar> <ci> x </ci> </bvar>
    <apply> <sin/>
      <apply> <minus/>
        <ci> x </ci>
        <apply> <divide/>
          <ci> x </ci>
          <cn> 2 </cn>
        </apply>
      </apply>
    </apply>
  </lambda>
</math>
```

piecewise, piece, otherwise

There are not so many elements that deal with combinations of formulas or conditions. The *piecewise* is the only real selector available. The following example defines how the state of n depends on the state of x.

$$n = \begin{cases} -1 & x < 0 \\ 1 & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

```
<math>
  <apply> <eq/>
    <ci> n </ci>
    <piecewise>
      <piece>
        <apply> <minus/>
          <cn> 1 </cn>
        </apply>
        <apply> <lt/>
          <ci> x </ci>
          <cn> 0 </cn>
        </apply>
      </piece>
      <piece>
        <cn> 1 </cn>
        <apply> <gt/>
          <ci> x </ci>
          <cn> 0 </cn>
        </apply>
      </piece>
      <otherwise>
        <cn> 0 </cn>
      </otherwise>
    </piecewise>
  </apply>
</math>
```

We could have use a third *piece* instead of (optional) *otherwise*.

forall, exists, condition

Conditions are often used in combination with elements like *forall*. There are several ways to convert and combine them in formulas and environments, so you may expect more alternatives in the future.

$$\forall_x \ x < 9 \,|\, x < 10$$

```
<math>
  <apply> <forall/>
    <bvar> <ci> x </ci> </bvar>
    <condition>
      <apply> <lt/>
        <ci> x </ci>
        <cn> 9 </cn>
      </apply>
    </condition>
    <apply> <lt/>
      <ci> x </ci>
      <cn> 10 </cn>
    </apply>
  </apply>
</math>
```

The next example is taken from the specifications with a few small changes.

$$\forall_x \ x \in \mathbb{N} \mid \exists_{p,q} \ p \in \mathbb{P} \wedge q \in \mathbb{P} \wedge p + q = 2x$$

```
<math>
  <apply> <forall/>
    <bvar> <ci> x </ci> </bvar>
    <condition>
      <apply> <in/>
        <ci> x </ci>
        <ci type="set"> N </ci>
      </apply>
    </condition>
    <apply> <exists/>
      <bvar> <ci> p </ci> </bvar>
      <bvar> <ci> q </ci> </bvar>
      <condition>
        <apply> <and/>
          <apply> <in/>
            <ci> p </ci>
            <ci type="set"> P </ci>
          </apply>
          <apply> <in/>
            <ci> q </ci>
            <ci type="set"> P </ci>
          </apply>
          <apply> <eq/>
            <apply> <plus/> <ci> p </ci> <ci> q </ci> </apply>
            <apply> <times/> <cn> 2 </cn> <ci> x </ci> </apply>
          </apply>
        </apply>
      </condition>
    </apply>
  </apply>
</math>
```

factorof, tendsto

The *factorof* element is applied to its two child elements and contrary to most functions, the symbol is placed between the elements instead of in front.

$$a \mid b$$

```
<math>
  <apply> <factorof/>
    <ci> a </ci>
    <ci> b </ci>
  </apply>
</math>
```

The same is true for the *tendsto* element.

$$a \to b$$

```
<math>
  <apply> <tendsto/>
    <ci> a </ci>
    <ci> b </ci>
```

```
  </apply>
</math>
```

compose

This is a nasty element since it has to take care of braces in special ways and therefore has to analyse its child elements.

$$f \circ g \circ h$$

```
<math>
  <apply> <compose/>
    <ci type="fn"> f </ci>
    <ci type="fn"> g </ci>
    <ci type="fn"> h </ci>
  </apply>
</math>
```

$$(f \circ g)\, x$$

```
<math>
  <apply>
    <apply> <compose/>
      <fn> <ci> f </ci> </fn>
      <fn> <ci> g </ci> </fn>
    </apply>
    <ci> x </ci>
  </apply>
</math>
```

laplacian

A laplacian function is typeset using a ∇ (nabla) symbol.

$$\nabla^2 x$$

```
<math>
  <apply> <laplacian/>
    <ci> x </ci>
  </apply>
</math>
```

mean, sdev, variance, median, mode

When statistics shows up in math text books, the *sum* element is likely to show up, probably in combination with the for statistics meaningful symbolic representation of variables. The mean value of a series of observations is defined as:

$$\overline{x} = \frac{\sum x}{n}$$

```
<math>
  <apply> <eq/>
    <apply> <mean/>
      <ci> x </ci>
    </apply>
    <apply> <divide/>
      <apply> <sum/>
        <ci> x </ci>
      </apply>
      <ci> n </ci>
```

```
    </apply>
  </apply>
</math>
```

or more beautiful:

$$\overline{x} = \frac{1}{n} \sum x$$

