MAPS

NUMMER 32 • VOORJAAR 2005

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

Voorzitter
Hans Hagen
pragma@wxs.nl

Secretaris
Willi Egger
w.egger@boede.nl

Penningmeester
Wybo Dekker
wybo@servalys.nl

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

Frans Goddijn
frans@goddijn.com

Karel Wesseling
k.h.wesseling@planet.nl

Postadres
Nederlandstalige TEX
Gebruikersgroep
Maasstraat 2
5836 BB Sambeek

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

E-mail bestuur
ntg@ntg.nl

E-mail MAPS redactie
maps@ntg.nl

WWW
www.ntg.nl

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

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

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

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

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

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

Inhoudsopgave

Redactioneel

Voor u ligt het 32e nummer van de Maps, 148 pagina's dik deze keer: u had tenslotte nog wat van ons tegoed …Ook nu weer is de tweede helft ervan in kleur uitgevoerd, en wederom zorgen we dat u op alle markten thuisblijft: er zijn artikelen over CTAN, fonts, XML, taalkunde, metapost, scripting, LaTeX, ConTeXt en ook nog een uitgebreid verslag van EuroTeX 2005:

□ Piet van Oostrum *(Nieuws van CTAN)* maakt ons weer attent op nieuwe CTAN bijdragen; deze keer met veel aandacht voor PSTricks en teTeX.

□ het CTAN Team *(CTAN plans)* geeft ons inzicht in de ontwikkelingen op CTAN, het Comprehensive TEX Archive Network. Vooral de ontwikkeling van de TEXCatalogue komt aan de orde. Het is zeker de moeite waard daar eens een aandachtige blik op te werpen: ik kwam er zelf pas onlangs achter hoeveel informatie daar is te vinden (http://www.ctan.org/tex-archive/help/Catalogue)

□ Taco Hoekwater *(Metapost Developments – Spring 2005)* vertelt namens het MetaPost-team, waar hij deel van uitmaakt, hoe MetaPost (althans de ontwikkeling daarvan) een nieuw leven wordt ingeblazen nadat het op dat gebied lange tijd stil is geweest. Maar nu gaat het dan ook wel heel enthousiast verder: aan 3D-mogelijkheden wordt gewerkt, maar ook meer dimensies worden niet uitgesloten. Het ziet er dus naar uit dat zelfs het genereren van films tot de mogelijkheden zal gaan behoren. Of daarna nog meer dimensies aan de orde komen wordt niet geheel duidelijk.

□ Adam T. Lindsay *(Font Variants)* vertelt over de mogelijkheden om in ConTEXt font-varianten te selecteren. Fonts worden in toenemende mate voorzien van meer varianten dan we gewend waren. Zo heeft Antykwa Toruńska (ook in deze Maps beschreven) niet alleen normal en bold varianten, maar ook medium en light. Maar dan moeten er natuurlijk wel commando's zijn om die te kunnen selecteren. ConTEXt heeft daar nu voorzieningen voor. Laten we hopen dat daar voor LaTEX ook over wordt nagedacht.

□ Adam T. Lindsay *(Euler in Use)* borduurt nog eens flink door op dit thema met vele voorbeelden van het gebruik van het Euler font in al z'n varianten in ConTEXt. LaTEX-adepten kunnen hiervoor het eulervm-pakket gebruiken.

□ Taco Hoekwater *(Lettrines for ConTEXt)* laat evenzeer zien hoe ConTEXt door LaTEX bevrucht kan worden (waarmee ik zeker niet wil zeggen dat het omgekeerde niet minstens zo vaak voorkomt!). Hij bevredigt liefhebbers van initialen (dropped capitals) met een grote verscheidenheid daarvan.

□ Steve Peter *(TEX and Linguistics)* geeft een zeer interessante kijk op een wereld die velen van ons onbekend is, de linguistiek, maar waar TEX veel goed werk blijkt te kunnen verrichten. We maken kennis met de filologie, de fonologie, de paleografie, fonetica, semantiek en alle symbolieken die daarmee samenhangen.

□ Taco Hoekwater *(Controlling Acrobat Reader under X11)* geeft de script-schrijvers onder ons een handvat voor het openhouden van Acrobat Reader en zorgt er daarbij steeds voor dat de inhoud op tijd ververst wordt.

□ Oscar Boot en Frans Absil *(Met XML van database naar LaTEX)* geven een helder inzicht in alle aspecten van XML. We worden tegenwoordig doodgegooid met XML, zonder dat schijnt niets meer zinvol te zijn, maar de achtergronden had ik nooit begrepen. Nu wel.

□ Hendri Hondorp *(Bundeling van conferentieverslagen)* laat zien hoe gemakkelijk het is een conferentieverslag samen te stellen uit een grote verzameling pdf's van verschillende pluimage.

□ Frans Goddijn en Hendri Adriaens *(... three, two, one ...)* presenteren het etaremune-pakket (in Ruby-termen: 'enumerate'.reverse) voor LaTEX, ontworpen door Hendri naar aanleiding van een vraag op de ntg-nl maillijst en hilarisch verwoord door Frans. Ook als u voor het slapen gaan niet bij voorkeur TEX-artikelen leest kan ik u dit aanraden; u wordt de volgende morgen nog nalachend wakker.

□ Karel H. Wesseling *(Compiling MetaPost figures under ConTEXt)* bereidt u, als u een ConTEXt-gebruiker bent, alvast voor op het lange artikel dat hierna gaat komen. Hij laat zien hoe MetaPost-figuren met texexec gecompileerd kunnen worden en hoe de resulterende PostScript files vervolgens in ConTEXt kunnen worden opgenomen.

□ André Heck *(Learning MetaPost by doing)* schreef de klapper voor deze maps: een intensieve zelfstudie-

cursus voor Metapost, met vele voorbeelden en oefeningen. Wie vaak figuren genereert met allerhande pakketten en daar veel tijd mee kwijt is kan hier leren hoe het allemaal eenvoudiger, beter en TEX-geïntegreerder kan.

□ Janusz Marian Nowacki *(Antykwa Toruńska)* creëerde de meest recente versie van een font dat in 1960 voor het eerst in lood gegoten werd: Antykwa Toruńska, een font met een zeer uitgebreide tekenset (bijvoorbeeld Cyrillisch, Grieks, de meestgebruikte wiskundige en monetaire symbolen, extra ligaturen), evenals extra varianten (licht, normaal, halfvet en vet in normale en versmalde vorm).

□ Ernst van der Storm *(Variabele faxdocumenten aanmaken in LaTEX)* laat zien hoe zakelijke faxen geautomatiseerd uit een database gegenereerd kunnen worden. Alleen ter goedkeuring komt er nog een mens aan te pas.

□ Ernst van der Storm *(Stroomdiagrammen maken met flow)* dacht bij zijn vorige verhaal nog een stroomdiagram op te zullen nemen, maar hoewel dat onnodig bleek, wilde hij u toch niet onthouden hoe gemakkelijk zulke diagrammen met het LaTEX-pakket flow kunnen worden gemaakt.

□ Taco Hoekwater *(Verslag EuroTEX 2005)* is namens de NTG naar EuroTEX 2005 in Pont-à-Mousson geweest en doet uitgebreid verslag van de conferentie. Wie een idee wil krijgen van wat zich momenteel in de TEX-wereld afspeelt mag dit verhaal niet missen. Achtereenvolgens komen aan de orde: het LaTEX-pakket Mem voor meertalige documenten, Omega, een taxonomie voor automatische zetsystemen, talen voor TEX-implementatie, CTAN-onwikkelingen (zie hierboven), de conversie van MetaPost naar C, Metapost-ontwikkelingen (zie hierboven), het geraffineerde LaTEX examplep pakket voor verbatim en listings, de conversie met Majix van Word naar LaTEX via RTF en XML, de uitbreiding van TEX met een python-interpreter (QaTEX), het zetten van XML met behulp van TEX, LaTEX3-ontwikkelingen, de ConTEXt-layout van het Nieuw Archief voor de Wiskunde, het Zapfino OpenType font, namespaces, Contextgarden (zie vorige Maps), microtypografische extensies voor pdfTEX, de Newmath-encoding, de Latin Modern fonts, de paneldiscussie met Hermann Zapf en Donald Knuth, de ProTEXt-distributie voor Windows, MlBibTEX voor het schrijven van bibliografie-stijlen, de inzet van TEX voor de bestrijding van Kafkaiaanse verschijnselen, SaferTEX voor opmaak met minimale invoer, iWrapper voor uitwisseling van TeX-documenten, een bedrijfscatalogus in TEX, en het LaTEX-pakket bigfoot voor kritische edities.

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

Errata Maps 31

□ In het redactioneel bij Maps 31 werd Uwe Kern als mede-auteur van het `xkeyval`-pakket genoemd. Hij is echter alleen mede-auteur van het artikel. Hij heeft verder een macro en veel ideeën bijgedragen aan de totstandkoming van het `xkeyval` pakket.
In het betreffende artikel staat de webstek van Hendri door een fontprobleem verkeerd vermeld; het moet zijn: `http://stuwww.uvt.nl/~hendri`. Zijn emailadres is `hendri@uvt.nl`.

□ In het redactioneel werd ten onrechte gemeld dat Siep Kroonenberg's artikel *(Exact layout with LaTEX)* ging over het ontwerp van een briefhoofd; het ging over de implementie van een al ontworpen briefhoofd.

Wybo Dekker
wybo@servalys.nl

Een uittreksel uit de recente bijdragen in het CTAN archief

Keywords
TEX, LaTEX, packages, CTAN, classes

Abstract
Dit artikel beschrijft een aantal recente bijdragen uit het CTAN archief (en mogelijk andere bronnen op het Internet). De selectie is gebaseerd op wat ik zelf interessant vind en wat ik denk dat voor veel anderen interessant is. Het is dus 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 lekker te maken.

Inleiding

Deze aflevering van de rubriek "Nieuws van CTAN" bevat geen spectaculaire ontwikkelingen. Er zijn weinig opvallende nieuwe dingen verschenen in de afgelopen tijd, maar wel veel updates van bestaande packages en systemen.

PSTricks

PSTricks is een LaTEX package voor het maken van allerlei grafische vormen door middel van Postscript. Aangezien TEX zelf geen grafische (in de betekenis van plaatjes) primitieven heeft moet je voor plaatjes bijna altijd uitwijken naar iets buiten TEX. De mogelijkheden zijn:

□ Het gebruik van speciale fonts. Dit is de aanpak van de LaTEX `picture` omgeving. Hier worden rechte lijnen getekend door TEX's rules, en schuine lijnen en cirkels door middel van symbolen uit speciale fonts. Dit beperkt het aantal mogelijkheden enorm: zo kunnen lijnen alleen onder bepaalde hoeken getekend worden en zijn cirkels alleen in bepaalde afmetingen beschikbaar.

□ Het invoegen van plaatjes die door andere programma's gemaakt zijn (zogenaamde externe plaatjes). In LaTEX wordt hiervoor het commando `\includegraphics` gebruikt waarmee een plaatje uit een extern bestand wordt ingevoegd. TEX genereert een zogenaamd `\special` commando waarmee aan de printer-driver aangegeven wordt welk bestand ingevoegd moet worden, en wat andere informatie (bijvoorbeeld of het plaatje geschaald moet worden). Het echte invoegen gebeurt dan door de printer-driver. Bij gebruik van dvips als driver kunnen bijvoorbeeld EPS-plaatjes gebruikt worden. Andere printer-drivers kunnen andere formaten ondersteunen.
PdfTEX werkt op een vergelijkbare manier, alleen gebeurt het invoegen hier door PdfTEX zelf, omdat er geen aparte printer-driver gebruikt wordt. In dit geval zijn de mogelijke formaten: PDF, JPEG en PNG.

□ Het kan zijn dat de printer-driver andere `\special`-commando's ondersteunt waarmee grafische elementen gegenereerd kunnen worden. Bij PdfTEX kunnen primitieve PDF-elementen ingevoegd worden. Het LaTEX package pgf werkt op deze manier.

□ Een methode die tegenwoordig nauwelijks nog gebruikt wordt is het tekenen door middel van Metafont, waarbij de tekening opgedeeld wordt in blokjes die elk als een teken in een speciaal font gegenereerd wordt. Er wordt dan TEX code gegenereerd om alle tekens van dit font op de juiste wijze naast elkaar te zetten. Voor elke tekening heb je dan een apart font nodig.

□ Het gebruik van Metapost om te tekenen is eigenlijk een combinatie van enkele bovengenoemde methoden: Metapost genereert een (onvolledig) Postscript bestand dat door dvips aangevuld wordt met onder andere font-informatie. Bij gebruik van PdfTEX wordt de gegenereerde Postscript-code geïnterpreteerd en omgezet in PDF-primitieven. Dit kan omdat de door Metapost gegenereerde Postscript bijzonder eenvoudig van structuur is, en je dus niet een complete Postscript-interpreter nodig hebt om het te verwerken.

Het package PSTricks gebruikt de methode om `\special`-commando's te genereren voor de grafische elementen. Het genereert in het bijzonder Postscript code. PSTricks bestaat uit een groot aantal LaTEX-macro's voor allerlei figuurelementen zoals lijnen, krommen, cirkels, opgevulde vlakdelen enzovoort. De gebruiker zelf ziet geen Postscript-code: deze wordt intern gegenereerd. Het voordeel van deze methode is dat er grafische elementen gemaakt kun-

nen worden die een relatie tot de tekst hebben. Bijvoorbeeld een cirkel die een stuk tekst omsluit, waarbij de tekst door TEX gezet wordt. Of een pijl die van één stuk tekst wijst naar een ander stuk tekst. Bij het gebruik van externe plaatjes is dit niet mogelijk.

Het nadeel van PSTricks is dat het alleen bruikbaar is in een Postscript-omgeving. In de eerste plaats dus bij gebruik van een Postscript printer-driver, zoals dvips. Met een truc kan het echter ook gebruikt worden met PdfTEX. Hiervoor zijn de packages ps4pdf en pdftricks beschikbaar. Bij gebruik van een van deze packages wordt de Postscript-gerelateerde code binnen een speciale omgeving of een speciaal commando gezet. Deze gedeelten worden dan naar een bestand geschreven, verwerkt met het programma ghostscript en eventuele andere programma's die er PDF van maken. In de volgende LaTEX-run worden de gegenereerde PDF-bestanden dan opgenomen in het document. Enigszins omslachtig dus, ook omdat het document ervoor aangepast moet worden, maar het maakt de kracht van PSTricks wel beschikbaar.

PSTricks is oorspronkelijk ontwikkeld door Timothy Van Zandt, maar deze heeft sinds een aantal jaren de belangstelling ervoor verloren, of heeft er geen tijd meer voor. Na een vrij lange tijd van rust (waarbij het package moeite had om compatibel te blijven met de LaTEX-ontwikkelingen) heeft Denis Girou er werk aan verricht. Momenteel is Herbert Voss zeer actief en heeft het package weer een nieuw leven gekregen. Ook anderen hebben (onder andere via een mailinglijst) nuttige bijdragen geleverd. Er zijn verscheidene uitbreidingen bijgekomen en diverse onderdelen zijn bijgewerkt. PSTricks is te vinden op `CTAN:/graphics/pstricks`. Aanvullende packages:

- `pst-fr3d` Een package voor het tekenen van 3-dimensionale frames (boxen) met de macro `\PstFrameBoxThreeDMacro`. Hiermee kunnen ook 3-dimensionale knoppen getekend worden. `CTAN:graphics/pstricks/contrib/pst-fr3d`
- `pst-poly` Een package voor het tekenen van polygonen (veelhoeken). De nieuwe versie gebruikt het xkeyval package (zie verderop) en heeft een macro `\pspolygonbox` voor het tekenen van een polygoon om een tekst. `CTAN:graphics/pstricks/contrib/pst-poly`
- `pstricks-add` Een verzameling aanvullingen en bugfixes. `CTAN:graphics/pstricks/contrib/pstricks-add`
- `pst-optic` Een package om optische lenzen en spiegels te tekenen. `CTAN:graphics/pstricks/contrib/pst-optic`
- `pst-vue3d` Een package om perspectieftekeningen van 3-dimensionale objecten te tekenen. `CTAN:graphics/pstricks/contrib/pst-vue3d`
- `pst-3d` Een package voor het tekenen van 3-dimensionale aanzichten. `CTAN:graphics/pstricks/generic/pst-3d`
- `pst-3dplot` Een package voor parallelle projectie van 3-dimensionale objecten. `CTAN:graphics/pstricks/contrib/pst-3dplot`
- `pst-circ` Een package om elektrische en elektronische schakelingen te tekenen. `CTAN:graphics/pstricks/contrib/pst-circ`
- `pst-uml` Een package voor het tekenen van UML diagrammen. `CTAN:graphics/pstricks/contrib/pst-uml`
- `pst-light3d` Een package voor 3-dimensionale lichteffecten. `CTAN:graphics/pstricks/contrib/pst-light3d`
- `pst-math` Een package met mathematische functies zoals tan, acos, asin, cosh, sinh, tanh, acosh, asinh, atanh, exp. Deze kunnen onder andere gebruikt worden om grafieken van deze functies te tekenen. `CTAN:graphics/pstricks/contrib/pst-math`
- `pst-func` Een package met nog meer mathematische constructies.
  - polynomen $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$
  - Fourier reeksen $f(x) = \frac{a_0}{2} + a_1 \cos(\omega x) + \cdots + b_1 \sin(\omega x) + \cdots$
  - Bessel functie
  - Gauss functie

  `CTAN:graphics/pstricks/contrib/pst-func`
- `pst-geo` Een package voor geografische projecties. `CTAN:graphics/pstricks/contrib/pst-geo`
- `pst-bar` Een package voor het tekenen van staafdiagrammen (bar charts). `CTAN:graphics/pstricks/contrib/pst-bar`

TeTEX

TeTEX is het TEX systeem voor Unix(-achtige) systemen. Dus onder andere voor Linux maar ook voor Mac OS X. Thomas Esser, de persoon die teTEX in elkaar gezet heeft (bouwend op bestaande elementen zoals Web2C en vele packages) is altijd heel voorzichtig met het uitbrengen van nieuwe releases. Dit komt omdat hij er heel zeker van wil zijn dat het systeem correct loopt op een veelheid van systemen. Zijn manier van testen is zo rigoureus dat bèta-versies soms al jaren tot grote tevredenheid gebruikt worden voordat er een officiële release komt. Maar nu is er dan teTEX 3.0. Een probleem met deze versie kan zijn dat de directorystructuur (TDS) op sommige punten flink gewijzigd is. Aan de andere kant zijn er ook weer scripts voorhanden om bestaande directory-bomen om te zetten.