```
<math>
  <apply> <eq/>
    <apply> <mean/>
      <ci> x </ci>
    </apply>
    <apply> <times/>
      <apply> <divide/>
        <cn> 1 </cn>
        <ci> n </ci>
      </apply>
      <apply> <sum/>
        <ci> x </ci>
      </apply>
    </apply>
  </apply>
</math>
```

Of course this definition is not that perfect, but we will present a better alternative in the chapter on combined markup. The definition of the standard deviation is more complicated:

$$\sigma(x) \approx \sqrt{\frac{\sum (x - \overline{x})^2}{n - 1}}$$

```
<math>
  <apply> <approx/>
    <apply> <sdev/>
      <ci> x </ci>
    </apply>
    <apply> <root/>
      <apply> <divide/>
        <apply> <sum/>
          <apply> <power/>
            <apply> <minus/>
              <ci> x </ci>
              <apply> <mean/>
                <ci> x </ci>
              </apply>
            </apply>
            <cn> 2 </cn>
          </apply>
        </apply>
        <apply> <minus/>
          <ci> n </ci>
          <cn> 1 </cn>
        </apply>
      </apply>
```

```
    </apply>
  </apply>
</math>
```

The next example demonstrates the usage of the *variance* in its own definition.

$$\sigma(x)^2 = \overline{(x-\overline{x})^2} \approx \frac{1}{n-1}\sum (x-\overline{x})^2$$

```
<math>
  <apply> <eq/>
    <apply> <variance/>
      <ci> x </ci>
    </apply>
    <apply> <approx/>
      <apply> <mean/>
        <apply> <power/>
          <apply> <minus/>
            <ci> x </ci>
            <apply> <mean/>
              <ci> x </ci>
            </apply>
          </apply>
          <cn> 2 </cn>
        </apply>
      </apply>
      <apply> <times/>
        <apply> <divide/>
          <cn> 1 </cn>
          <apply> <minus/>
            <ci> n </ci>
            <cn> 1 </cn>
          </apply>
        </apply>
        <apply> <sum/>
          <apply> <power/>
            <apply> <minus/>
              <ci> x </ci>
              <apply> <mean/>
                <ci> x </ci>
              </apply>
            </apply>
            <cn> 2 </cn>
          </apply>
        </apply>
      </apply>
    </apply>
  </apply>
</math>
```

The *median* and *mode* of a series of observations have no special symbols and are presented as is.

moment, momentabout, degree

Because MATHML is used for a wide range of applications, there can be information in a definition that does not end up in print but is only used in some cases. This is illustrated in the next example.

$$\left\langle X^3 \right\rangle$$

```
<math>
  <apply> <moment/>
    <degree>
      <cn> 3 </cn>
    </degree>
    <momentabout>
      <ci> p </ci>
    </momentabout>
    <ci> X </ci>
  </apply>
</math>
```

determinant, transpose, selector

This threesome is used to manipulate matrices, either or not in a symbolic way. A simple determinant or transpose looks like:

$$|\boldsymbol{A}|$$

```
<math>
  <apply> <determinant/>
    <ci type="matrix"> A </ci>
  </apply>
</math>
```

$$\boldsymbol{A}^{\mathrm{T}}$$

```
<math>
  <apply> <transpose/>
    <ci type="matrix"> A </ci>
  </apply>
</math>
```

When the *determinant* element is applied to full blown matrix, the braces are omitted and replaced by the vertical bars.

```
[file wh-m-002.xml does not exist]
```

The *selector* element honors the braces.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}_1$$

```
<math>
  <apply> <selector/>
    <matrix>
      <matrixrow> <cn> 1 </cn> <cn> 2 </cn> </matrixrow>
      <matrixrow> <cn> 3 </cn> <cn> 4 </cn> </matrixrow>
    </matrix>
    <cn> 1 </cn>
  </apply>
</math>
```

card
A cardinality is visualized using vertical bars. [But what exactly is meant with cardinality?]

$$|A| = 5$$

```
<math>
  <apply> <eq/>
    <apply> <card/>
      <ci> A </ci>
    </apply>
    <ci> 5 </ci>
  </apply>
</math>
```

domain, codomain, image
The next couple of examples are taken from the MATHML specification and demonstrate the usage of the not that spectacular domain related elements.

$$\text{domain}(f) = \mathbb{R}$$

```
<math>
  <apply> <eq/>
    <apply> <domain/>
      <fn> <ci> f </ci> </fn>
    </apply>
    <reals/>
  </apply>
</math>
```

These are typically situations where the *fn* element may show up.

$$\text{codomain}(f) = \mathbb{Q}$$

```
<math>
  <apply> <eq/>
    <apply> <codomain/>
      <fn> <ci> f </ci> </fn>
    </apply>
    <rationals/>
  </apply>
</math>
```

This example from the MATHML specification demonstrates a typical usage of the *image* element. As with the previous two, it is applied to a function, in this case the predefined *sin*.

$$\text{image}(\sin) = [-1, 1]$$

```
<math>
  <apply> <eq/>
    <apply> <image/>
      <sin/>
    </apply>
    <interval>
      <cn> -1 </cn>
      <cn>  1 </cn>
    </interval>
```

```
  </apply>
</math>
```

domainofapplication
This is another seldom used element. Actually, this element is a further specification of the outer level applied function.

$$\int_C f$$

```
<math>
  <apply> <int/>
    <domainofapplication>
      <ci> C </ci>
    </domainofapplication>
    <ci> f </ci>
  </apply>
</math>
```

semantics, annotation, annotation-xml
We will never know what Albert Einstein would have thought about MATHML. But we do know for sure that coding one of his famous findings in XML takes much more tokens that it takes in TEX.

Within a *semantics* element there can be many *annotation* elements. When using CONTEXT, the elements that can be identified as being encoded in TEX will be treated as such. Currently, the related *annotation-xml* element is ignored.

$$e = mc^2$$

```
<math>
  <semantics>
    <apply> <eq/>
      <ci> e </ci>
      <apply> <times/>
        <ci> m </ci>
        <apply> <power/>
          <ci> c </ci>
          <cn> 2 </cn>
        </apply>
      </apply>
    </apply>
    <annotation encoding="TeX">
      e = m c^2
    </annotation>
  </semantics>
</math>
```

integers, reals, ...
Sets are characterized with special (often blackboard) symbols. These symbols are not always available.

integers	$\mathbb{Z}$
reals	$\mathbb{R}$
rationals	$\mathbb{Q}$
naturalnumbers	$\mathbb{N}$

complexes	$\mathbb{C}$
primes	$\mathbb{P}$

pi, imaginaryi, exponentiale

Being a greek character, π is a distinctive character. In most math documents the imaginary i and exponential e are typeset as any math identifier.

pi	π
imaginaryi	i
exponentiale	e

eulergamma, infinity, emptyset

There are a couple of more special tokens. As with the other ones, they can be changed by reassigning the corresponding entities.

eulergamma	γ
infinity	∞
emptyset	$\emptyset$

notanumber

Because MATHML is used for more purposes than typesetting, there are a couple of elements that do not make much sense in print. One of these is *notanumber*, which is issued by programs as error code or string.

$$\frac{x}{0} = \text{NaN}$$

```
<math>
  <apply> <eq/>
    <apply> <divide/>
      <ci> x </ci>
      <cn> 0 </cn>
    </apply>
    <notanumber/>
  </apply>
</math>
```

true, false

When assigning to a boolean variable, or in boolean expressions one can use 0 or 1 to identify the states, but if you want to be more verbose, you can use these elements.

$$1_2 \equiv \text{true}$$

```
<math>
  <apply> <equivalent/>
    <cn type="integer" base="2"> 1 </cn>
    <true/>
  </apply>
</math>
```

declare

Reusing definitions would be a nice feature, but for the moment the formal specification of this element currently does not give us the freedom to use it the way we want.

$$\text{declare } A \text{ as } (a,b,c)$$

```
<math>
  <declare>
    <ci> A </ci>
    <vector>
      <ci> a </ci>
      <ci> b </ci>
      <ci> c </ci>
    </vector>
  </declare>
</math>
```

csymbol
This element will be implemented as soon as I have an application for it.

Mixed markup

The advantage of presentational markup is that you can build complicated formulas using super- and subscripts and other elements. The drawback is that the look and feel is rather fixed and cannot easily be adapted to the purpose that the document serves. Take for instance the difference between

$$\log_2 x$$

and

$$^2\log x$$

Both formulas were defined in content MATHML, so no explicit super- and subscripts were used. In the next chapter we will see how to achieve such different appearances.

There are situations where content MATHML is not rich enough to achieve the desired output. This omission in content MATHML forces us to fall back on presentational markup.

$$P_1 = P_2 = 1.01 \approx 1$$

Here we used presentational elements inside a content *ci* element. We could have omitted the outer *ci* element, but since the content MATHML parser may base its decisions on the content elements it finds, it is best to keep the outer element there.

```
<math>
  <apply> <eq/>
    <ci> <msub> <mi> P </mi> <mi> 1 </mi> </msub> </ci>
    <ci> <msub> <mi> P </mi> <mi> 2 </mi> </msub> </ci>
    <apply> <approx/>
      <cn> 1.01 </cn>
      <cn> 1 </cn>
    </apply>
  </apply>
</math>
```

The lack of an index element can be quite prominent. For instance, when in an expose about rendering we want to explore the mapping from coordinates in user space to those in device space, we use the following formula.

$$\left(D_x,D_y,1\right) = \left(U_x,U_y,1\right) \begin{pmatrix} s_x & r_x & 0 \\ r_y & s_y & 0 \\ t_x & t_y & 1 \end{pmatrix}$$