De nieuwe versie van teTEX heeft niet meer verschillende executables, zoals 'gewoon' TEX, PdfTEX,

en eTEX, maar alles wordt gedaan met de executable PdfeTEX die dan met verschillende opties aangestuurd wordt.

De nieuwe versie bevat onder andere ConTeXt versie 2005.01.24, de Latin Modern fonts, beamer, komascript en memoir packages, en natuurlijk vele oude en vertrouwde onderdelen.

Andere bijdragen

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

extarrows A LaTEX package voor extra pijl-symbolen. CTAN:macros/latex/contrib/extarrows

QCM A LaTEX package en een class voor het maken van multiple choice vragenlijsten. Het maakt zowel een invulformulier als een masker voor de controle. CTAN:macros/latex/contrib/qcm

greektex Een LaTEX package voor teksten die zowel teksten met Latijns letterschrift als Grieks bevatten. De README file spreekt over "a mixture of Greek and English", maar ik neem aan dat dit een kortzichtigheid in de beschrijving is, en dat andere talen die met Latijnse letters geschreven worden net zo goed gebruikt kunnen worden. Ook zegt de README: "To use them, you need to have an ASCII editor that can handle mixed Greek and English input," maar ASCII kent geen Griekse letters. Mijn inschatting is dat bedoeld is dat je een tekenverzameling kunt gebruiken die zowel Griekse als Latijnse tekens bevat.

LaTeXMng Een geïntegreerd LaTEX ontwikkelingssysteem (IDE) voor MS Windows. Dit is een shareware product. CTAN:nonfree/systems/win32/latexmng

BaKoMa TeX Word Een ander shareware LaTEX systeem (IDE). Deze heeft ook een WYSIWYG LaTEX-editor. CTAN:nonfree/systems/win32/bakoma. Meer informatie is te vinden op http://www.ctan.org/tex-archive/systems/win32/bakoma/programs/texword.html

ctx-math Een update van de ConTeXt math modules, die ik genoemd heb in de vorige MAPS. CTAN:macros/context/contrib/maths

numprint Dit LaTEX package heeft macro's om getallen op verschillende manieren af te drukken, bijvoorbeeld met een punt of komma om de drie cijfers, afgerond op een vast aantal decimalen, etc. CTAN:macros/latex/contrib/numprint

xkeyval Dit LaTEX package, ook de vorige keer besproken heeft intussen diverse updates doorgemaakt. CTAN:macros/generic/xkeyval

dvipng Een programma om dvi files om te zetten in PNG of GIF plaatjes. Het is een heel snel programma. Het is in eerste instantie ontwikkeld voor gebruik in preview-latex, een uitbreiding van de Emacs editor om in LaTEX files de grafische vorm van ingevoerde formules (en andere niet-tekstuele elementen) te kunnen zien, maar is ook als los programma te gebruiken. Dvipng is standaard geïnstalleerd in teTEX 3.0 systemen. CTAN:dviware/dvipng

ps4pdf Een package voor het gebruiken van Postscript-commando's binnen PdfLaTEX. Zie hierboven. CTAN:macros/latex/contrib/ps4pdf

pkfix Dit is een zelfstandig programma dat gebruikt kan worden om door dvips gegenereerde Postscript-bestanden die bitmap-fonts bevatten om te zetten naar een bestand met Type-1 fonts. Dit kan alleen als de Postscript-bestanden niet door vreselijk oude versies van dvips gegenereerd zijn. De oorspronkelijke dvips gaf geen informatie over de fonts mee in de Postscript-bestanden, alleen de bitmaps zelf. Modernere versies van dvips zetten extra informatie in de vorm van commentaarregels in het bestand en pkfix gebruikt dit om te bepalen welk font gebruikt werd. CTAN:support/pkfix

l2tabu (Das LaTeX-Sündenregister door Mark Trettin). Dit is een document dat allerlei dingen beschrijft die je *niet* moet doen in een LaTEX document. Bijvoorbeeld het gebruik van verouderde commando's en packages. De Engelse versie is **l2tabuen**. Misschien is dit iets om eens een keer in een MAPS op te nemen. CTAN:info/l2tabu/(german,english)

Piet van Oostrum
piet@cs.uu.nl

CTAN plans*

Abstract
The readers of Tugboat likely know the Comprehensive TEX Archive Network as a great pile of TEX stuff. That is, it is full of TEX materials, and it is great, but it is also something of a pile – a bit of a mess. We will sketch some plans for improving CTAN. As part of that, we will outline its architecture, history, and some present issues.

Preamble
Taking it from the top: CTAN is an Internet archive of material related to TEX that is available for public download. We now hold five gigabytes of material. Each day about ten thousand visitors download a large number of files, and others upload some more.

CTAN is the definitive collection of TEX material; we try hard to live up to the "Comprehensive" name. We hold everything from LaTEX macro packages up to entire distributions such as MikTEX and teTEX.

Present
CTAN is not a single site, but instead is a set of sites.

Three of these are core sites that actively manage the material, for instance installing new or updated packages.

- `dante.ctan.org` in Germany is sponsored by the German TEX group Dante, and is maintained by Rainer Schöpf and Rienhard Zierke.
- `cam.ctan.org` is sponsored by UK-TUG and is maintained by Robin Fairbairns, in England.
- `tug.ctan.org` in the USA is sponsored by the TUG, and is maintained by Jim Hefferon.

To ensure that we have the same policies, we rely on an active mailing list. To ensure that we hold the same material, we rely on a number of custom scripts.

In addition to the core sites, about seventy-five sites around the world help out by offering a mirror – every day they sync up with a core site and then make their copy also publicly available. This gives users more options and relieves the core sites of traffic. We keep a list of mirrors[1] and encourage people to use them.

Past
Before CTAN there were a number of sites with TEX materials available for download but there was no authoritative collection. At a podium discussion that Joachim Schrod organized at the 1991 EuroTEX conference, the idea arose to bring together the separate collections. (Joachim was involved because he ran one of the largest FTP servers in Germany at this time, and had heavily modified the basic tool `mirror.pl` for this purpose.)

CTAN was built in 1992, by Rainer Schöpf and Joachim Schrod in Germany, Sebastian Rahtz in the UK, and George Greenwade in the US (George came up with the name). The site structure was put together at the start of 1992 – Sebastian did the main work – and synchronized at the start of 1993. TUG provided a framework, a Technical Working Group, for this task's organization. CTAN was officially announced at the EuroTEX conference in Aston, 1993.

When CTAN was founded, the main way to access files over the network was FTP. So the system was built with an expectation that visitors would get materials that way (and perhaps also with an expectation that visitors are experienced users). In 1999 a try at a more extensive web interface was put on the TUG site, but it is weak and was not adopted by the other two core sites.

Problems
Nobody likes complainers, but to describe our plans we must describe the issues that they address. There are problems with the collection itself, and problems with the administration of that collection.

One problem with the collection is that it is big. Its structure has been outgrown and needs updating. Most people in the TEX community have had the experience of being unable to find the solution to a problem, only to later discover that a solution was in fact on CTAN. That is, we have found that as we have grown, the information available to archive users to help locate materials has not grown fast enough to allow them to find what they need. This has been eased by the metadata[2] assembled by Graham Williams into his *Catalogue*.[3] But nonetheless, we need to be more information-rich.

*First published in *TUGboat* **24**:2 (2003), pp. 205–207. Reprinted with permission.

A longstanding request about the collection[4] has been for CTAN to keep histories of packages, so that users can compile documents that rely on old versions. (Now, when a package author sends an update, we overwrite the old material.)

Another problem in the didn't-think-we'd-get-big (or-old) category involves mirrors. Often, the best way for a users to get a package from CTAN is to get its entire directory at once, so that they don't miss some files. To that end, the core sites support on-the-fly creation of `.zip` and `.tar.gz` file bundles.[5] The web front end at `http://www.ctan.org` uses this capability, and something like it must be a part of any future interface. However, it doesn't work with mirrors. In order to send users who want a bundle to a mirror, the system needs to know which mirrors correctly do on-the-fly-ing. So we wrote a script to check. When we ran it we found that not one mirror actually made both `.zip` and `.tar.gz` bundles without error. Consequently, the great majority of downloads come from the three core sites.

The flip side of people getting things from us is our getting things from the community. We are concerned by a trend whereby some package authors do not upload their work, but instead leave it on a personal web server. This is bad because it brings us back to the pre-CTAN days of materials that are scattered and that may disappear; that is, CTAN could end up not-comprehensive. It is also bad because, even if we know that the author's work exists, we have trouble gathering this material since the web protocol HTTP makes it hard for us to fetch things into our holdings.[6] Obviously we must work with the world as it is, but this is a problem.

The final collection issue that we will mention is that early developers, including Knuth, expected that most users would be fairly sophisticated: they would have developed basic typographic knowledge, and would need a minimum of computer and development support (e.g., they would write their own macros). This has not proved to be so. We perceive instead that the majority of TEX users want a distribution that comes with LATEX, etc., already set up. If they need to get something else, then they would like the distribution to have a module that can interface with CTAN and set it up for them. So a key goal is that we must – in conjunction with distributions – produce a system that meets those expectations.

We next describe some issues with the administration of the archive. Users do not see these directly, but they have an effect on what users do see.

The first is that we are a shoestring operation. The machines have been granted by user groups, but the critical network connections are donated by each maintainer's institution. The maintainers are far apart – some mailing list members have never met any other member – which slows progress and adds chances for miscommunication. All of the maintainers are volunteers and satisfy CTAN's time demands in the face of other things that must come first.

The time demands on maintainers are relevant because they have slowed our development. In particular, we must promptly handle materials that are uploaded. To help a reader get a sense of it – and for the satisfaction of bellyaching – consider a new package upload. The machine's maintainer gets it from the upload area and unpacks it. He checks the license, and decides where the package will go. He checks that there is a `README` file, and that there is documentation in PDF format that uses Type 1 fonts. Often, these checks involve corresponding with the author or with the CTAN mailing list, introducing a delay of a day or more. He then uses our custom install script to copy the material into the public area, and to trigger the mirroring process. He notifies the CTAN announcement list (and thus `comp.text.tex`). Finally, he edits the package *Catalogue* metadata and puts it into the CVS tree. In all, it averages perhaps a half hour per package.

More people in the administration might help. However, in addition to the necessary expertise with TEX, systems administration, and with the layout of CTAN, our work takes place in the context of an increasingly complex computing world. To name one example, in recent years licenses have become a big issue. We need people, but we also need a way to bring them in so that they can learn gradually.

Plans

Suppose that a colleague gives you a paper that requires a package not in your TEX setup. At present, you would visit CTAN, find the package, and then install it. Imagine if, instead, your TEX distribution got the package, installed it, and proceeded with running the paper, without your having to know anything about it. Technology that would make this kind of negotiation between the user's computer and CTAN reasonable is called "web services".[7] Our most important goal is to develop – in coordination with existing distributions – a capable spectrum of web services for CTAN to offer.

One step toward that, and toward accomplishing present goals also, is to better organize our holdings. For instance, we have already combined the subdirectories `supported` and `other` of `/macros/latex/contrib`, and we plan also to meld the `/info` and `/help` directories. A bigger job is to break all of our holdings into packages, and have each package in its own directory (no more `misc`). This is the natural way to answer a web services query, "what is the latest version of the file `f` in the package `p`?"

In support of web services, and also to help visitors get more information out we must get more informa-

tion into the *Catalogue*. We must both (1) expand the information of the kind that is in there already, and (2) also expand the kinds of information that can go in there.

Part of (1) is an effort to provide an easy way to edit this metadata on the web, for instance, when an author uploads or updates a package. As a bonus, this may provide a way to bring people in to help CTAN. A person could make a reasonable contribution by editing metadata and checking it into the CVS tree, without having to do system administration of a CTAN site.

An example of (2) is that we need to retain keywords, so that users can search for a package in this way (this search could happen on a CTAN web page, or from a user's desktop through a web service). For some time we've been discussing the underlying model for the metadata and in support of this, the *Catalogue* recently moved to a CVS tree.[8]

All that information should be in a database. This fits into our plans in many ways because, while CTAN grew up as an FTP archive, the web has changed everything and we need to fix our web system to be database-backed. It should provide an interface that is uniform across all three core sites.

That interface could allow users to find packages in alternate ways (this was first suggested by a comment made by Sebastian). At present, users can look through the FTP directories, can search the list of all files, can do a crude text search of the *Catalogue*, or can do a web search of our holdings using a standard search engine, and we've mentioned above that we'd like to add a keyword search. But, we'd also like to add a search of packages by functionality: a user trying to work with page headers might click through a branch of choices like `Top > LaTeX > Page layout > Headers and footers`.

One of the things that a modern site should have is the ability to search documentation. At present, many packages do not have documentation, or have it in a format that is not suitable as a search result (e.g., if the result of a search is a link to a `.dtx` file then clicking on it is unlikely to be helpful). We have begun enforcing that package contributers provide documentation just in PDF format, which is the only format that combines widespread accessability and typographic excellence.

Two problems listed above are the question of keeping package histories, and the question of mirrors providing `.tar.gz` and `.zip` bundles. We believe that we can solve these together, saving each version of a package as a bundle – then we have a bundle available, and mirrors need not create them.

We need to convince authors to upload their materials. We have in the past urged authors to do so,[9] but here also a volunteer, who can find materials and politely persuade authors, would help.

Finally, we are constantly thinking about the maintainer's work flow. There has been some wild talk about an administration GUI, but the problem is that there are so many exceptions and special cases that we often cannot see how to do it any other way than by hand.

Prognosis

We will close with a summary of where we are.

We plan to make CTAN more of a "Comprehensible" TEX Archive Network. These plans have been under discussion and in development for two to three years.

Dante has helped out greatly by sponsoring key people to come to meetings in the last two years, at Bremen and at Darmstadt, for in-person discussions. We must say that even beyond the grace of the invitations, the Dante people were kindness itself: in particular, Volker and Klaus moved the entire process forward greatly.

At present, the *Catalogue* format has been adjusted to allow development (in addition, moving it to the CVS tree was a big step forward because it allowed contributions in parallel), structures for the databases are in place, and we have beta code for the web editing of metadata and other parts of the new web system that all three core sites will use. Now, that system must be tested. Also, the static data (the web page content, the list of keywords, the tree of by-function categories, etc.) must be supplied. And finally, the data about the packages for the database must be be developed.

Briefly: progress is maddeningly slow, but there *is* progress.

Notes

1. See `http://www.dante.de/mirmon/` and also `ftp://tug.ctan.org/tex-archive/README.mirrors` .
2. Data about data, that is, information about the packages.
3. `http://www.ctan.org/tex-archive/info/Catalogue`
4. Notably by Nelson Beebe.
5. For example, visiting the url `ftp://ftp.ctan.org/tex-archive/macros/latex/contrib/shadethm.zip` will get the entire `shadethm` directory as a zip archive. From a command line FTP client, `get /tex-archive/macros/latex/contrib/shadethm.zip` will do the same.
6. On the scale at which we work, this is not as simple as just using a program like `wget`.
7. A rough definition is: a web server will respond to queries beyond just requests for pages.
8. `http://texcatalogue.sarovar.org`
9. If you have something that others would find useful, please consider sharing it!

Robin Fairbairns, Jim Hefferon,
Rainer Schöpf, Joachim Schrod,
Graham Williams, Reinhard Zierke
`ctan@dante.de`

MetaPost Developments – Spring 2005

Keywords
MetaPost, development, sarovar, bugs, extensions

Abstract
The MetaPost development team is pleased to announce version 0.9 of MetaPost. This article documents the changes since the previous version, and provides a road map for future development.