```
<math>
  <apply> <eq/>
    <vector>
      <ci> <msub> <mi> D </mi> <mi> x </mi> </msub> </ci>
      <ci> <msub> <mi> D </mi> <mi> y </mi> </msub> </ci>
      <cn> 1 </cn>
    </vector>
    <apply> <times/>
      <vector>
        <ci> <msub> <mi> U </mi> <mi> x </mi> </msub> </ci>
        <ci> <msub> <mi> U </mi> <mi> y </mi> </msub> </ci>
        <cn> 1 </cn>
      </vector>
      <matrix>
        <matrixrow>
          <ci> <msub> <mi> s </mi> <mi> x </mi> </msub> </ci>
          <ci> <msub> <mi> r </mi> <mi> x </mi> </msub> </ci>
          <cn> 0 </cn>
        </matrixrow>
        <matrixrow>
          <ci> <msub> <mi> r </mi> <mi> y </mi> </msub> </ci>
          <ci> <msub> <mi> s </mi> <mi> y </mi> </msub> </ci>
          <cn> 0 </cn>
        </matrixrow>
        <matrixrow>
          <ci> <msub> <mi> t </mi> <mi> x </mi> </msub> </ci>
          <ci> <msub> <mi> t </mi> <mi> y </mi> </msub> </ci>
          <cn> 1 </cn>
        </matrixrow>
      </matrix>
    </apply>
  </apply>
</math>
```

Again, the *msub* element provides a way out, as in the next examples, which are adapted versions of formulas we used when demonstrating the statistics related elements.

$$\overline{x} = \frac{1}{n} \sum_i x$$

```
<math>
  <apply> <eq/>
    <apply> <mean/>
      <ci> x </ci>
    </apply>
    <apply> <times/>
      <apply> <divide/>
        <cn> 1 </cn>
        <ci> n </ci>
      </apply>
      <apply> <sum/>
        <bvar> <ci> i </ci> </bvar>
        <ci> x </ci>
      </apply>
```

```
    </apply>
  </apply>
</math>
```

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x$$

```
<math>
  <apply> <eq/>
    <apply> <mean/>
      <ci> x </ci>
    </apply>
    <apply> <times/>
      <apply> <divide/>
        <cn> 1 </cn>
        <ci> n </ci>
      </apply>
      <apply> <sum/>
        <bvar> <ci> i </ci> </bvar>
        <lowlimit> <cn> 1 </cn> </lowlimit>
        <uplimit> <cn> n </cn> </uplimit>
        <ci> x </ci>
      </apply>
    </apply>
  </apply>
</math>
```

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

```
<math>
  <apply> <eq/>
    <apply> <mean/>
      <ci> x </ci>
    </apply>
    <apply> <times/>
      <apply> <divide/>
        <cn> 1 </cn>
        <ci> n </ci>
      </apply>
      <apply> <sum/>
        <bvar> <ci> i </ci> </bvar>
        <lowlimit> <cn> 1 </cn> </lowlimit>
        <uplimit> <cn> n </cn> </uplimit>
        <ci> <msub> <mi> x </mi> <mi> i </mi> </msub> </ci>
      </apply>
    </apply>
  </apply>
</math>
```

Directives

Some elements can be tuned by changing their attributes. Especially when formulas are defined by a team of people or when they are taken from a repository, there is a good chance that inconsistencies will show up.

In ConTEXt, you can influence the appearance by setting the typesetting parameters of (classes of) elements. You can do this either by adding processing instructions, or by using the ConTEXt command `\setupMMLappearance`. Although the first method is more in the spirit of XML, the second method is more efficient and consistent. As a processing instruction, a directive looks like:

```
<?context-mathml-directive element key value ?>
```

This is equivalent to the ConTEXt command:

```
\setupMMLappearance [element] [key=value]
```

Some settings concern a group of elements, in which case a group classification (like `sign`) is used.

scripts

By default, nested super- and subscripts are kind of isolated from each other. If you want a combined script, there is the *msubsup*. You can however force combinations with a directive.

$$x_1{}^2$$
$$x_1^2$$

```
<math>
  <msup>
    <msub> <mi> x </mi> <mn> 1 </mn> </msub>
    <mn> 2 </mn>
  </msup>
</math>
<?context-mathml-directive scripts alternative b ?>
<math>
  <msup>
    <msub> <mi> x </mi> <mn> 1 </mn> </msub>
    <mn> 2 </mn>
  </msup>
</math>
```

sign

The core element of MathML is *apply*. Even simple formulas will often have more than one (nested) *apply*. The most robust way to handle nested formulas is to use braces around each sub formula. No matter how robust this is, when presented in print we want to use as less braces as possible.

$$7 + 5 - 3$$

This expression shows addition as well as subtraction.

```
<math>
  <apply> <plus/>
    <cn> 7 </cn>
    <cn> 5 </cn>
    <apply> <minus/>
      <cn> 3 </cn>
```

```
    </apply>
  </apply>
</math>
```

In principle subtraction is adding negated numbers, so it would have been natural to have just an addition (*plus*) and negation operator. However, MATHML provides both a *plus* and *minus* operator, where the latter can be used as a negation. So in fact we have:

$$7 + 5 + (-3)$$

Now imagine that a teacher wants to stress this negation in the way presented here, using parentheses. Since all the examples shown here are typeset directly from the MATHML source, you may expect a solution, so here it is:

```
<math>
  <?context-mathml-directive sign reduction no ?>
  <apply> <plus/>
    <cn> 7 </cn>
    <cn> 5 </cn>
    <apply> <minus/>
      <cn> 3 </cn>
    </apply>
  </apply>
</math>
```

By default signs are reduced, but one can disable that at the document and/or formula level using a processing instruction at the top of the formula. There are of course circumstances where the parentheses cannot be left out.

$$a + b + c + d$$

```
<math>
  <apply> <plus/>
    <ci> a </ci>
    <apply> <plus/> <ci> b </ci> <ci> c </ci> </apply>
    <ci> d </ci>
  </apply>
</math>
```

$$a - (b - c) - d$$

```
<math>
  <apply> <minus/>
    <ci> a </ci>
    <apply> <minus/> <ci> b </ci> <ci> c </ci> </apply>
    <ci> d </ci>
  </apply>
</math>
```

$$a + b - c + d$$

```
<math>
  <apply> <plus/>
    <ci> a </ci>
    <apply> <minus/> <ci> b </ci> <ci> c </ci> </apply>
    <ci> d </ci>
  </apply>
</math>
```

$$a - (b + c) - d$$

```
<math>
  <apply> <minus/>
    <ci> a </ci>
    <apply> <plus/> <ci> b </ci> <ci> c </ci> </apply>
    <ci> d </ci>
  </apply>
</math>
```

Another place where parentheses are not needed is the following:

```
<math>
  <apply> <minus/>
    <apply> <exp/>
      <cn> 3 </cn>
    </apply>
  </apply>
</math>
```

This means that the interpreter of this kind of MATHML has to analyze child elements in order to choose the right way to typeset the formula. The output will look like:

$$-\mathrm{e}^3$$

By default, as less braces as possible are used. As demonstrated, a special case is when *plus* and *minus* have one sub element to deal with. If you really want many braces there, you can turn off sign reduction.

`sign`	`reduction`	`yes`	use as less braces as possible
		`no`	always use braces

We will demonstrate these alternatives with an example.

$$a + \sin b + c^5 + \sin^2 d + e$$

We need quite some code to encode this formula.

```
<math>
  <apply> <plus/>
    <ci> a </ci>
    <apply> <sin/>
      <ci> b </ci>
    </apply>
    <apply> <power/>
      <ci> c </ci>
      <cn> 5 </cn>
    </apply>
    <apply> <power/>
      <apply> <sin/>
        <ci> d </ci>
      </apply>
      <cn> 2 </cn>
    </apply>
    <ci> e </ci>
  </apply>
</math>
```

With power reduction turned off, we get:

$$a + \sin b + c^5 + (\sin d)^2 + e$$

As directive we used:

```
<?context-mathml-directive power reduction no ?>
```

The following example illustrates that we should be careful in coding such formulas; here the *power* is applied to the argument of *sin*.

$$a + \sin b + c^5 + \sin\left(d^2\right) + e$$

```
<math>
  <apply> <plus/>
    <ci> a </ci>
    <apply> <sin/>
      <ci> b </ci>
    </apply>
    <apply> <power/>
      <ci> c </ci>
      <cn> 5 </cn>
    </apply>
    <apply> <sin/>
      <apply> <power/>
        <ci> d </ci>
        <cn> 2 </cn>
      </apply>
    </apply>
    <ci> e </ci>
  </apply>
</math>
```

divide

Divisions can be very space consuming but there is a way out: using a forward slash symbol. You can set the level at which this will take place. By default, fractions are typeset in the traditional way.

$$\frac{1}{1+\frac{1}{x}}$$

```
<math>
  <apply> <divide/>
    <cn> 1 </cn>
    <apply> <plus/>
      <cn> 1 </cn>
      <apply> <divide/>
        <cn> 1 </cn>
        <ci> x </ci>
      </apply>
    </apply>
  </apply>
</math>
```

$$\frac{1}{1+\frac{1}{1+\frac{1}{x}}}$$

```
<math>
  <apply> <divide/>
```

```
    <cn> 1 </cn>
    <apply> <plus/>
      <cn> 1 </cn>
      <apply> <divide/>
        <cn> 1 </cn>
        <apply> <plus/>
          <cn> 1 </cn>
          <apply> <divide/>
            <cn> 1 </cn>
            <ci> x </ci>
          </apply>
        </apply>
      </apply>
    </apply>
  </apply>
</math>
```

$$\frac{1}{1+1/x}$$

$$\frac{1}{1+1/\left(1+1/x\right)}$$

```
<?context-mathml-directive divide level 1?>
```

$$\frac{1}{1+\frac{1}{x}}$$

$$\frac{1}{1+\frac{1}{1+1/x}}$$

```
<?context-mathml-directive divide level 2?>
```

relation

You should keep in mind that (at least level 2) content MATHML is not that rich in terms of presenting your ideas in a visually attractive way. On the other hand, because the content is highly structured, some intelligence can be applied when typesetting them. By default, a relation is not vertically aligned but typeset horizontally.

If an application just needs raw formulas, definitions like the following are all right.