About this report

This article will be published in three different publications more or less simultaneously: In the proceedings of EuroTEX2005 and BachoTEX2005, as well as in issue 32 of the Maps. Parts of the contents of the starting page have already appeared in Maps 31, and some of the decisions documented by this report have not taken place until *after* EuroTEX, so this article is a bit of a cheat when it comes to chronology.

We apologise for that, but it makes more sense to write a single 'spring report' than to create 3 slightly different articles.

Introduction

The MetaPost system by John D. Hobby implements a picture-drawing language very much like that of Metafont, except that it outputs Encapsulated PostScript instead of bitmapped images. These graphics can be printed directly, or in embedded form from within TEX documents. MetaPost includes facilities for directly integrating TEX text and mathematics with the graphics.

In the summer of 2004, the version number of the MetaPost executable was still well below the 1.0 mark (0.641 was the current release at that time), but not much had happened in recent years. For some years, John Hobby simply could not find the time to solve the known bugs, let alone handle feature requests.

At that time, a group of people made a proposal to Dr. Hobby for the creation of a development team that would take care of the development of MetaPost from then on. Luckily, he agreed, on the condition that he will only allow tested code to be inserted into the MetaPost distribution. Among the currently active group are the following people: Karl Berry, Giuseppe Bilotta, Hans Hagen, Taco Hoekwater, and Bogusław Jackowski. This list is not fixed, people can be added or removed when needed or desired.

Contact information

A home page for MetaPost has been created on the TUG server http://www.tug.org/metapost, and TUG also provides a mailing list for discussions and questions (`metapost@tug.org`). Details on subscription to the mailing list can be found on the home page. MetaPost development is currently hosted at sarovar.org; visit http://www.sarovar.org/projects/metapost for the current development team members, sources, and much else.

Please report bugs and request enhancements either on the `metapost@tug.org` list, or through Sarovar. (Please do not send reports directly to John Hobby any more.)

The current release

MetaPost is Public Domain software. Recently the manuals (mpman and mpgraph) have been released under a BSD-ish license. Dylan Thurston at Debian converted the sources to LaTEX, and from now on they will become a standard part of the distribution.

We are pleased to announce MetaPost 0.9
For those of you who have noted the large gap between 0.641 and 0.9, and the fact that version numbers disagree with the slides that were presented at EuroTEX: We have deliberately chosen to step over some minor version numbers, such that the first major release by the current team (this release is planned for the autumn of 2005) can become version 1.0 without causing additional confusion.

Bug fixes

1. Documentation improvements: all known errata and typos have been removed, better explanations of e.g. dash patterns and dotlabel have been provided, and a number of omissions has been rectified.
2. The Bounding Box was not computed correctly when a filldraw command with a noticeable pen size was used at the edge of the picture.
3. Paths starting with degenerate constructions like in this example:

```
draw (0,0)--(0,0)--(0,0)--...;
```

could cause an overflow error of MetaPosts internal memory.

4. The PostScript output could accidentally contain 8-bit characters within PostScript strings in previous versions because the `is_printable` test was shared between terminal printing and PostScript printing.
5. A bug has been found in the assignment of serial numbers to independent variables in metafont 2.71828. This bug affected MP as well, and the same patch has been applied. Diagnosis and patch were supplied by Thorsten Dahlheimer.
6. The return value of the `turningnumber` primitive was sometimes wrong in unexpected ways (`turningnumber` is supposed to report whether a cyclic path turns clockwise or anticlockwise). The new implementation is still sometimes wrong when there are strange path segments involved, but in a much more predictable way: the new code always draws straight lines between the actual points, and calculates the turningnumber based on that path instead of the actual path. The effect is that cusps and loops within segments are now completely ignored. A more thorough fix of turningnumber is planned for the next release.
7. There was an 'off by one' error in dvitomp, in the interpretation of virtual fonts.
8. The `mpto` command (which is part of `makempx`) uses a new and improved TEX macro for the generation of labels, making it more robust with respect to strange user–supplied code within the actual label. The old version would sometimes loose track of the label's size information.
9. A few macro bugs have been solved: a missing colon in boxes.mp that has been added, a missing save in mfplain.mp has been added, and the `generisize` macro in boxes.mp has been fixed so that it now accepts `[[` as a valid variable name.

mpversion

Immediately after the presentation at EuroTEX, it became clear that both Hans Hagen and Donald Knuth have a considerable amount of pictures that more or less count on bugs in 0.641. For Knuth, this is the 'incorrect label size' that he himself reported as a bug to JDH. For Hans, it is the miscalculation of the picture's bounding box when using filldraw.

After some thinking, we decided it was best to add the already planned `mpversion` primitive right away, instead of waiting for a feature release. That way, this bug fix release can be identified from within the language. The result of `mpversion` is of type `<string>`, and those version strings are of the form `<major digit>.<minor digit>`. Possible trailing digits indicate beta releases (when the first extra digit is a nine), bug fix releases (when the first extra digit is a zero), or releases for development purposes only. For example, `"0.89"`, `"0.891"` and `"0.892"` were used to designate beta's for the current release.

For instance:

```
if known mpversion:
  message "mp = " \& mpversion;
  if mpversion > "1.0":
    message "time has flown by"
  fi
fi;
```

prints 'mp = 0.9' on the terminal. (Incidentally, > does a simple ASCII comparison of strings; that works here, because of our particular version numbering—until and unless MetaPost reaches version 10!)

The version number is also included in the `Creator` comment in the PostScript output.

What is new

1. It has been mentioned already that the LaTEX sources of the mpman, mpintro, and mpgraph manuals have become part of the distribution package.
2. There is a new internal string variable called `mpversion`, that reports the current MetaPost version, as explained above.
3. The macro file `TEX.mp` acquired two additional routines to facilitate using LaTEX to typeset labels: `TEXPRE` and `TEXPOST`. Their values are remembered, and included before and after (respectively) each call to `TEX`. Otherwise, each `TEX` call is effectively typeset independently. `TEX` calls also do not interfere with uses of `verbatimtex`. An example is given below.
4. MetaPost now writes a %%HiReSBoundingBox comment to the PostScript output file. The values of the picture's bounding box in this comment are not rounded to integer values as the ones in %%BoundingBox are.
5. The EPS output no longer contains actual spaces within PostScript strings. For example, the output of

```
label("a space")
```

```
(a\040space) cmr10 9.96265 fshow
```

instead of `(a space)`. This makes the generated PostScript easier to tokenize for post–processors.
6. The EPS output now also has a %%BeginProlog DSC comment as well as %%EndProlog

7. The comments in the Web source have been changed to point out that on modern machines, acquiring the random seed has actually become a system-dependant operation: a granularity of whole seconds is no longer small enough on new machines, where it is has become possible to start two separate mpost runs within one second.
8. The 'newer' command now accepts more than two arguments. All of the supplied file arguments are tested in turn.

TEX.mp extension
Here is an example of how to use the two new macros in `TEX.mp`, using the LaTeX inline math command `\( \)` instead of dollar signs:

```
input TEX;
TEXPRE("%&latex" &
       char(10) &
       "\documentclass{article}" &
       "\begin{document}");
TEXPOST("\end{document}");
beginfig(100)
  last := 10;
  for i := 0 upto last:
   label(TEX("\(n_{" & decimal(i) & "}\)"),
         (5mm*i,0));
  endfor
  ...
endfig;
```

- The `%&latex` causes LaTeX to be invoked instead of TeX. (See below, also.) Web2C- and MiKTeX-based TeX implementations, at least, understand this `%&` specification; see, e.g., the Web2C documentation for details, http://tug.org/web2c. (Information on how to do the same with other systems would be most welcome.)
- The `char(10)` puts a newline (ASCII character code 10, decimal) in the output.
- The `\documentclass...` is the usual way to start a LaTeX document.
- The `TEXPOST("\end{document}")` is not strictly necessary, due to the behaviour of `mpto`, but it is safer to include it.

Bugs remaining

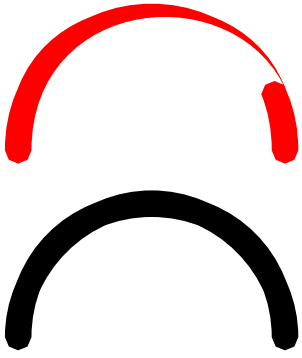
A few bugs have not been solved yet. We promise that those will be addressed before the following (1.0) release.

web2c–specific problems
These are usually caused by re–loading the mem file under different memory size settings than it was dumped with:

- strings reloading problems
- specials missing from the first output file

The solution is probably to save all of the configurable values inside the MEM file (like TeX does already for the format files).

Polygonal pen with 180 degree angles
When a path segment turns exactly 180 degrees, MetaPost cannot decide which side of a polygonal pen to use on the 'return trip'.



```
beginfig(1);
pickup makepen(fullcircle scaled 10 pt);
draw (0,0){up}..(50,50)..{down}(100,0);
draw (0,70){up}..{down}(100,70) withcolor red;
endfig;
end;
```

This happens because MetaPost cannot differentiate between an angle of 180 degrees and an angle of −180 degrees. The black line is what *should* be drawn, but the grey one is the one that current MetaPost actually outputs.

linecaps for polygonal pens
MetaPost does no attempt to handle linecap for polygonal pens. This can produce unexpected results when the pen has a 'funny' shape, as can be seen in this picture:

```
beginfig(1);
linecap := butt;
draw (0,0)--(90,0) withpen pencircle scaled 10;
draw (0,20)--(90,20) withpen pensquare
                     scaled 10 withcolor red;
endfig;
end
```

On the left side of the grey line, MetaPost switches from the top left corner of the square pen to the bottom right corner of the pen. This behaviour can be rationalised after some thinking, but at first sight, the produced diagonal line is a very strange phenomenon indeed.

graph.mp bugs
There are a number of bugs in the graph.mp package. None of these have been tackled yet because in the current team nobody has used the package extensively. Most of these problems require familiarity with the macros and their usage:

- □ arrow heads are sometimes mis-rotated
- □ axis labels centre strangely
- □ data file format is too strict
- □ exponents cannot have a capital E, as in 1E4.
- □ curves from data are always polygonal
- □ axes can only be in the left of the graph, not through the origin.
- □ frames can only be avoided if an internal variable is set to false.

Future plans

The Birds of a Feather (special interest) gathering at EuroTEX had a slow start before dinner, followed by a much better attended continuation after dinner. This session was about all desired new features, both identification and prioritisation. We started with the list from the slides (and that list, in turn, was created by compiling the feature requests in the Sarovar database).

For the next release
After a few hours of lively discussion, The top of the todo list became the following:

1. Create an independent packaging system.
   This item was not really on anybody's wishlist, but it simply needs to be done. The current distribution depends on Web2C (it is, in fact, a stripped down snapshot of TEXLive) and we are far from happy with that. Even when we ignore the `troff` support, we feel that we should make the build process (how to compile MetaPost from source) as simple as possible.
   Taco intends to write a replacement for the current 'tangle + convert' construction. That replacement will output C code directly, and will maintain as much of the symbolic names as possible. In doing so, it will make source–level debugging easier.
2. Remove `emergency_line_length` from the Web source.
   The web sources define a variable named `emergency_line_length`, with a compile–time value of 255, that is used solely to limit the maximum lines size of the PostScript output under unexpected conditions.
   The benefits of this check are marginal and it interferes with web2c's dynamic arrays considerably.
3. Fix the still present bugs.
   As explained above.
4. Allow access to the internally computed envelope of a path drawn with a polygonal pen.

```
<path>:= envelope <acyclic path> withpen <pen>;
<path>:= inner envelope <cyclic path> withpe...
<path>:= outer envelope <cyclic path> withpe...
```

It has meanwhile been noted by several people that these actual names will not work, because MetaPost already assigns a meaning to `inner` and `outer`. The final implementation will therefore have to use different keywords. Those names are still under consideration.
The implementation of this extension would be easy. Because MetaPost already does the needed work when writing the PostScript output file, it is just a matter of making this information available to the macro language.

In all likelihood, this is the limit of what we can put into the next release, version 1.0, to be released in the autumn of 2005.

Possibly for the next release

1. Allow basic EPS inclusion.
   The idea is to allow the same kind of EPS inclusion one can find in the `dvips` driver. This EPS file could contain PostScript drawing commands or a bitmap expressed using PostScript's `image` operator.
2. Implement something that is like TEX's `\special`.
   This is so that user–supplied stuff can end up in the middle of the PostScript output, not only at the very top. Both Jacko and Hans need this to do special effects in post-processing.
3. Improvements to string handling.
   This was put on the board for Hans, who hopes that it is possible to improve the primitives that deal with strings.
4. Implement edge structures.
   Edge structures is Metafont's data model to express images.
   This item is a huge undertaking, because it implies the recreation of all of Metafont's edge structure operations like "good.x", "withweight"

and "cullit", and some of those in turn imply real unfill and overlap removal routines.

More wishes
A number of items were kept for the future, but tagged as 'more thinking needed'. These are things for which a partial solution would be easy to do, but for a complete solution it is not even clear how it should be implemented

- □ Support for alternative colour models.
  Cmyk and grayscale would be easy but not very useful. Named colours and especially transparency would be very useful to have, but implementing that would be quite hard.
- □ Macro language enhancements
  The first item on this list is the desire to have namespace support. Everybody knows what is meant by this, but a precise specification is lacking.
  A second one of these is that symbolic token existence tests are missing from the language at the moment; `if known` is not always good enough. For instance, you may want to know if a variable name has been declared, regardless of whether or not it has a known value at the moment.
  Yet another is that `let` behaves a bit strange compared to the `\let` command in TEX.
- □ Support for 3D (or more dimensions).
  Perhaps this support can be added gradually, but that needs to be investigated in the Web source of MetaPost first.

Even more wishes
Some ideas were tagged as '(much) later':

- □ virtual font support.
  Any code on this might become superseded within a year because of upcoming improvements to pdfTEX, so it makes no sense to do anything in this area yet.
- □ Proper EPSF creation.
  Embedded and possibly subsetted fonts should be included in the output.
- □ PDF output.
  This has to be very much later, because it needs lots of extra code.
- □ Bitmapped output format
  Some form of netpbm probably, to debug the edge structures. There is no sense in adding it before the edge structures are implement, because then it would need ghostscript or a similar PostScript rasterizer.

Remaining wishes

The rest of the list was agreed on as being nice, but remained unprioritized (by mistake or lack of time):

- □ A way to check if a path is completely within another path
- □ Non-continuous paths (as in PostScript)
- □ XML as an output format (nicer to parse for post-processors)
- □ Fixed number arithmetic using 64 bits instead of the current 32 bits (also known as 'megapost').
- □ Unicode/UTF-8 support for easier string handling

Scratched ideas

Some ideas were scratched for the moment, either because it is strictly speaking unneeded (a macro solution is doable), or because there has not been a well–argumented request for that feature.

Among those are some shortcuts for existing macros and a switch to output PostScript level 2 or 3.

Acknowledgements

We would like to thank everybody who has helped us creating this new release of MetaPost, especially to John D. Hobby for allowing us to do so.

Our next report will be available in the autumn of 2005.

Taco Hoekwater / MetaPost Team
metapost@tug.org

Font Variants

A new ConTEXt feature for organising rich fonts

Abstract
Font Variants are a new (meta-)feature in ConTEXt, offering the opportunity of easier access to advanced or unusual font features, without the hassle of using full typescript switches. This article briefly runs through the basic theory and practice of so–called font variants, and gives a few strategies for adapting it for your own uses.

What are font variants?

Font Variants are a somewhat nebulous concept, hard to define precisely. They can be any 'variation' on a main font, where the variation can be glyph shape, weight, width, or other font feature. Often, variants share the same *characters*, but have different *glyphs*, and therefore different encodings. Other times variants will be a visual variation sharing the same encoding.

This distinction between features and encodings can be blurred when you consider different fonts may have different strategies for encodings. Consider the different strategies needed with a source font that contains old–style figures and small caps glyphs within the same font as lining figures and normal lower case, and one that has those characters within a 'small caps' font. This is one of the reasons why font variants were introduced, to provide a layer of abstraction to hide the font–specific details on how the variants are implemented.

At surface level, a font variant is a temporary font switch, to be used as a grouped command, with a user–determined argument determining the variation to select. Font variants may exist for each font style (e.g., `\rm`, `\ss`) and alternative (e.g., `\bf`, `\it`), but the features do not 'stack' atop one another. A bit of code for illustration, with the results in figure 1:

```
Hello all. This is a bit of text written for
MAPS \#32 on {\Var[osf]31} March
{\Var[osf]2005} in {\it Lancaster,
\Var[sc]uk}. {\bf I can't \Var[lt]expect}
accumulative {\Var[lt]effects} with
variants like{\Var[cond] condensed
{\Var[sc]caps}}, however.
```

Hello all. This is a bit of text written for MAPS #32 on 31 March 2005 in *Lancaster, UK*. **I can't expect** accumulative effects with variants like condensed CAPS, however.