```
<math>
  <apply> <eq/>
    <apply> <plus/>
      <ci> a </ci>
      <ci> b </ci>
      <ci> c </ci>
    </apply>
    <apply> <plus/>
      <ci> d </ci>
      <ci> e </ci>
    </apply>
    <apply> <plus/>
      <ci> f </ci>
      <ci> g </ci>
      <ci> h </ci>
      <ci> i </ci>
    </apply>
```

```
    <cn> 123 </cn>
  </apply>
</math>
```

The typeset result will bring no surprises:

$$a + b + c = d + e = f + g + h + i = 123$$

But, do we want to show a formula that way? And what happens with much longer formulas? You can influence the appearance with processing instructions.

`relation`	`align`	`no`	don't align relations
		`left`	align all relations left
		`right`	align all relations right
		`first`	place the leftmost relation left
		`last`	place the rightmost relation right

The next couple of formulas demonstrate in what way the previously defined formula is influenced by the processing instructions.

$$\begin{aligned} a + b + c &= \\ d + e &= \\ f + g + h + i &= \\ 123 & \end{aligned}$$

```
<?context-mathml-directive relation align left ?>
```

$$\begin{aligned} a + b + c & \\ = d + e & \\ = f + g + h + i & \\ = 123 & \end{aligned}$$

```
<?context-mathml-directive relation align right ?>
```

$$\begin{aligned} a + b + c &= d + e \\ &= f + g + h + i \\ &= 123 \end{aligned}$$

```
<?context-mathml-directive relation align first ?>
```

$$\begin{aligned} a + b + c &= \\ d + e &= \\ f + g + h + i &= 123 \end{aligned}$$

```
<?context-mathml-directive relation align last ?>
```

base

When in a document several number systems are used, it can make sense to mention the base of the number. There are several ways to identify the base.

`base`	`symbol`	`numbers`	a (decimal) number
		`characters`	one character
		`text`	a mnemonic
		`no`	no symbol

By default, when specified, a base is identified as number.

```
<math>
  <cn type="integer" base="8"> 1427 </cn>
</math>
```

$$1427_{8}$$

```
<?context-mathml-directive base symbol numbers ?>
```

$$1427_{\mathrm{O}}$$

```
<?context-mathml-directive base symbol characters ?>
```

$$1427_{\mathrm{OCT}}$$

```
<?context-mathml-directive base symbol text ?>
```

function

There is a whole bunch of functions available as empty element, like *sin* and *log*. When a function is applied to a function, braces make not much sense and placement is therefore disabled.

function	reduction	yes	chain functions without braces
		no	put braces around nested functions

```
<math>
  <apply> <sin/> <ci> x </ci> </apply>
</math>
```

$$\sin x$$

```
<?context-mathml-directive function reduction yes?>
```

$$\sin\left(x\right)$$

```
<?context-mathml-directive function reduction no?>
```

limits

When limits are placed on top of the limitation symbol, this generally looks better than when they are placed alongside. You can also influence limit placement per element. This feature is available for *int*, *sum*, *product* and *limit*.

limit	location	top	place limits on top of the symbols
		right	attached limits as super/subscripts

```
<math>
  <apply> <int/>
    <bvar> <ci> x </ci> </bvar>
    <lowlimit> <cn> 0 </cn> </lowlimit>
    <uplimit> <cn> 1 </cn> </uplimit>
  </apply>
</math>
```

$$\int_{0}^{1} \mathrm{d}x$$

```
<?context-mathml-directive int location top?>
```

$$\int_{0}^{1} \mathrm{d}x$$

```
<?context-mathml-directive int location right?>
```

declare
Currently declarations are not supposed to end up in print. By default we typeset a message, but you can as well completely hide declarations.

`declare`	`state`	`start`	show declarations
		`stop`	ignore (hide) declarations

lambda
There is more than one way to visualize a lambda function. As with some other settings, changing the appearance can best take place at the document level.

`lambda`	`alternative`	`b`	show lambda as arrow
		`a`	show lambda as set

```
<math>
  <lambda>
    <bvar> <ci> x </ci> </bvar>
    <apply> <log/>
      <ci> x </ci>
    </apply>
  </lambda>
</math>
```

$$\lambda(x, \log x)$$

```
<?context-mathml-directive lambda alternative a?>
```

$$x \mapsto \log x$$

```
<?context-mathml-directive lambda alternative b?>
```

power
Taking the power of a function looks clumsy when braces are put around the function. Therefore, by default, the power is applied to the function symbol instead of the whole function.

`power`	`reduction`	`yes`	attach symbol to function symbol
		`no`	attach symbol to function argument

```
<math>
  <apply> <power/>
    <apply> <ln/>
      <ci> x </ci>
    </apply>
    <cn> 3 </cn>
  </apply>
</math>
```

$$\ln^3 x$$

```
<?context-mathml-directive power reduction yes?>
```

$$(\ln x)^3$$

```
<?context-mathml-directive power reduction no?>
```

diff
Covering all kind of differential formulas is not trivial. Currently we support two locations for the operand (function). By default the operand is placed above the division line.

diff	location	top	put the operand in the fraction
		right	put the operand after the fraction