Figure 1. Antykwa Toruńska, with variants

The command `\Var[]` is shorthand for `\variant`, which can be used in case of a name clash (e.g., from mathematics).

Example usage

A couple font variants have already been named within current ConTEXt distributions, as the features were available in relatively new publically available fonts. Antykwa Toruńska is now a fully–featured serif family with many features available as variants. We see three classes of variation, in fact:

Glyph variation Beside each of the normal, roman fonts with lining figures, there lies a variant with variant glyphs: lower case letters are replaced with small caps, and lining numerals are replaced with old-style (lower case) numbers.

Weight variation Antykwa Toruńska offers four sets of weights: bold, medium, regular, and light. ConTEXt ordinarily allows easy access to only a normal and bold pair of weights, so the other pair can be accessed as a lighter (or darker) variant of the pair of fonts.

Width variation Antykwa Toruńska also offers condensed versions of the same fonts. These are defined as a typescript family themselves, but are also acces-

sible as a condensed variant on the normal width. Conversely, the normal width is accessible from the condensed family as an expanded variant.

Usage then follows with the defined variants for the given family. For the normal width, normal weight Antykwa Toruńska family that is set up as follows:

```
\definetypeface[antt][rm][serif]
 [antykwa-torunska][default][encoding=texnansi]
```

...the following predefined variants are defined: 'osf' (old–style figures), 'sc' (small caps), 'lt' (light), and 'cond' (condensed). Markup proceeds with the `\Var[]` commands acting as grouped font switches.

Implementing variants yourself

There is one key command to set up font variants for implementors:
`\definefontvariant[style][name][suffix]`.
The *style* argument registers the variants with the Serif or Sans family of fonts. The *name* argument registers the name to be called within `\Var[name]`. The suffix argument reaches down within the typescripts and looks for a fontsynonym `style`*variation*`suffix`. For example, given a definition:

```
\definefontvariant[Serif][osf][-OldStyle]
```

and in an italic context, invoking the variant switch `\Var[osf]` means that that ConTEXt then tries to switch to the font `SerifItalic-Oldstyle`.

That simple name resolution means that the other half of implementing font variants consists of naming font synonyms within typescripts correctly. The Antykwa Toruńska typescripts in `type-syn` provide an extended example. There are a few simple guidelines to follow, however:

- You should always create a font synonym for `SerifRegular` to `Serif` and `SansRegular` to `Sans`. These are the fallback cases, and need to be accommodated by the somewhat simple–minded name resolution.
- After the fallback case, you need to define seven font synonyms per variant: Regular, RegularItalic, RegularSlanted, the same for Bold, and Caps.
- The synonyms can be arbitrary, so long as they resolve to actual fonts on your system. This is typically achieved via the encoding synonyms.

Design strategies

[N.B. This section is for people comfortable working with font encoding files, and understanding the occasional need to make one up. Don't be discouraged or distracted if you're not one of them.]

If you need to install a font yourself, then you may need to concern yourself with devising a new encoding, especially if you're dealing with a font that includes several glyph variations within the same font (e.g., *a, Asmall, a.swash*). You should go ahead and write your own encoding based on an existing one, for example a `texnansi` variant that has small caps and old-style figures. The name should then be `baseencoding-`*`variantname`*`.enc`, for ease of installation.

TEXFont has an option (*–variant=*) for using variant *encodings*, which causes it to look for an `.enc` file as above. This variant encoding is used in the creation of the `.tfm` file and within the `.map` file, but on the ConTEXt side, the font acts as if it were in the base encoding. This way, ConTEXt is able to use its internal encodings correctly, but externally (in the driver that deals with fonts), the actual glyphs are those selected in your custom `.enc` file.

Conclusion

Font variants are actually a fairly simple mechanism, but coupled with correctly–written typescripts, and by following some simple conventions, you can unlock the potential of today's increasingly sophisticated and rich fonts. Antykwa Toruńska has font variant support built into ConTEXt already, so you can start experimenting with support on a user level today.

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

Euler in Use

ConTEXt support for the Euler math font, with examples

Abstract
The Euler math font was designed by Hermann Zapf. ConTEXt support was limited until now. We show how to use the Eulervm LaTEX package in combination with some new math definitions and typescripts to give a more informal look to your equations.

Keywords
ConTeXt, fonts, Euler, math

Introduction

The Euler math font was designed by Hermann Zapf. The underlying philosophy of Zapf's Euler design was to "capture the flavor of mathematics as it might be written by a mathematician with excellent handwriting." ConTEXt support had previously been limited, but now includes nearly all features available with ConTEXt's mathematics support. We show how to use the Eulervm LaTEX package in combination with some relatively new math definitions and typescripts to give a new look to your equations. Along the way, we provide several examples using ConTEXt's typescript and typeface mechanisms.

Although Euler has a somewhat informal feel that may make it inappropriate in certain situations, it does have certain advantages: as it does not directly mirror the form of any particular roman font (and has a robust weight), it mates with many more fonts than Computer Modern Math or others that are available. As D.E. Knuth himself had a hand in its creation, it offers features like optical scaling (e.g., script and scriptscript sizes) and multiple weights (i.e., bold math) not seen in most 'alternative' fonts. Most of the disadvantages that previously came with Euler use, chief among them poor glyph coverage, have disappeared with Walter Schmidt's Euler-VM package.

Typography can be very individual. I present the following possibilities as mere suggestions of how to get started. Please feel free to mix and match fonts and typescripts as you desire . . . I would hate for my peculiar tastes to be taken as some sort of dogma. My main motivation was to carry the freely available TEX fonts as far as possible.

Many thanks to Otared Kavian, who provided some of the source material that I liberally pilfered, and good beta-testing and initial feedback. Hans Hagen and Guy Worthington also provided advice, test material, and feedback at the early stages. Walter Schmidt provided the `Eulervm` package that I heavily rely upon.

Installation and basic usage

The Euler typescript requires the `Eulervm` virtual font package. For the `eulervm` file see http://www.ctan.org/tex-archive/help/Catalogue/entries/eulervm.html. Once

you uncompress/unpack the file, examine the contents. It should contain, among others, two directories, `vf` and `tfm` the contents of the `vf` directory must be installed into the directory $TEXMFFONT/fonts/vf/public/eulervm, and the contents of the `tfm` directory must be installed into the directory $TEXMFFONT/fonts/tfm/public /eulervm. You are welcome to install the other, LaTeX–specific files if you like, but only the font files are of concern for use with ConTEXt. Don't forget to update your TEX hashes.

The other files that ConTEXt support relies upon are now integrated with the standard distribution. This includes several typescripts and a `math-eul` file that gives the location of specific math characters.

With the virtual fonts installed, you can test things with this simple document:

```
\definetypeface[eulermath][mm][math][euler][euler][rscale=1]
\setupbodyfont [eulermath]
\starttext
\showmathcharacters
\stoptext
```

This should yield the set of math characters shown at the end of the article in figure▶. Most distinctive to Euler are the greek and roman letters, while other characters are drawn from the default Computer Modern family.

Usage follows the general model shown above:

```
\definetypeface[name][mm][math][euler][euler][rscale=1]
```

The fourth argument ('euler') means to call the euler `name` and `encoding` typescripts, while the fifth argument ('euler') calls the Euler `size` (optical scaling) typescripts. The optional sixth argument allows a scaling to be applied to the Euler fonts relative to the main face. The whole cluster of font families grouped by `name` is then called with:

```
\setupbodyfont[name]
```

This will be illustrated throughout the rest of the article.

Examples of Euler in use

As Palatino was designed by Hermann Zapf, as Euler was, it forms the basis of the first suitable pairing in figure 1. We use Helvetica as the sans, not from any love, but because it is commonly and freely available.

```
\definetypeface[peul][rm][serif][palatino] [default][encoding=texnansi]
\definetypeface[peul][ss][sans] [helvetica][default][encoding=texnansi,
                                                    rscale=.90]
\definetypeface[peul][tt][mono] [modern]   [default][encoding=texnansi,
                                                    rscale=1.1]
\definetypeface[peul][mm][math] [euler]    [euler]  [rscale=1.03]
```

1 Mixed faces on a page

In order to demonstrate the current typeface combination, we show the current faces *in situ* with an unusual, perhaps inordinate amount of ***style* switching**, as well as using macros that switch faces in order to demonstrate **key terms**, such as those that might be used in a textbook. Naturally, this somewhat affected approach will yield a fairly *extreme* page, but it can be worthwhile to look at a worst–case scenario, as well as a more moderate case, as with the other example.

Some of the most typographically offensive pages I have seen have been in international standards, with their `codeSnippets`, terminology, perhaps even mixed with a $\sqrt{x}$ radical function or two. Various typographers would certainly take issue with these practices, but on the other hand, certain conventions have been established.

2 MPEG-7 Audio

To extract the spectrum spread:

1. Calculate the power spectrum, $P'_x(n)$, and corresponding frequencies, $f(n)$, of the waveform as for `AudioSpectrumCentroid` extraction, parts a–b.

2. Calculate the spectrum centroid, C, as described in `AudioSpectrumCentroid` extraction in part d.

3. Calculate the spectrum spread, S, as the RMS deviation with respect to the centroid, on an octave scale:

$$S = \sqrt{\frac{\sum_n \left(\log_2(\frac{f(n)}{1000}) - C\right)^2 \cdot P'_x(n)}{\sum_n P'_x(n)}}$$

Figure 1. Palatino, Helvetica, CM-Typewriter, and Euler

The second pairing is the original usage of Euler: with DEK's Concrete font from *Concrete Mathematics* (shown in figure 2). We have to use a bit of a hack to get there, as the only freely available PostScript version[1] of the Concrete family is an adaptation of the font for Polish. As the encoding is a superset of the standard 7–bit TEX encoding, and already has ConTeXt support, it is easy to call this typescript as an

extension of the pre–existing ConTEXt typescripts for Concrete. It could optionally be saved and called as a typescript on its own:

```
%\definetypescript[serif][concrete][pl0]
\loadmapfile      [pcr.map]
\definefontsynonym[ccr10]  [pcr10]  [encoding=pl0]
\definefontsynonym[ccti10] [pcti10] [encoding=pl0]
\definefontsynonym[ccsl10] [pcsl10] [encoding=pl0]
\definefontsynonym[cccsc10][pccsc10][encoding=pl0]
%\stoptypescript
```

Here are some of the simple operations we can do with the O-notation:

$$f(n) = O(f(n)), \tag{5}$$
$$c.O(f(n)) = O(f(n)), \qquad \text{if } c \text{ is a constant}, \tag{6}$$
$$O(f(n)) + O(f(n)) = O(f(n)), \tag{7}$$
$$O(O(f(n))) = O(f(n)), \tag{8}$$
$$O(f(n))O(g(n)) = O(f(n)g(n)), \tag{9}$$
$$O(f(n)g(n)) = f(n)O(g(n)). \tag{10}$$

The O-notation is also frequently used with functions of a complex variable z, in the neighborhood of $z = 0$. We write $O(f(z))$ to stand for any quantity $g(z)$ such that $|g(z)| \leqslant M|f(z)|$ whenever $|z| < r$. (As before, M and r are unspecified constants, although we could specify them if we wanted to.) The context of O-notation should always identify the variable that is involved and the range of that variable. When the variable is called n, we implicitly assume that $O(f(n))$ refers to functions of a large integer n; when the variable is called z, we implicitly assume that $O(f(z))$ refers to functions of a small complex number z.

Figure 2. Concrete and Euler

As the encoding is somewhat special, this combination may be of limited use with demanding or extensive accent requirements. Still, it is interesting and informative to see Euler with its first husband, so to speak:

```
\definetypeface[ceul][rm][serif][concrete][default][encoding=default]
\definetypeface[ceul][tt][mono] [modern]   [default][encoding=texnansi,
                                                      rscale=1.05]
\definetypeface[ceul][mm][math] [euler]    [euler]   [rscale=1]
```

The `default` encoding listed above is correct if used with the bare synonyms shown above. If enclosed in a typescript, it is most correct to call (and label) that typescript with the `pl0` encoding.

Here are some of the simple operations we can do with the O-notation:

$$f(n) = O(f(n)), \tag{5}$$
$$c.O(f(n)) = O(f(n)), \qquad \text{if } c \text{ is a constant}, \tag{6}$$
$$O(f(n)) + O(f(n)) = O(f(n)), \tag{7}$$
$$O(O(f(n))) = O(f(n)), \tag{8}$$
$$O(f(n))O(g(n)) = O(f(n)g(n)), \tag{9}$$
$$O(f(n)g(n)) = f(n)O(g(n)). \tag{10}$$

The O-notation is also frequently used with functions of a complex variable z, in the neighborhood of $z = 0$. We write $O(f(z))$ to stand for any quantity $g(z)$ such that $|g(z)| \leqslant M|f(z)|$ whenever $|z| < r$. (As before, M and r are unspecified constants, although we could specify them if we wanted to.) The context of O-notation should always identify the variable that is involved and the range of that variable. When the variable is called n, we implicitly assume that $O(f(n))$ refers to functions of a large integer n; when the variable is called z, we implicitly assume that $O(f(z))$ refers to functions of a small complex number z.

Figure 3. Antykwa Toruńska and Euler

I have a certain fondness for the somewhat alien (to western eyes), but utterly charming Antykwa Toruńska family[2], which in recent months has got an update to be one of the most complete font families freely available in TEX land (it is illustrated in figure 3). I discuss in another article methods of unlocking the features in this family, but for now we examine the basic pairing of Antykwa Toruńska with Euler, which seem to hold echoes of each others' letterforms:

```
\loadmapfile[texnansi-antt.map]
\definetypeface[aeul][rm][serif][antykwa-torunska][default]
                                                    [encoding=texnansi]
\definetypeface[aeul][tt][mono] [modern]          [default]
                                                    [encoding=texnansi,
                                                           rscale=1.05]
\definetypeface[aeul][mm][math] [euler]           [euler]   [rscale=1]
```

1 Mixed faces on a page

In order to demonstrate the current typeface combination, we show the current faces *in situ* with an unusual, perhaps inordinate amount of ***style* switching**, as well as using macros that switch faces in order to demonstrate **key terms**, such as those that might be used in a textbook. Naturally, this somewhat affected approach will yield a fairly *extreme* page, but it can be worthwhile to look at a worst–case scenario, as well as a more moderate case, as with the other example.

Some of the most typographically offensive pages I have seen have been in international standards, with their `codeSnippets`, terminology, perhaps even mixed with a $\sqrt{x}$ radical function or two. Various typographers would certainly take issue with these practices, but on the other hand, certain conventions have been established.

2 MPEG-7 Audio

To extract the spectrum spread:

1. Calculate the power spectrum, $\mathbf{P}'_x(n)$, and corresponding frequencies, $f(n)$, of the waveform as for `AudioSpectrumCentroid` extraction, parts a–b.

2. Calculate the spectrum centroid, $\mathbf{C}$, as described in `AudioSpectrumCentroid` extraction in part d.

3. Calculate the spectrum spread, $\mathbf{S}$, as the RMS deviation with respect to the centroid, on an octave scale:

$$S = \sqrt{\frac{\sum_n \left(\log_2(\frac{f(n)}{1000}) - C\right)^2 \cdot P'_x(n)}{\sum_n P'_x(n)}}$$

Figure 4. Bitstream Charter, Bera Sans and Mono, and Euler

Finally, we examine a popular combination between Bitstream Charter and Euler. The editors of MAPS seem to like it, and I found it useful as well in my first major ConTEXt excursion. In this variation, I marry Bera[3] (the TEXified version of the Open Source Bitstream Vera) with the others as the mono and sans font families (shown

in figure 4). I'm not entirely convinced of the pairing (inspired by a more successful local experiment with Adobe Myriad), but I hope others find it instructive.

```
\definetypeface[cheu][rm][serif] [charter][default][encoding=texnansi]
\definetypeface[cheu][ss][sans]  [bera]   [default][encoding=texnansi,
                                                    rscale=.89]
\definetypeface[cheu][tt][mono]  [bera]   [default][encoding=texnansi,
                                                    rscale=.89]
\definetypeface[cheu][mm][math]  [euler]  [euler]  [rscale=1.05]
```

Mixed bold math is also available with Euler. General usage is as described in the ConTEXt magazine 'This Way #5'[4], and you simply need to add this to the above typescript to enable mixed math:

```
\definetypeface[cheu][mm][bfmath][euler]  [euler]  [rscale=1.05]
\setupformulas [method=bold]
```

Footnotes

1. Available from http://www.ctan.org/tex-archive/fonts/psfonts/polish/cc-pl/
2. Antykwa Toruńska's TEX package can be downloaded from http://www.janusz.nowacki.strefa.pl/torunska-e.html. Only the font directory needs to be installed for ConTEXt purposes.
3. TeXfont should be used to install Bera: http://www.ctan.org/tex-archive/fonts/bera/.
4. Download from http://pragma-ade.com/general/magazines/mag-0005.pdf

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

math characters – eul

α	0	alpha	K	0	Kappa	$\spadesuit$	2	spadesuit	$\mp$	2	mp
β	0	beta	Λ	0	Lambda	$\coprod$	3	coprod	$\pm$	2	pm
γ	0	gamma	M	0	Mu	$\bigvee$	3	bigvee	$\circ$	2	circ
δ	0	delta	N	0	Nu	$\bigwedge$	3	bigwedge	$\bigcirc$	2	bigcirc
ϵ	0	epsilon	Ξ	0	Xi	$\biguplus$	3	biguplus	$\setminus$	2	setminus
ζ	0	zeta	O	0	Omicron	$\bigcap$	3	bigcap	$\cdot$	2	cdot
η	0	eta	Π	0	Pi	$\bigcup$	3	bigcup	$*$	2	ast
θ	0	theta	R	0	Rho	$\int$	3	intop	$\times$	2	times
ι	0	iota	Σ	0	Sigma	$\prod$	3	prod	$\star$	1	star
κ	0	kappa	T	0	Tau	$\sum$	3	sum	$\propto$	2	propto
λ	0	lambda	Υ	0	Upsilon	$\bigotimes$	3	bigotimes	$\sqsubseteq$	2	sqsubseteq
μ	0	mu	Φ	0	Phi	$\bigoplus$	3	bigoplus	$\sqsupseteq$	2	sqsupseteq
ν	0	nu	X	0	Chi	$\bigodot$	3	bigodot	$\parallel$	2	parallel
ξ	0	xi	Ψ	0	Psi	$\oint$	3	ointop	$\mid$	2	mid
o	0	omicron	Ω	0	Omega	$\bigsqcup$	3	bigsqcup	$\dashv$	2	dashv
π	0	pi	$\aleph$	2	aleph	$\int$	2	smallint	$\vdash$	2	vdash
ρ	0	rho	$\imath$	1	imath	$\triangleleft$	1	triangleleft	$\nearrow$	2	nearrow
σ	0	sigma	$\jmath$	1	jmath	$\triangleright$	1	triangleright	$\searrow$	2	searrow
τ	0	tau	ℓ	1	ell	$\bigtriangleup$	2	bigtriangleup	$\nwarrow$	2	nwarrow
υ	0	upsilon	$\wp$	1	wp	$\bigtriangledown$	2	bigtriangledown	$\swarrow$	2	swarrow
ϕ	0	phi	$\Re$	2	Re	$\wedge$	2	wedge	$\Leftrightarrow$	2	Leftrightarrow
χ	0	chi	$\Im$	2	Im	$\vee$	2	vee	$\Leftarrow$	2	Leftarrow
ψ	0	psi	∂	1	partial	$\cap$	2	cap	$\Rightarrow$	2	Rightarrow
ω	0	omega	∞	2	infty	$\cup$	2	cup	$\leqslant$	2	leq
ε	0	varepsilon	$\prime$	2	prime	$\ddagger$	2	ddagger	$\geqslant$	2	geq
ϑ	0	vartheta	$\emptyset$	2	emptyset	$\dagger$	2	dagger	$\succ$	2	succ
ϖ	0	varpi	∇	2	nabla	$\sqcap$	2	sqcap	$\prec$	2	prec
ϱ	0	varrho	$\top$	2	top	$\sqcup$	2	sqcup	$\approx$	2	approx
ς	0	varsigma	$\bot$	2	bot	$\uplus$	2	uplus	$\succeq$	2	succeq
φ	0	varphi	$\triangle$	2	triangle	$\amalg$	2	amalg	$\preceq$	2	preceq
A	0	Alpha	$\forall$	2	forall	$\diamond$	2	diamond	$\supset$	2	supset
B	0	Beta	$\exists$	2	exists	$\bullet$	2	bullet	$\subset$	2	subset
Γ	0	Gamma	$\neg$	2	neg	$\wr$	2	wr	$\supseteq$	2	supseteq
Δ	0	Delta	$\flat$	1	flat	$\div$	2	div	$\subseteq$	2	subseteq
E	0	Epsilon	$\natural$	1	natural	$\odot$	2	odot	$\in$	2	in
Z	0	Zeta	$\sharp$	1	sharp	$\oslash$	2	oslash	$\ni$	2	ni
H	0	Eta	$\clubsuit$	2	clubsuit	$\otimes$	2	otimes	$\gg$	2	gg
Θ	0	Theta	$\diamondsuit$	2	diamondsuit	$\ominus$	2	ominus	$\ll$	2	ll
I	0	Iota	$\heartsuit$	2	heartsuit	$\oplus$	2	oplus	$/$	2	not

Figure 5. Euler character list

↔	2 leftrightarrow	|	2 vert	⊠	C boxtimes	≳	C gtrsim
←	2 leftarrow	↑	2 uparrow	□	C square	≳	C gtrapprox
→	2 rightarrow	↓	2 downarrow	□	C Box	⊸	C multimap
↦	2 mapstochar	↕	2 updownarrow	■	C blacksquare	∴	C therefore
∼	2 sim	⇑	2 Uparrow	▪	C centerdot	∵	C because
≃	2 simeq	⇓	2 Downarrow	◇	C Diamond	≒	C doteqdot
⊥	2 perp	⇕	2 Updownarrow	◊	C lozenge	≒	C Doteq
≡	2 equiv	\	2 backslash	⧫	C blacklozenge	≜	C triangleq
≍	2 asymp	⟩	2 rangle	↻	C circlearrowright	≺	C precsim
⌣	1 smile	⟨	2 langle	↺	C circlearrowleft	≲	C lesssim
⌢	1 frown	}	2 rbrace	⇌	C rightleftharpoons	≲	C lessapprox
↼	1 leftharpoonup	{	2 lbrace	⇋	C leftrightharpoons	⩽	C eqslantless
↽	1 leftharpoondown	⌉	2 rceil	⊟	C boxminus	⩾	C eqslantgtr
⇀	1 rightharpoonup	⌈	2 lceil	⊩	C Vdash	≺	C curlyeqprec
⇁	1 rightharpoondown	⌋	2 rfloor	⊪	C Vvdash	≻	C curlyeqsucc
↩	1 lhook	⌊	2 lfloor	⊨	C vDash	≼	C preccurlyeq
↪	1 rhook	√	2 sqrt	↠	C twoheadrightarrow	≦	C leqq
.	1 ldotp	†	2 dag	↞	C twoheadleftarrow	⩽	C leqslant
·	2 cdotp	‡	2 ddag	⇇	C leftleftarrows	≶	C lessgtr
:	0 colon	§	2 S	⇉	C rightrightarrows	`	C backprime
	0 acute	¶	2 P	⇈	C upuparrows	-	C dabar@
	0 grave	◯	2 Orb	⇊	C downdownarrows	≒	C risingdotseq
	0 ddot	.	1 mathperiod	↾	C upharpoonright	≒	C fallingdotseq
	0 tilde	.	1 textperiod	↾	C restriction	≻	C succcurlyeq
	0 bar	,	1 mathcomma	⇂	C downharpoonright	≧	C geqq
	0 breve	,	1 textcomma	↿	C upharpoonleft	⩾	C geqslant
	0 check	Γ	0 varGamma	⇃	C downharpoonleft	≷	C gtrless
^	0 hat	Δ	0 varDelta	↣	C rightarrowtail	⊏	C sqsubset
→	1 vec	Θ	0 varTheta	↢	C leftarrowtail	⊐	C sqsupset
˙	0 dot	Λ	0 varLambda	⇆	C leftrightarrows	▷	C vartriangleright
~	3 widetilde	Ξ	0 varXi	⇄	C rightleftarrows	▷	C rhd
^	3 widehat	Π	0 varPi	↰	C Lsh	◁	C lhd
⏜	3 lmoustache	Σ	0 varSigma	↱	C Rsh	◁	C vartriangleleft
⏝	3 rmoustache	Υ	0 varUpsilon	⇝	C rightsquigarrow	⊵	C trianglerighteq
(	0 lgroup	Φ	0 varPhi	⇝	C leadsto	⊵	C unrhd
)	0 rgroup	Ψ	0 varPsi	↭	C leftrightsquigarrow	⊴	C trianglelefteq
|	2 arrowvert	Ω	0 varOmega	↫	C looparrowleft	⊴	C unlhd
‖	2 Arrowvert	↘	2 internalAnd	↬	C looparrowright	★	C bigstar
⏐	3 bracevert	⊡	C boxdot	≗	C circeq	≬	C between
‖	2 Vert	⊞	C boxplus	≻	C succsim	▼	C blacktriangledow

Figure 5. Euler character list – continued

Figure 5. Euler character list – continued

Lettrines for ConTEXt

Keywords
lettrines, module, initials, dropped capitals, ConTEXt

Abstract
The ConTEXt module `lettri` is port of the LaTEX package `lettrine` by Daniel Flipo that provides a way to typeset dropped capitals at the beginning of paragraphs.

Introduction

Daniel Flipo's LaTEX package "lettrine.sty" provides the command `\lettrine` for the creation of dropped capitals at the beginning of a paragraph. Various parameters are provided to control the size and layout of the dropped capital, using a key–value system to specify the options.

Last februari, Gerben Wierda asked on the ConTEXt mailing list if "Would someone be able to take lettrine.sty as an example and produce a version that works with ConTEXt (and plain TEX)?". I never considered making a version for plain TEX, but a ConTEXt version was doable. So I've created the 'lettri' module, for use in a `\usemodule` command.

Commands

The module defines two user–level commands, one for setup and one for actual use. Most of the parameter names are a bit different from their LaTEX counterparts. There are two reasons for this, both a side–effect of the implementation in ConTEXt.

- The first reason is my laziness, I did not want to create lots of new constants for internationalization of the interface, so I just used an initial uppercase character. This makes the keywords impervious to differences in the ConTEXt language interfaces.
- The second reason is that some parameter names seemed a bit odd, probably because of namespace conflicts within LaTEX, and I sanitized those names where that was possible without confusing the users.

So, for example, the LaTEX parameter keyword `lhang` became the ConTEXt parameter `Hang`.

Usage command: `\lettrine`
The command `\lettrine` uses an optional parameter for settings, and two required arguments that are texts to be typeset.

```
\lettrine [..,.1=.,..] {.2.} {.3.}
          OPTIONEEL
1   inherits from \setuplettrine
2   TEXT
3   TEXT
```

THE TWO typeset arguments are the dropped capital and the run–in text following it; the TEX source of this paragraph started with "`\lettrine{T}{he two} typeset`". The optional parameter is explained below.

Setup command: `\setuplettrine`

```
\setuplettrine [.1.] [..,.2=.,..]
               OPTIONEEL
1   TEXT
2   Lines     =  NUMBER
    Hang      =  TEXT
    Oversize  =  TEXT
    Raise     =  TEXT
    Findent   =  DIMENSION
    Nindent   =  DIMENSION
    Slope     =  DIMENSION
    Ante      =  TEXT
    FontHook  =  COMMAND
    TextFont  =  COMMAND
    Image     =  yes  no
```

- `Lines` controls how many lines the dropped capital will occupy (the default value is 2);
- `Hang` sets how much of the dropped capital's width should hang into the margin (the default is 0, values should be between 0 and 1);
- `Oversize` enlarges or decreases the dropped capital's height: with `Oversize=0.1.` its height is enlarged by 10% so that it raises above the top paragraph's line (default=0, values should be between -1 and 1);

- `Raise` does not affect the dropped capital's height, but moves it up (if positive) or down (if negative); useful with capitals like `J` or `Q` which have a positive depth (default=0, values should be between −1 and 1);
- `Findent` (positive or negative) controls the horizontal gap between the dropped capital and the indented block of text (default=0pt);
- `Nindent` shifts the indented lines, starting from the second line, horizontally by the specified amount (default=0.5em);
- `Slope` can be used with dropped capitals like `A` or `V` to add an extra shift (positive or negative) to the indentation of each line starting from the third one (no effect if `Lines=2`, default=0pt);
- `Ante` can be used to typeset something before the dropped capital (typical use is for French guillemets starting the paragraph).
- `Image` will force `\lettrine` to replace the letter normally used as dropped capital by an image. `\lettrine[Image=yes]{A}{n exemple}` will load `A.eps` or `A.pdf` instead of letter A.
- `FontHook` can be used to change the font and/or color of the dropped capital (default: empty)
- `TextFont` can be used to change the font and/or color of the run–in text (default: `\sc`)

The first, optional argument to the `\setuplettrine` command allows you to create presets: The settings that follow will apply only if the first text argument of `\lettrine` (see below) matches this string exactly. I have used this command at the top of this article:

```
\setuplettrine[T][Findent=0.2em,Nindent=0.2em,
                  Oversize=.05,Hang=.15]]
```

because otherwise the example on the previous page would not have been as nice as it is.

Examples

The following examples were all adapted from the file `demo.tex` that is part of Daniel Flipo's original distribution. I've been forced to make some changes here and there because the font for the Maps is quite different from the font in the original examples, but I have not made changes to the original french text.

Standard options (using 2 lines)

```
\lettrine{E}{n} plein marais...
```

EN plein marais de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges.

Lettrine on a single line

```
\lettrine[Lines=1]{E}{n} plein marais...
```

EN plein marais de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière.

Lettrine on a three lines

```
\lettrine[Lines=3]{E}{n} plein marais...
```

EN plein marais de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges.

Lettrine in the margin

```
\lettrine[Hang=1, Nindent=0pt, Lines=3]
         {J}{ustement},...
```

JUSTEMENT, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges. L'abbaye était ravitaillée — dit-on — par un souterrain qui la reliait au château de Treillan.

Lettrine oversised, and partly in the margin

```
\lettrine[Lines=3,Hang=0.2,Oversize=0.25]
         {E}{n} ...
```

EN plein marais de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» . . .

A guillemet in front of the lettrine

```
\lettrine[Ante={<<}]{E}{n} plein marais ...
```

«EN plein marais de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville

fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» ...

The following four lettrines have all been typeset after changing the default settings with the following command:

```
\setuplettrine[Lines=4,FontHook={\color[gray]}]
```

A somewhat smaller and slightly raised lettrine

```
\lettrine[Oversize=-0.15, Raise=0.15]
         {Q} {u'en plein marais} ...
```

QU'EN PLEIN MARAIS de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière, surprend les visiteurs. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges. L'abbaye était ravitaillée par un souterrain qui la reliait au château de Treillan.

The same lettrine, without corrections

```
\lettrine{Q}{u'en plein marais} de ...
```

QU'EN PLEIN MARAIS de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière, surprend les visiteurs. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges. L'abbaye était ravitaillée par un souterrain qui la reliait au château de Treillan.

Using the Slope option for the following lines

```
\lettrine[Slope=0.4em,Findent=-0.5em,
          Nindent=0.4em]
         {\'A}{quelques kilom\'etres}...
```

À QUELQUES KILOMÈTRES au nord d'Aigues-Mortes, se trouve la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges. L'abbaye était ravitaillée — dit-on — par un souterrain qui la reliait au château de Treillan.

Using the Slope option for the opposite effect
Also note the move into the margin

```
\lettrine[Slope=-0.5em,Hang=0.5,Findent=0.2em]
         {V}{oici} \'a...
```

VOICI à quelques kilomètres au nord d'Aigues-Mortes la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges. L'abbaye était ravitaillée — dit-on — par un souterrain qui la reliait au château de Treillan.

Using a different font by using the FontHook

```
\def\myhook
  {\definefontsynonym[LettrineFont][SansBold]}
\lettrine[FontHook={\myhook},
          Hang=.2, Findent=.3em]
         {E}{n} plein marais...
```

EN plein marais de la Souteyranne, à quelques kilomètres au nord d'Aigues-Mortes, se trouve la Tour Carbonnière. Construite au XIII^{e} siècle, elle contrôlait l'unique voie d'accès terrestre de la ville fortifiée, celle qui menait à Psalmody, l'une des «abbayes de sel» dont il ne reste que quelques vestiges. L'abbaye était ravitaillée par un souterrain qui la reliait au château de Treillan.

Use of an image instead of an actual letter

```
\lettrine[Image=yes,Hang=.1, Oversize=.25,
          Findent=0.1em, Raise=-.1]
          {W} {er} reitet ...
```

ER reitet so spät durch Nacht und Wind?
Es ist der Vater mit seinem Kind;
Er hat den Knaben wohl in dem Arm,
Er faßt ihn sicher, er hält ihn warm.

Availability

The module can be downloaded from the new ConTEXt module repository, at http://modules.contextgarden.net.

I have released this module into the public domain.

Taco Hoekwater
taco@elvenkind.com

TEX and Linguistics

Keywords
babel, bigfoot, colortab, devnag, edmac, ednotes, fontenc, latexsym, ledmac, makor, omega, pscyr, psgreek, pstricks, syntax, xetex, grammar, tipa, philology, phonology, lemmata, linguistics, semantics, cyrillic, teubner, unicode, arabic, armenian, chinese, devanagari, greek, hebrew, japanese, korean, russian, sanskrit

Abstract
TEX has long been associated with mathematics and "hard" sciences such as physics. But even during the early days of TEX, linguists were attracted to the system, and today a growing number of them are turning to TEX (LaTEX, ConTEXt). Aside from the general advantages of TEX for producing academic papers, it offers linguists largely intuitive means for dealing with often complex notational issues. In this paper, an abbreviated version of my Practical TEX 2004 talk, I show some notational issues and their solutions in TEX.