```
<math>
  <apply> <diff/>
    <bvar>
      <ci> x </ci>
      <degree> <cn> 2 </cn> </degree>
    </bvar>
    <apply> <fn> <ci> f </ci> </fn>
      <apply> <plus/>
        <apply> <times/>
          <cn> 2 </cn>
          <ci> x </ci>
        </apply>
        <cn> 1 </cn>
      </apply>
    </apply>
  </apply>
</math>
```

$$\frac{\mathrm{d}^2 f(2x+1)}{\mathrm{d}x^2}$$

```
<?context-mathml-directive diff location top?>
```

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}\,(f(2x+1))$$

```
<?context-mathml-directive diff location right?>
```

vector

Depending on the complication of a vector or on the available space, you may wish to typeset a vector horizontally or vertically. By default a vector is typeset horizontally.

vector	direction	horizontal	put vector elements alongside
		vertical	stack vector elements

```
<math>
  <apply> <eq/>
    <vector>
      <ci> x </ci>
      <ci> y </ci>
      <ci> z </ci>
    </vector>
    <vector>
      <cn> 1 </cn>
      <cn> 0 </cn>
      <cn> 1 </cn>
    </vector>
  </apply>
</math>
```

$$(x,y,z) = (1,0,1)$$

```
<?context-mathml-directive vector direction horizontal?>
```

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

```
<?context-mathml-directive vector direction vertical?>
```

times

Depending on the audience, a multiplication sign is implicit (absent) or represented by a regular times symbol or a dot.

`times`	`symbol`	`no`	don't add a symbol
		`yes`	separate operands by a times ($\times$)
		`dot`	separate operands by a dot ($\cdot$)

```
<math>
  <apply> <plus/>
    <ci> x </ci>
    <apply> <times/>
      <cn> 2 </cn>
      <ci> x </ci>
    </apply>
  </apply>
</math>
```

$$x + 2x$$

```
<?context-mathml-directive times symbol no?>
```

$$x + 2 \times x$$

```
<?context-mathml-directive times symbol yes?>
```

$$x + 2 \cdot x$$

```
<?context-mathml-directive times symbol dot?>
```

log

The location of a logbase depends on tradition and/or preference, which is why we offer a few alternatives: as pre superscript (in the right top corner before the symbol) or as post subscript (in the lower left corner after the symbol).

`log`	`location`	`right`	place logbase at the right top
		`left`	place logbase at the lower left

```
<math>
  <apply> <log/>
    <logbase>
      <ci> 3 </ci>
    </logbase>
    <apply> <plus/>
      <ci> x </ci>
      <cn> 1 </cn>
    </apply>
  </apply>
</math>
```

$$\log_3 (x + 1)$$

```
<?context-mathml-directive log location right?>
```

$$^{3}\log(x+1)$$

```
<?context-mathml-directive log location left?>
```

polar

The location of a logbase depends on tradition and/or preference, which is why we offer a few alternatives: as pre superscript (in the right top corner before the symbol) or as post subscript (in the lower left corner after the symbol).

polar	alternative	a	explicit polar notation
		b	exponential power notation
		c	exponential function notation

```
<math>
  <cn type="polar"> 2 <sep/> <pi/> </cn>
</math>
```

$$\mathrm{Polar}(2,\pi)$$

```
<?context-mathml-directive polar alternative a?>
```

$$2e^{\pi i}$$

```
<?context-mathml-directive polar alternative b?>
```

$$2\exp(\pi i)$$

```
<?context-mathml-directive polar alternative c?>
```

e-notation

Depending on the context, you may want to typeset the number `1.23e4` not as this sequence, but using a multiplier construct. As with the *times*, we support both multiplication symbols.

enotation	symbol	no	no interpretation
		yes	split exponent, using $\times$
		dot	split exponent, using $\cdot$

```
<math>
  <cn type="e-notation">10<sep/>23</cn>
</math>
```

$$10e23$$

```
<?context-mathml-directive enotation symbol no?>
```

$$10 \times 10^{23}$$

```
<?context-mathml-directive enotation symbol yes?>
```

$$10 \cdot 10^{23}$$

```
<?context-mathml-directive enotation symbol dot?>
```

Typesetting modes

Math can be typeset in line or in display. In order not to widen up the text of a paragraph too much, inline math is typeset more cramped. Since MATHML does provide just a general purpose *math* element we have to provide the information needed using other elements. Consider the following text.

To what extent is math supposed to reflect the truth and nothing but the truth? Consider the simple expression 10 = 3 + 7. Many readers will consider this the truth, but then, can we assume that the decimal notation is used?

$$10 = 3 + x$$

In many elementary math books, you can find expressions like the previous. Because in our daily life we use the decimal numbering system, we can safely assume that $x = 7$. But, without explicitly mentioning this boundary condition, more solutions are correct.

$$10 = 3 + 5 \tag{1a}$$

In formula 1a we see an at first sight wrong formula. But, if we tell you that octal numbers are used, your opinion may change instantly. A rather clean way out of this confusion is to extend the notation of numbers by explicitly mentioning the base.

$$10_8 = 3_8 + 5_8 \tag{1b}$$

Of course, when a whole document is in octal notation, a proper introduction is better than annotated numbers as used in formula 1b.

In terms of XML this can look like:

```
To what extent is math supposed to reflect the truth and nothing but
the truth? Consider the simple expression <math> <apply> <eq/> <cn>
10 </cn> <apply> <plus/> <cn> 3 </cn> <cn> 7 </cn> </apply> </apply>
</math>. Many readers will consider this the truth, but then, can we
assume that the decimal notation is used?