Why TEX?

As a linguist and an avid user of TEX, I'm frequently asked why linguists would want to use TEX, as opposed to a word processor, to write their papers. Of course, there are the general reasons why any academic would benefit from TEX, such as easy handling of numbered examples, footnotes that make sense, bibliographic management via BibTEX.

For me, the main reason to use TEX to typeset linguistic papers and books is due to the complex, but often mathematically-inspired, notational systems used in the various subfields of linguistics. In fact, there are some cases[1] where the ability of TEX to format certain constructs aided their adoption by the field.

In this paper, I discuss various aspects of using TEX and LaTEX to typeset linguistics. One thing you will find largely absent here is a discussion of Omega, which should offer great hope for linguists using TEX. Until its development is further along than it currently is, discussion of it only presents users with a utopian taste of what might be. I would dearly love for Omega to advance, but the wait has been painful. Your mileage may of course vary. *Digital Typography Using LaTEX* ([1]) has a good overview of language support for Omega.

I should note that the union of TEX and linguistics goes back quite far. For example, see the articles by Christina Thiele in *TUGboat*, [2], [3]. Donald E. Knuth[2] notes that linguists were among the first outside of mathematics to embrace TEX.

The field of linguistics

Linguistics is a large field that stands at the crossroads of several other fields, but that is united in dealing with the scientific study of language. As you might expect from a large field, linguistics is commonly subdivided into various disciplines, each of which has various notational traditions and goals.

What is important for TEXnicians is that the notational issues presented here fall essentially into either special symbols or special layouts. In order to partition the field into units we can deal with here, let us adopt a fairly traditional division that linguists often use, rather than the coarser special symbols vs. special layouts.

1. philology
2. phonology
3. phonetics
4. syntax
5. semantics
6. "hyphenated"
   a. psycholinguistics
   b. sociolinguistics
   c. biolinguistics
   d. discourse analysis

Fortunately for us, the "hyphenated" subdivisions largely make due with notation from other subfields, so they need not concern us further here.

In the remainder of the paper, I will take up each subfield in turn and discuss notational issues and how they may be solved with TEX. Most of the discussion deals with various LaTEX packages, which reflects the market share of LaTEX. Some of what follows can be done with more or less difficulty in Plain TEX or ConTEXt.[3]

Due to the complexity of syntactic notation and the generality of application of tree structures, I will postpone discussion of Syntax until installment two of this paper.

Philology

Philology was once the general term for linguistic science, but is now more commonly used to refer to textual (rhetoric, poetics, textual criticism) and historical/diachronic linguistics. Most of the notational issues here deal either with different writing systems or with modifications of Latin.

The latter is usually quite straightforward in TEX, such as ā, ē, ī, ō, ū, š, ṣ, obtained by `\=a`, `\=e`, `\=\i`, `\=o`, `\=u`, `\v{s}`, `\d{s}`. For example, Figure 1 is an example of something you might encounter in a paper on Indo-European.[4]

to me (although the suggestion of Kuryłowicz, *Apophonie* 170, that the ablaut *CeHi* : *CHī* is paralleled by a type *CeRi* : *CRī* seems worth considering).

2.2.4. When the laryngeal followed **r*, **l*, **m*, or **n*, we expect the resonants to become *αρ*, *αλ*, *αμ*, *αν*, with the same *α* that appears outside laryngeal environments, and this is what we often get, e.g. *ἔβαλον* 'they threw' < **ɛǧ^{w}lE-*, *ἔκαμον* 'they toiled' < **ɛ́kmA-*. But where the following laryngeal was *O*, we generally get *ορ*, *ολ* : *ἔμολον* 'they went', *ἔθορον* 'they leapt', *ἔτορε* 'pierced', *ἔπορον* 'they granted' to the presents *βλώσκω*, *θρώ(ι)σκω*, *τιτρώσκω* and the perfect *πέπρωται*. An ablaut *ρω* : *αρ* or *λω* : *αλ* seems not to occur in Greek. F. B. J. Kuiper has suggested, *India Antiqua* 199, that the development to *ορ* and *ολ* here was phonetically regular, the laryngeal influencing the vocalization of the resonant. I doubt that this is correct, because I believe that Dor. *πρᾶτος* represents a direct outcome of **pr̥O-tos* (cf. §2.3.2). If preconsonantal *O* failed to color the

Figure 1. What an Indo-Europeanist reads at the breakfast table

Paleography, for example, often uses a dot below a letter to indicate an obscured reading, which is quite easy to do in TEX (via `\d{}`), but in Word requires a special font, and a utility to access the needed characters.[5] A few other symbols require the use of the TIPA package, which we will discuss below in the section on Phonology.

A major undertaking in philology is the production of critical editions. The requirements of line numbers, cross-references, lemmata, layer upon layer of notes, marginalia, et cetera can bring a typesetter to the brink of madness, but for the edmac (Plain TEX), ledmac and ednotes (LaTEX) packages. Unfortunately, none of these packages offers a complete solution, so you will need to select one based on your specific goals and circumstances. Uwe Lück offers a critical overview of these critical edition packages in [4]. Recently, David Kastrup ([5]) has created the bigfoot package, but I have yet to try it, so I cannot offer an opinion.

1 A dhuine gan chéill do mhaisligh an chléir
b is tharcaisnigh naomhscruipt na bhfáige,
c na haitheanta réab 's an t-aifreann thréig
d re taithneamh do chlaonchreideamh Mhártain,
e cá rachair 'od dhíon ar Íosa Nasardha
f nuair chaithfimid cruinn bheith ar mhaoileann Josepha?
g Ní caraid Mac Crae chuim t'anama ' phlé
h ná Calvin bhiais taobh ris an lá sin.

2 Nách damanta an scéal don chreachaire chlaon
b ghlac baiste na cléire 'na pháiste
c 's do glanadh mar ghréin ón bpeaca ró-dhaor
d trí ainibhfios Éva rinn Ádam,
e tuitim arís fé chuing na haicme sin
f tug atharrach brí don scríbhinn bheannaithe,
g d'aistrigh béasa agus reachta na cléire
h 's nách tugann aon ghéilleadh don Phápa?

3 Gach scolaire baoth, ní mholaim a cheird
b 'tá ag obair le géilleadh dá tháille
c don doirbhchoin chlaon dá ngorthar Mac Crae,
d deisceabal straeigh as an gcolláiste.
e Tá adaithe thíos in íochtar ifrinn,
f gan solas gan soilse i dtíorthaibh dorcha,
g tuigsint an léinn, gach cuirpeacht déin
h is Lucifer aosta 'na mháistir.

22 *Teideal*: Dhuinnluinng T, Seághan Mac Domhnaill cct B
1.a dhuinne T 1.a mhaslaidh T, mhaslaig B 1.c raob T 1.d le B 1.e dod B 1.f chaithfamíd T 1.f maoilinn B 1.g phleidh T 1.h bhíos B 1.h leis B 2.a claon B 2.c glannuig T 2.d ainnibhfios T, ainnbhfios B 2.d Éabha B 2.g is B 2.h tuigionn T 3.a sgollaire T 3.a mholluim T 3.b 'tág ccobar T 3.b re B 3.c dorbhchon daor B 3.d straodhaig T 3.e fhadoghthe tsíos T 3.e fadaighthe B 3.f sollus T 3.g cuirripeacht T 3.h Luicifer T, Lúcifer B 3.h mhaighistir T

Figure 2. A critical edition

Typesetting Greek critical editions presents the same problems as above, plus the need for good Greek fonts. Claudio Beccari ([6]) has extended babel to produce a remarkable facsimile of the famous Teubner editions. It still lacks some refinement for producing the critical apparatus, but the package is under active development, and the results thus far are quite pleasing.

But Greek fonts aren't an issue just when doing Greek critical editions. For whatever historical accident, Greek examples in philology are usually typeset in Greek, even while other languages that don't use the Latin alphabet (such as Sanskrit, Russian, Armenian, Tocharian) are transliterated. Fortunately there are several options for getting and entering Greek examples. The Beccari Greek fonts are excellent, and there is also the PSGreek package ([7]), which bundles Greek PostScript fonts and a style file to make accessing them easier by hiding some of the horrors of encoding vectors. The quality of the PS fonts bundled is somewhat uneven, and installing new fonts for use in the same manner is not easy. To do so requires the grkfinst fontinst plugin ([8]) and some time configuring. I wish it were a bit easier, since the PSGreek

interface is one I find quite comfortable to use, and it has proven to be a lifesaver for switching Greek fonts.

The Greek in Figure 1 was produced with PSGreek. For example, to get 'they went', you enter `\textgreek{>'emolon}`. I am now quite used to entering Greek in this manner, and therefore I can do it quite rapidly. However, you may be more comfortable entering Greek in Unicode, given an appropriate text editor. For that, put the following in your preamble:

```
\usepackage{ucs}
\usepackage[utf8]{inputinc}
\usepackage[polutonikogreek,english]
                                    {babel}
```

There are two aspects to typesetting languages in alphabets other than Latin. First, there are times when you need to typeset a single language solely for speakers of that language, such as setting a Russian text in Cyrillic for a Russian reader. On the other hand, at times it is necessary to mix two or more languages, such as in dictionaries or instructional material.

Both scenarios are supported in TEX, although dealing with encoding vectors can cause a headache or two.[6] Since I can't detail all possible language packages, let me limit myself here to a couple of packages I've found to be useful.

Underpinning nearly all multilingual endeavours in LaTEX is babel ([9]) by Johannes Braams. It is included in (I believe) all TEX distros, the manual is comprehensive and well written, and you should spend some time familiarising yourself with it.

For Russian and the other Cyrillic-alphabet languages, there is the default Computer Modern Cyrillic font, which matches the standard Soviet look nicely.[7] At some point, though, you'll no doubt want a change of pace. The pscyr package ([10]) contains a number of serif, sans serif, and a couple of display faces.

Languages that use Indic scripts, such as Devān,agarī, have a complication that not all graphemes occur in the same order as they are pronounced, plus there are many, many di- and trigraphs. The devnag package ([11]) provides a preprocessor to take care of these complexities, plus good fonts and macros for both Plain TEX and LaTEX. Using devnag makes it possible to typeset a bilingual critical edition with essentially the same input for both the Devānagarī and the transliterated text. Figure 3 shows the vowels of Marāṭhī, typeset with the devnag package.

For languages written in the Arabic alphabet (such as Arabic, Persian, Pashto), Klaus Lagally's ArabTEX is a must. The system is by now quite stable, and the output is very good. Several people are working on various extensions, especially for typesetting Arabic mathematics. See for example, Lazrek et al. ([12], [13]).

अ	आ	इ	ई	उ	ऊ
*a*bout	c*a*r	s*i*t	s*ea*t	p*u*t	r*oo*t
ऋ	ऌ	ए	ऐ	ओ	औ
und*er*	bott*le*	s*ay*	b*y*	r*oa*d	l*ou*d
अं	अः				

Figure 3. The vowels of Marāṭhī

While it is possible to typeset Hebrew using ArabTEX, Alan Hoenig's Makor ([14]) is worth every penny.[8]

Typesetting Chinese using TEX is possible with the CJK ([15]) package (which provides for much more than just Chinese, Japanese, and Korean support). However, I prefer ConTEXt, due to its support of visual debugging via `\tracechinesetrue`. Numbering can be toggled between Chinese and western styles via `[conversion=chinese]` or `[conversion=numbers]`. More traditional vertical typesetting is possible essentially by flowing the text into narrow columns.

Semantics

Semantics is the study of meaning, and the notation used is tied closely to formal logic. Thus it is very straightforward to typeset with TEX. So the function of the set of things similar to houses is denoted by $\lambda x \mathit{Similar\_to}(x, \mathit{houses})$. The TEX to get this is `$\lambda x \mathit{Similar\_to}(x, \mathit{houses})$`. We had to wrap the 'English' inside the function with `\mathit` to prevent TEX from interpreting the words as a series of variables. In some cases `\mbox` will work, and note that sometimes spaces inside the `\mbox`es are important. So a possible interpretation for the sentence *I have told one friend of mine all those stories*[9] is given as $\exists x[\forall y[(x \in \text{friends of mine} \wedge y \in \text{those stories}) \rightarrow \text{I have told } y \text{ to } x]]$, or in TEX terms `$\exists x [\forall y[(x \in \mbox{friends of mine} \wedge y \in \mbox{those stories}) \rightarrow \mbox{I have told } y \mbox{ to } x]]$`.

Double brackets (representing semantic evaluation) are provided by the stmaryrd package ([16]). So, typing `$\llbracket(MN)\rrbracket^{\mathcal{M}}$` yields $[\![(MN)]\!]^{\mathcal{M}}$. You may also need to load the latexsym package for a few symbols.

Phonetics and Phonology

Phonetics is a branch of acoustics that deals with speech sounds and their production and perception.

The notation used is a combination of transcription symbols (as covered below) and diagrams representing articulatory spaces. For example, Figure 4 shows a typical representation of a vowel system.[10]

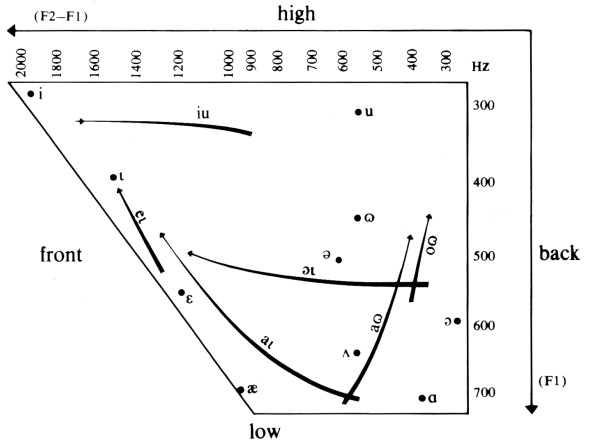


Figure 4. Some American English vowels

Also frequent are diagrams of the human vocal tract. Unfortunately there is no easy way to handle automatic generation of notation of this type. A typical way to handle them is to create the illustration in a vector program (such as Inkscape or Illustrator) and then to import it into TEX. Given the complexity of creating them, publishers (and therefore also authors) have been reluctant to use them aside from in very specialised books.[11]

The International Phonetic Association came into existence in 1886 with a goal of promoting phonetics in education and the creation of an international phonetic alphabet (now known as the IPA) for the universal transcription of languages.[12] A separate tradition of transcription developed among anthropological linguists in America. Both systems of transcription ([18], [19]) are supported via the TIPA package ([17]).

While there are numerous fonts that provide IPA symbols that more or less match existing typefaces, there are to my knowledge still only a small number of type families that have complete sets of corresponding IPA symbols: Computer Modern, Lucida, Times, Le Monde, Gentium, Garamond, and Stone – and two of them are provided for by TIPA. To wit, the Computer Modern IPA symbols work best with Computer Modern,[13] but they will fit in reasonably with other vertical-stress typefaces. The Times IPA symbols, again, work best with Times, but in a pinch, they fit in with other oblique-stress typefaces. In a sans-serif environment, TIPA provides a Helvetica-like symbol set.

In addition to the IPA fonts and the interface (more on which below), TIPA provides a style file to produce simple vowel diagrams (simpler than the one shown in Figure 4). I could conceive – given enough labour – of creating the more complex charts like Figure 4, with arrows and swooshes, programmatically via the tools provided by PSTricks ([20]) or MetaFun ([21]).

TIPA provides for a couple of different ways to enter phonetic notation. There are long forms that have generally mnemonic names, so I can write [əhˈa] as `[\textschwa h\textprimstress a]` if my paper uses a limited set of symbols, and I don't want to learn the more involved transcription.

On the other hand, if you need to input larger amounts of transcription, it is useful to enter the IPA environment via `\textipa{}`, `{\tipaencoding }`, or `\begin{IPA}` and `\end{IPA}`. So, if we enter the IPA environment and type

```
D@ "nO;T "wInd @nd D@ "s2n w@ dIs"pju;tIN
wItS w@z D@ "str6Ng5, wEn @ "tr\ae vl5
keIm @"l6N "r\ae pt In @ "wO:m "kloUk. DeI
@"gri:d D@t D@ "w2n hu; f3;st s@k"si;dId
In "meIkiN D@ "tr\ae vl5  teIk hIz "kloUk
6f SUd bI k@n"sId@d "str6Ng@ D@n DI "2D@.
```

This will be rendered into IPA as follows:[14]

ðə ˈnɔˑθ ˈwɪnd ənd ðə ˈsʌn wə dɪsˈpjuˑtɪŋ wɪtʃ wəz ðə ˈstrɒŋgɐ, wɛn ə ˈtrævlɐ keɪm əˈlɒŋ ˈræpt ɪn ə ˈwɔːm ˈkloʊk. ðeɪ əˈgriːd ðət ðə ˈwʌn huˑ fɜˑst səkˈsiˑdɪd ɪn ˈmeɪkɪŋ ðə ˈtrævlɐ teɪk hɪz ˈkloʊk ɒf ʃʊd bɪ kənˈsɪdəd ˈstrɒŋgə ðən ðɪ ˈʌðə.

TIPA allows you to enter tonal specifications and has many other nice features to explore. I heartily recommend reading the excellent manual included in the package.

As I mentioned earlier in the paper, the TIPA package also allows you to enter Indo-European reconstructed forms. For example, the work for '100' is reconstructed as **k̂mtóm*, which can be entered as `\textroundcap{k}\textsubring{m}t\'om` or as `\textipa{*\|c{k}\r*mt\'om}`.

A new notational twist entered both phonology and syntax within the past decade as Optimality Theory grew in popularity. The central part of its notation are the so-called optimality tableaux. There are a number of ways to enter them, but I've settled on using PSTricks together with colortab ([22]). On the next page is source and output using some totally nonsense data.

```
\begin{tabular}[t]{r|c|c|c|}
\cline{2-4}
 & /ba/  & \textipa{\!@} & b\textturna\\
\LCC
 & & & \lightgray \\ \cline{2-4}
\w   & [ba] & & * \\ \cline{2-4}
 & [*ba] & *! & \\ \cline{2-4}
\ECC
\end{tabular}
```

/ba/	ɓə	bɐ
[ba]		*
[*ba]	*!	

One other subdivision of phonology, computational phonology, uses a mixture of standard phonological notation plus tree structures. As such it will be covered in the second installment of this paper.

Next time

The subfield of linguistics with perhaps the widest variety of notations is syntax. I will postpone until installment two of this paper a discussion of the trees, matrices, and derivations, as I wish to cover them in greater detail than I have space or time at present. In particular, the macros for drawing trees have a far wider application than just for linguistics.[15]

Notes

1. Such as Attribute Value Matrices in Head-driven Phrase Structure Grammar.
2. Personal communication at the Practical TEX banquet.
3. Hans Hagen is aware of many of these issues, and we are working on adding more support for linguistics to ConTEXt.
4. From *The Collected Writings of Warren Cowgill*, Beech Stave Press, 2005.
5. I once tried to explain to my advisor how to get at some of these characters on a Mac. He shook his head and told me, "If I learn how to do that, I'll forget Hittite."
6. For Mac users, the XETEX system frees you from many of these problems. See the website at `http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=xetex`. However, you cannot exchange source files with colleagues who use other operating systems.
7. The Soviets heavily standardised book typefaces at a point when "modern" fonts were popular. There were some fantastic Russian typefaces developed during the 1920s that were neglected for decades.
8. Yes, it is free software, and yes, I am making an exception to not discussing Omega software.
9. From Ray Jackendoff, *Semantic Interpretation in Generative Grammar*, Cambridge: MIT Press, 1972, p. 308.
10. From Peter Ladefoged, *A Course in Phonetics*, San Diego: Harcourt Brace Jovanovich, 1982, p. 198.
11. They can hardly be avoided in an introduction to phonetics, for example.
12. Remember that there were no tape recorders in 1886!
13. And variants such as Latin Modern.
14. *The north wind and the sun were disputing which was the stronger, when a traveller came along wrapped in a warm cloak. They agreed that the one who first succeeded in making the traveller take his cloak off should be considered stronger than the other.* The text comes from the International Phonetic Association.
15. As Nelson Beebe remarked at the PracTEX conference. Installment two will I hope serve as his requested paper.

References

[1] Apostolos Syropoulos, Antonis Tsolomitis, and Nick Sofroniou, *Digital Typography Using LaTEX*, New York: Springer, 2003.
[2] Christina Thiele, "TEX and Linguistics," *TUGboat* **16**, 42–44.
[3] Christina Thiele, "TEX and the Humanities," *TUGboat* **17**, 388–393.
[4] Uwe Lück, "ednotes—critical edition typesetting with LaTEX," *TUGboat* **24**, 224–236.
[5] David Kastrup, "The bigfoot bundle for critical editions," Preprints for the 2004 Annual TUG Meeting, 105–110.
[6] `CTAN/macros/latex/contrib/teubner`
[7] `CTAN/fonts/greek/psgreek`
[8] `CTAN/fonts/utilities/fontinst-contrib/grkfinst`
[9] `http://www.braams.cistron.nl/babel/index.html`
[10] `http://www.opennet.ru/prog/info/1117.shtml`
[11] `http://devnag.sarovar.org`
[12] Mustapha Eddahibi, Azzeddine Lazrek, and Khalid Sami, "Arabic mathematical e-documents," Preprints for the 2004 Annual TUG Meeting, 42–47.
[13] Azzeddine Lazrek, "CurExt, typesetting variable-sized curved symbols," *TUGboat* **24**, XX–XX.
[14] `CTAN/language/hebrew/makor`
[15] `CTAN/language/chinese/CJK`
[16] `CTAN/fonts/stmaryrd`
[17] `CTAN/fonts/tipa`

[18] Geoff Pullum and William Ladusaw, *Phonetic Symbol Guide*, Chicago: University of Chicago Press, 1996.
[19] *Handbook of the International Phonetic Association*, Cambridge: Cambridge University Press, 1999.
[20] `http://www.tug.org/applications/PSTricks`
[21] `http://contextgarden.net/MetaFun`
[22] `CTAN/macros/generic/colortab`

Steve Peter
Beech Stave Press
310 Hana Road
Edison NJ 08817, USA
`speter@beechstave.com`

Controlling Acrobat Reader under X11

Keywords
pdf, acrobat reader, X window system, scripting, pdfopen

Abstract
The command–line programs pdfopen and pdfclose allow you to control the X Window System version of Adobe's Acrobat Reader from the command line or from within a script.

Most people who use the Acrobat Reader to preview PDF files generated from TEX documents will know that it is a hassle to deal with documents that need to be compiled while being viewed.

The Linux version of Adobe's program simply does not notice that the PDF file has changed, and the Microsoft Windows version is even worse: it opens the PDF file using mandatory locking, making it absolutely impossible to recompile the document while it is still open in the Reader.

Common practice using Acrobat Reader for viewing PDF's generated from pdfTEX is therefore to cycle through these actions, either from the command line or from a script:

1. edit the TEX source
2. compile to PDF
3. view with Acrobat Reader
4. close Acrobat Reader

It follows that the Reader has to make a complete restart for each cycle, which is a slow operation. The alternative would be to run the Reader in the background and ask users to, after steps 1. and 2., manually close and reopen the document. That is not user-friendly, of course, and that's where `pdfopen` and `pdfclose`, described in this article, will be useful.

Because the problem was much more severe under Windows, a few years ago Fabrice Popineau has written two small programs that use DDE calls to control the Reader from an external script or batchfile:

- `pdfclose` to make the Reader close the file before the compilation starts
- `pdfopen` to re-open the file afterwards.

The Linux X11 versions are command–line compatible with Fabrice's originals, but they do not function completely identically.

`pdfclose --file <pdf file>`
This will close an X window with the name `<pdf file>` (for Acrobat Reader 5) or the name `Adobe Reader - <pdf file>` (for Adobe Reader 7).

`pdfclose --all`
The Linux `pdfclose` command ignores the `--all` command–line switch. The Windows version will close *only* the files that were opened through `pdfopen` when `--all` is given, and this cannot easily be done under X. Ignoring the options seems wiser than unconditionally closing all open PDF documents.

`pdfopen`
This command–line sends a "go to previous document" to an already existing, but empty, Adobe Reader window. There are perhaps some situations where this possibility might come in handy.

`pdfopen --file <pdf file> [--page <pagenumber>]`
The Linux version silently ignores a given `--page` option, because its behaviour would be near–impossible to predict. The program also reacts a bit differently to the `--file` option: if the file is already open in the Reader, it will close and re–open the document.

Normally, this is the command you want to use under Linux, because it will immediately re–open the PDF file you have given as an argument in the Reader, using the same page & view settings.

I've tested my programs with Acrobat Reader 5.0.10 under Mandrake Linux 10.1 using X.org 6.8.2, but the code is reasonably generic and should work out of the box using most X11 implementations.

It will work using the new Adobe Reader 7.0 under Linux as well, assuming you keep your PDF files maximized within the main Adobe Reader window. The PDF document's name has to appear in the window title for the progams to work. Also, you probably want to set the preference *Reopen documents to last viewed page* to "All files". You can find this setting in the *Startup* page of the preferences screen.

Source and binaries of the programs can be downloaded from : http://tex.aanhet.net/pdfopen

Taco Hoekwater
taco@aanhet.net

Met XML van database naar LaTeX

Het automatisch publiceren van database informatie in LaTeX documenten

Keywords
conversie, transformatie, XML, XSLT, XPath, Saxon, database, Access, publiceren.

Abstract
In dit artikel wordt de geautomatiseerde aanmaak van het programmaboek voor de Bacheloropleiding Technische Wetenschappen bij de Hogere Defensie Opleidingen (HDO) toegelicht. Een belangrijk aspect hierbij vormt de omzetting van een Microsoft Access database naar LaTeX tabellen en bijlages in de uiteindelijke documentatie: XML-bestanden worden als tussenstap gehanteerd.

Inleiding en aanleiding

De Hogere Defensie Opleidingen (HDO, de samenvoeging van de Koninklijke Militaire Academie in Breda, het Koninklijk Instituut voor de Marine in Den Helder en het Instituut Defensie Leergangen in Rijswijk) zijn begonnen met de introductie van nieuwe opleidingsprogramma's volgens de Bachelor-Master structuur. In het kader van die transformatie is ook een opleiding Technische Wetenschappen (TW) ontwikkeld; een opleidingsteam heeft een ontwerp en de documentatie gegenereerd.

Het ontwerpproces kent een aantal uitgangspunten en karakteristieken:

□ Aangezien het team veelal geografisch gescheiden en in een uiteenlopende IT-omgeving werkt is bij dit proces zoveel mogelijk gebruik gemaakt van IT-hulpmiddelen. Zo vond de interne informatievoorziening via een Website plaats, waar met name een archief van alle tussenproducten ingericht was.

□ De standaard software in de defensieomgeving is Microsoft Office. Die draait op alle PC's en veel gebruikers zijn ermee vertrouwd. Daarom moesten Word brondocumenten en zoveel mogelijk Office applicaties gehanteerd worden in de ontwikkeling;

□ Desalniettemin is in een vroeg stadium door beide auteurs nagedacht over eenduidigheid en consistentie in de brongegevens, het werken in teamverband en beheersbaarheid van de gegevens op lange termijn. Dan kom je al gauw uit op het gebruik van een centrale database.

□ Daarbij stond het streven naar *Open Standards* voorop en dat bracht als oplossingsrichting al snel het gebruik van XML en LaTeX in beeld[1]. Vooral het artikel van Berend de Boer [1] gaf de doorslag om met XML en ConTeXt/LaTeX verder te gaan.

□ Gezocht is naar een zoveel mogelijk geautomatiseerd proces bij de aanmaak van documentatie, waaronder het programmaboek voor de bachelor Technische Wetenschappen (TW). Teksthoofdstukken in het programmaboek TW veranderen qua inhoud zeer beperkt. Daartoe is brontekst direct in LaTeX ingevoerd of heeft een eenmalige conversie met *Word2LaTeX* plaatsgevonden. Gedetailleerde curriculumgegevens veranderen frequent en zijn daarom als tabellen en bijlages in het boek opgenomen.

□ Belangrijk is daarbij een keuze voor het conversieproces en de integratie van de gegevens in de documentatie. Berend de Boer noemde in [1] de mogelijkheid om XML data direct om te zetten naar ConTeXt code, zonder gebruik te maken van de door hem beschreven ConTeXt macro's. Deze directe omzetting, en dan naar LaTeX, is voor het invoegen van curriculumgegevens in het programmaboek TW gebruikt, enerzijds omdat deze flexibeler gevonden werd, anderzijds omdat ConTeXt binnen het Koninklijk Instituut voor de Marine (KIM) niet gebruikt wordt.

De geautomatiseerde aanmaak van de documentatie is het onderwerp van dit artikel.

Wat is XML

Er is momenteel sprake van een XML-hype. Er kan geen programma meer uitkomen zonder dat de fabrikant uitgebreid verklaart dat het programma XML ondersteunt.

De afkorting XML staat voor *Extensible Markup Language*. Een bekende markup taal is HyperText Markup

Language (HTML). Het kenmerk van een *markup taal* is dat gegevens, weergegeven als reeksen van karakters, op een of andere manier gemarkeerd zijn, zodat de *betekenis* van die gegevens duidelijk wordt. Pas als de betekenis duidelijk is kan een computerprogramma of de mens er iets mee doen. De gegevens worden dan *informatie*. LaTeX is ook een markup taal, immers in LaTeX worden stukken tekst (secties, paragrafen, lijsten, titel etc.) gemarkeerd zodat een typesettingprogramma deze op de juiste manier kan weergeven.

Het verschil tussen HTML en LaTeX enerzijds en XML anderzijds is dat XML *extensible* is. HTML en LaTeX hebben een min of meer vaste set van *markup codes*, waarmee het gebruik van de informatie beperkt is tot één toepassing, te weten presentatie in een browser respectievelijk printen (typesetting) van de gegevens. XML daarentegen schrijft slechts de *syntax* voor van de markup codes en laat de betekenis ervan volledig vrij. Om iets met XML te kunnen doen, moeten eerst afspraken over de betekenis gemaakt zijn. De toegestane markup codes en hun onderlinge relaties en beperkingen worden in een *Document Type Definition (DTD)* vastgelegd, die ingevoegd wordt in het XML-bestand of waarnaar verwezen wordt. Er bestaan gestandaardiseerde DTD's, bijvoorbeeld XHTML (HTML versie die aan de XML standaarden voldoet), MathML (wiskunde) en XFRML (accountants).

Om de hype te relativeren, eerst een opsomming wat XML *niet* is (uit [2]):

□ XML is geen programmeertaal. XML kan niet uitgevoerd worden en er bestaan geen XML compilers;

□ XML is geen netwerkprotocol. XML-bestanden kunnen net als ieder ander bestand over een netwerk getransporteerd worden, waarbij gebruik gemaakt wordt van protocollen als HTTP, FTP, NFS;

□ XML is geen database. XML is een representatie van informatie en kan daarmee wel gebruikt worden om informatie tussen databases uit te wisselen;

□ XML is niet zaligmakend. Hoewel vele databases en teksverwerkers XML kunnen importeren en exporteren, maken ze meestal gebruik van eigen DTD's, zodat ze elkaars XML-bestanden jammer genoeg niet accepteren!

Om te laten zien wat XML *wel* is volgt hier een fragment uit een XML-bestand:

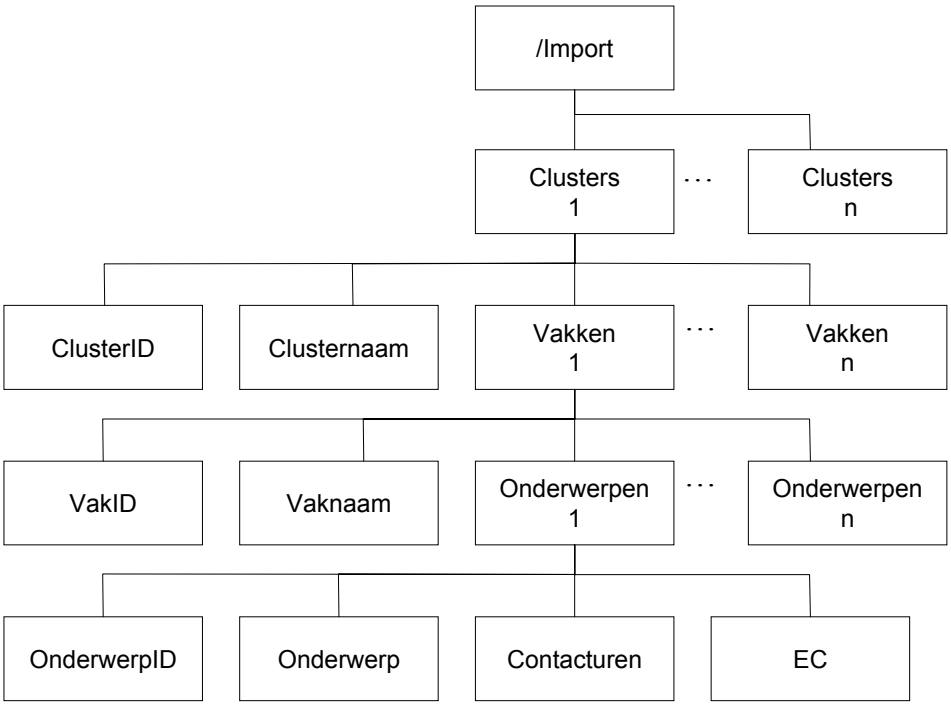
```
<?xml version="1.0" encoding="UTF-8"?>
<Import>
  <Clusters>
    <ClusterID>11</ClusterID>
    <Clusternaam>Wiskunde</Clusternaam>
    <Vakken>
      <VakID>25</VakID>
      <Vaknaam>Analyse 1</Vaknaam>
      <Vaksoort>Grondslag</Vaksoort>
      <Datum>19-11-2004</Datum>
      <Onderwerpen>
        <OnderwerpID>21</OnderwerpID>
        <Onderwerp>Integralen</Onderwerp>
        <Werkvorm>Werkcollege</Werkvorm>
        <Contacturen>8</Contacturen>
        <ECTS>0,6</ECTS>
      </Onderwerpen>
      <Onderwerpen>
        <OnderwerpID>22</OnderwerpID>
        .....
      </Onderwerpen>
    </Vakken>
    <Vakken>
      <VakID>27</VakID>
      <Vaknaam>Statistiek</Vaknaam>
      .....
    </Vakken>
  </Clusters>
  <Clusters>
    <ClusterID>10</ClusterID>
    .....
</Import>
```

In bovenstaande is te zien dat gegevens in XML gemarkeerd worden met zogenaamde *tags*: `<tag>Te markeren informatie</tag>`. Zo'n samenstel van tags en gegevens heet een *element*. Elementen representeren dus een stuk informatie. Markeren gaat met een begin `<tag>` en eind `</tag>`. Elementen kunnen ook leeg zijn: `<tag/>` (let op de plaats van de "/"). Elementen kunnen *genest* worden d.w.z. de gegevens kunnen zelf ook weer elementen bevatten. In dat geval is er sprake van een *gelaagde informatieboom*. Zie Figuur 1 voor een grafische representatie van bovenstaand XML-bestand.