<formula>
  <math>
    <apply> <eq/>
      <cn> 10 </cn>
      <apply> <plus/>
        <cn> 3 </cn>
        <ci> x </ci>
      </apply>
    </apply>
  </math>
</formula>

In many elementary math books, you can find expressions like the
previous. Because in our daily life we use the decimal numbering system,
we can safely assume that <math> <apply> <eq/> <ci> x </ci> <cn> 7 </cn>
</apply> </math>. But, without explicitly mentioning this boundary
condition, more solutions are correct.

<formula label="octal" sublabel="a">
  <math>
    <apply> <eq/>
      <cn> 10 </cn>
      <apply> <plus/>
        <cn> 3 </cn>
        <cn> 5 </cn>
      </apply>
```

```
    </apply>
  </math>
</formula>

In <textref label="octal">formula</textref> we see an at first sight
wrong formula. But, if we tell you that octal numbers are used, your
opinion may change instantly. A rather clean way out of this confusion
is to extend the notation of numbers by explicitly mentioning the base.

<subformula label="octal base" sublabel="b">
  <math>
    <apply> <eq/>
      <cn type="integer" base="8"> 10 </cn>
      <apply> <plus/>
        <cn type="integer" base="8"> 3 </cn>
        <cn type="integer" base="8"> 5 </cn>
      </apply>
    </apply>
  </math>
</subformula>

Of course, when a whole document is in octal notation, a proper
introduction is better than annotated numbers as used in <textref
label="octal base">formula</textref>.
```

Math that is part of the text flow is automatically handled as in line math. If needed you can encapsulate the code in an *imath* environment. Display math is recognized as such when it is a separate paragraph, but since this is more a TEX feature than an XML one, you should encapsulate display math either in a *dmath* element or in a *formula* or *subformula* element.

Getting started

A comfortable way to get accustomed to MATHML is to make small documents of the following form:

```
\usemodule[mathml]

\starttext

\startXMLdata
<math>
  <apply> <cos/>
    <ci> x </ci>
  </apply>
</math>
\stopXMLdata

\stoptext
```

As you see, we can mix MATHML with normal TEX code. A document like this is processed in the normal way using TEXEXEC. If you also want to see the original code, you can say:

```
\usemodule[mathml]
```

```
\starttext

\startbuffer
<math>
  <apply> <cos/>
    <ci> x </ci>
  </apply>
</math>
\stopbuffer

\processXMLbuffer

\typebuffer

\stoptext
```

Like TEX and METAPOST code, buffers can contain MATHML code. The advantage of this method is that we only have to key in the data once. It also permits you to experiment with processing instructions.

```
\startbuffer[mml]
<math>
  <apply> <log/>
    <logbase> <cn> 3.5 </cn> </logbase>
    <ci> x </ci>
  </apply>
</math>
\stopbuffer

\startbuffer[pi]
 <?context-mathml-directive log location right?>
\stopbuffer

\processXMLbuffer[pi,mml]

\startbuffer[pi]
 <?context-mathml-directive log location left?>
\stopbuffer

\processXMLbuffer[pi,mml]
```

If you like coding your documents in TEX but want to experiment with MATHML, combining both languages in the way demonstrated here may be an option. When you provide enough structure in your TEX code, converting a document to XML is then not that hard to do. Where coding directly in XML is kind of annoying, coding MATHML is less cumbersome, because you can structure your formulas pretty well, especially since the fragments are small so that proper indentation is possible.

Further reading

The MathML spec

You can fetch the latest version of this document, which is written by the MATHML committee, can be fetched from the www.w3c.org web–site. Depending on the state of development, you can grab the draft, recommendation or standard.

The TEXbook
Once you have read this book, you will see why MATHML is not embraced by those who love to optimize the look and feel of their math formulas. Although your documents will be more consistent when you code in (content) MATHML, you also lose many fine points of math typesetting. Of course you can always fall back on the *annotation* element. This book is written by Donald Knuth and published by Addison Wesley.

The XML Companion
Written by Neil Bradley and published by Addison Wesley, this book is a good introduction to XML and its relatives. More compact but not less useful, is Robert Eckstein's *XML Pocket Reference*, published by O'Reilly.

XML in ConTEXt
This document describes how you can use CONTEXT for processing your XML documents. You may also want to take a look at the beginners manual, the reference manual, guides and examples that can be fetched from www.pragma-ade.com.

MathML in ConTEXt
We have keyed in a lot of realistic MATHML examples and turned them into a document suitable for viewing on your computer display. Over time, this collection will grow.