Conversie van XML naar tekst

Op zich is XML al nuttig om informatie in een bestand vast te leggen. Pas echt krachtig wordt XML door de diverse gerelateerde standaarden en definities waarmee op een standaard manier met XML omgegaan kan worden. En, zoals het een echte *open standaard* betaamt, zijn deze standaarden gepubliceerd (`www.w3c.org`) en zijn er veel (gratis) programma's en bibliotheken te krijgen voor XML. De volgende standaarden zijn relevant voor dit artikel (er zijn er echter veel meer, zie [3] voor een compleet overzicht):

□ *XPath*: een krachtige set van commando's waarmee in een XML-bestand elementen *geselecteerd* kunnen worden;



Figuur 1. De gelaagde informatie in het XML voorbeeld.

□ *XSLT: Extensible Stylesheet Language Transformations*, een taal waarmee na selectie van elementen *iets met de informatie gedaan* kan worden. Elegant is dat XSLT-bestanden zelf qua formaat voldoen aan de XML standaard. XSLT maakt gebruik van XPath.

Zoals Berend de Boer in [1] al vermeldde, kan XSLT een XML-bestand transformeren naar:

□ een ander XML-bestand. Dan wordt de informatie in een nieuwe elementenstructuur (met een andere DTD) vertaald;

□ een HTML-bestand. Logisch, want HTML lijkt erg veel op XML;

□ een tekstbestand. En dat kan dus een LaTeX-bestand worden, immers in LaTeX worden alle formateringscommando's in tekst gecodeerd;

□ formatted output (zgn. XSL-FO), bijvoorbeeld naar PDF of naar TeX.

In dit artikel zullen we de XSLT transformatie van XML naar een *tekstbestand* demonstreren. Het doel zal zijn het eerder gegeven XML-bestand om te zetten naar onderstaande LaTeX code. Deze is na typesetting te herkennen als Figuur 2.

Hoofdstuk 1: Cluster- en vakbeschrijvingen

1.1 Wiskunde

Vakken	CU	EC
⇒Analyse 1	64	5
⇒Statistiek	45	4
...		

1.2 ...

Figuur 2. Resultaat van de conversie.

```
\chapter{Cluster- en vakbeschrijvingen}
\label{chap:archdetail}

\section{Wiskunde}
\label{sec:11}

\begin{tabular}{@{}l@{}lrr}\hline
     & Vakken & CU & EC \\\hline
\htmlref{$\Rightarrow$}{vak:25}&
                          Analyse 1 & 64 & 5\\
\htmlref{$\Rightarrow$}{vak:27}&
                          Statistiek & 45 & 4\\
```

```
%volgende vakken in wiskunde cluster
\hline
\end{tabular}
\newline

\section{....
%volgend cluster

%etc...
```

Hierin is te zien dat de cluster- en vakinformatie, afkomstig uit het XML-bestand, ingevoegd is tussen LaTeX commando's. Om dit te demonstreren het begin van het XSLT-bestand dat deze conversie verzorgt:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
     "http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"
                omit-xml-declaration="yes"/>
<!-- Einde header, begin XSLT code -->
<xsl:template match="/import|Import">
  <xsl:text>
\chapter{Cluster- en vakbeschrijvingen}
\label{chap:archdetail}
  </xsl:text>
```

De eerste regel is de identificatie van de gebruikte XML standaard. Ieder XML-bestand moet met een dergelijke regel beginnen. Te zien is dat XSLT gecodeerd is als XML. De tweede regel verwijst naar de DTD van XSLT, deze verwijzing is verplicht in XSLT. De derde regel geeft aan dat er alleen tekst geschreven mag worden. Commentaar in een XML-bestand staat tussen `<!--` en `-->`.

Bij het uitvoeren van het XSLT-bestand (*hoe* dat uitgevoerd wordt zien we verderop) wordt door het XML-bestand heen gelopen en wordt telkens gekeken welk "template" van toepassing is. Templates worden gedefinieerd met het element `<xsl:template match=...>`, dat als gegevens de XSLT commando's bevat die bij een "match" toegepast moeten worden. In ons geval bevat het XSLT-bestand slechts één `template`, te weten `<xsl:template match="/import|Import">`. Daarmee wordt het element `/import` *of* `/Import`[2] uit het XML-bestand gematched. De gegevens in het element `Import` worden daarna als *huidig niveau* beschouwd voor de verdere verwerking. De "/" staat voor de *root*, deze is met "|" (*of-operator*) afkomstig uit XPath. De XPath commando's staan in XSLT tussen dubbele aanhalingstekens.

Het `<xsl:text>Informatie</xsl:text>` commando schrijft de `Informatie` direct naar de output, met behoud van lege regels, spaties etc. In bovenstaand voorbeeld worden daarmee de LaTeX commando's geschreven.

We gaan nu verder met het invoegen van de gewenste informatie:

```
  <xsl:for-each select="Clusters">
    <xsl:text>

\section{</xsl:text>
    <xsl:value-of select="Clusternaam"/>
    <xsl:text>}
\label{sec:</xsl:text>
    <xsl:value-of select="ClusterID"/>
    <xsl:text>}

\begin{tabular}{@{}l@{}lrr}\hline
&amp; Vakken &amp; EC &amp; CU \\\hline
    </xsl:text>
```

Met `<xsl:for-each select="Clusters">` worden alle cluster-elementen om de beurt geselecteerd en als *huidig niveau* beschouwd. Per cluster wordt eerst een lege regel gevolgd door `\section{` geschreven. Daarna moet de clusternaam ingevoegd worden: dat gebeurt met `<xsl:value-of select="Clusternaam"/>`. Deze zoekt het actuele element `Clusternaam` en copieert de bijbehorende informatie (`Wiskunde`) naar de uitvoer. Merk op dat we inmiddels twee niveau's diep in het XML-bestand zitten (/Import/Clusters) in de eerste `Cluster` en dat `Clusternaam` dus *eenduidig bepaald* is.

Hierna wordt het `\section` commando afgesloten, waarna `\label{sec:11}` (immers het *huidige* `ClusterID` is 11) geschreven wordt. Belangrijk is dat het XSLT-bestand de volgorde van de informatie in de uitvoer bepaalt, niet het XML bronbestand. XPath commando's in XSLT selecteren de gewenste elementen. Als je bijvoorbeeld de naam van het vijfde cluster wilt hebben, en je zit op het niveau `<Import>`, dan kan dat met

```
<xsl:value-of select="Clusters[5]/Clusternaam"/>
```

Ook selecties zijn mogelijk:

```
<xsl:value-of select="Clusters
          [Clusternaam='Wiskunde']/ClusterID"/>
```

geeft het `ClusterID` van het cluster met `Clusternaam = 'Wiskunde'`.

Na het schrijven van het `ClusterID` wordt de LaTeX tabeldefinitie geschreven. Merk daarbij op dat de "&" in XML een speciaal teken is en daarom altijd als "`&`" (voor ampersand) geschreven moet worden. Het hebben van verschillende speciale tekens en afwijkende tekensets in XML en LaTeX compliceert de conversie erg!

Na deze basis volgt de rest van de XSLT code:

```
    <!--List of Vakken within Cluster-->
    <xsl:for-each select="Vakken">
      <xsl:sort select="Vaknaam"/>
```

```
      <!-- create a reference -->
      <xsl:text>
\htmlref{$\Rightarrow$}{vak:</xsl:text>
      <xsl:value-of select="VakID"/>
      <xsl:text>} &amp; </xsl:text>
      <xsl:value-of select="Vaknaam">
      <xsl:text> &amp; </xsl:text>
      <!-- compute total Contacturen -->
      <xsl:value-of select=
              "sum(Onderwerpen/Contacturen)"/>
      <xsl:text> &amp; </xsl:text>
      <!-- compute total EC -->
      <xsl:value-of select=
     "round(10*sum(Onderwerpen/ECTS)) div 10"/>
      <xsl:text>\\</xsl:text>
    </xsl:for-each> <!--Vakken-->
    <xsl:text>
\hline
\end{tabular}
\newline
    </xsl:text>
  </xsl:for-each> <!--Clusters-->
</xsl:template> <!--template end-->
</xsl:stylesheet>
```

We zien hier de volgende relevante zaken:

□ Het `<xsl:sort>` commando sorteert de uitvoer, in dit geval de vakken, op vaknaam.

□ Met `\htmlref{$\Rightarrow$}{vak:xx}` (uit het HTML package, onderdeel van LaTeX2HTML) wordt een hyperlink gemaakt. Als de ID's in het XML-bestand *uniek* zijn, geeft dat correcte links door het hele document.

□ Het `<xsl:value-of select="sum(Onderwerpen/Contacturen)"/>` commando sommeert de getallen die in het *huidige vak* per onderwerp (dat kunnen er meerdere zijn) in het element `Contacturen` gegeven zijn. `sum()` Is een XPath functie.

□ XSLT gaat niet goed om met de sommatie van *niet-gehele* getallen. `<xsl:value-of select="round(10*sum(Onderwerpen/ECTS)) div 10"/>` sommeert de EC's (European Credits, maat voor studielast) per vak en doet een afronding. Ook `round()` en `div` komen uit XPath.

Tot zover de behandeling van de conversie van XML naar LaTeX met XSLT/XPath. Realiseer je dat hier slechts een fractie behandeld is van de mogelijkheden die XSLT en XPath bieden! Zie [3] voor een gedetailleerd overzicht van alle mogelijkheden. We gaan nu kijken hoe het gehele documentatiesysteem gerealiseerd is.

Het document systeem

In dit deel wordt behandeld hoe de hiervoor beschreven conversie gebruikt wordt om informatie vanuit een master-database via LaTeX te publiceren. Figuur 3 geeft een overzicht van de diverse stappen met de tussenformaten.

Stap 1: De master-database

Als relationeel database-programma is *Microsoft (MS) Access* gebruikt. Dit heeft de volgende voordelen:

□ Een *relationele database* garandeert consistentie van de informatie, omdat een gegeven slechts eenmaal in een *tabel* opgeslagen wordt, met een *unieke identificatie*. Informatie in verschillende tabellen kan vervolgens onderling *gerelateerd* worden. De tabellen met de onderlinge relaties kunnen worden weergegeven in een zgn. *Entity-Relation Diagram (ERD)* of *Informatie model*. Figuur 4 geeft het vereenvoudigde informatiemodel van de gebruikte master-database. Daarin is de informatie te herkennen die we ook in het XML-bestand gezien hebben, zoals clusters, vakken en onderwerpen.

□ MS Access is het standaard database-programma binnen Defensie en is een redelijk eenvoudig te leren. Het heeft goede invoer- en uitvoermogelijkheden die gebruikt zijn bij de vulling van de master-database door het opleidingsteam.

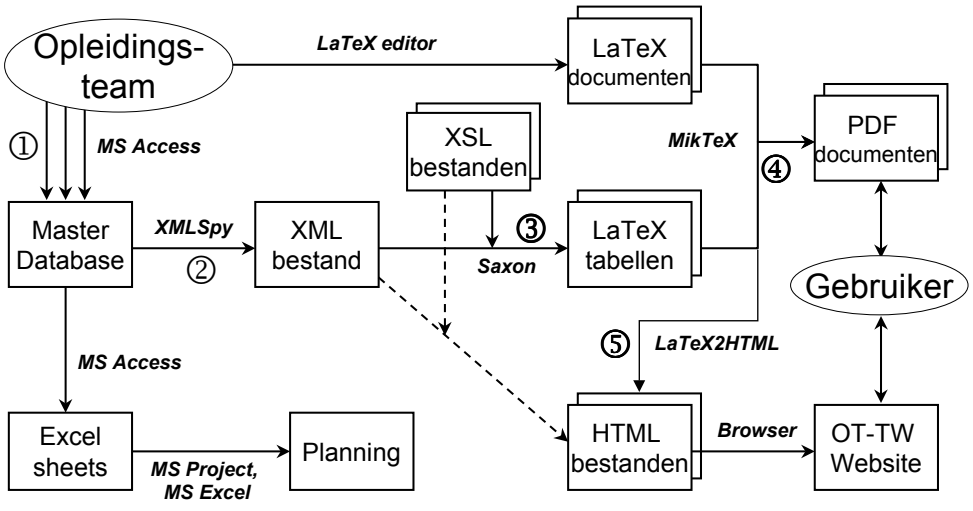
□ MS Access voorziet in een replicatiemechanisme waardoor gegevensinvoer parallel gedaan kan worden. Bij het samenvoegen in de master-database worden eventuele conflicten gemeld, die dan opgelost kunnen worden.

□ In MS Access kunnen geavanceerde programmeerfuncties ingebouwd worden met Visual Basic. Dit is gebruikt om volgorderelaties binnen clusters automatisch te berekenen.

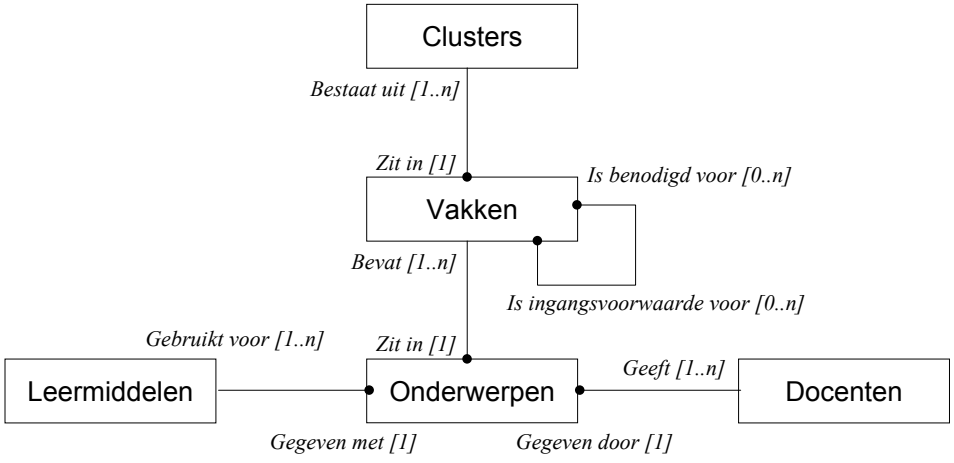
Naast de in dit artikel beschreven LaTeX uitvoer kunnen vanuit de master-database eenvoudig allerlei overzichten gemaakt worden voor bijvoorbeeld analyses, brainstorm sessies en planningen. De master-database is de primaire bron voor overzichten over de opleiding.

Stap 2: Maken van de XML-bestanden uit de master-database

Ondanks dat de nieuwste MS Access versie volgens de fabrikant XML ondersteunt, wordt het programma *XMLSpy Professional Edition* [4] gebruikt om vanuit MS Access de juiste XML-bestanden te maken[3]. Belangrijk is dat in deze stap een gelaagd XML-bestand conform Figuur 1 gemaakt wordt door informatie uit meerdere



Figuur 3. Overzicht van de conversie van de informatie.



Figuur 4. Het informatiemodel van de master-database (vereenvoudigd).

tabellen te combineren tot één uitvoer, daarbij gebruik makend van de relaties. Dit is niet triviaal omdat daarvoor selectie-opdrachten (*queries*) nodig zijn. Of MS Access deze mogelijkheid biedt is ons onbekend, de gebruikte MS Access versie (versie 2000) kent geen XML uitvoer.

XMLSpy haalt de benodigde informatie direct uit het MS Access `.mdb`-bestand. In totaal zijn er 5 XML-bestanden nodig vanuit de master-database. Voor de verdere bewerking moeten de XML-bestanden omgezet worden naar *UTF-8*, dat afwijkt van de standaard windows karakterset. Dit kan met XMLSpy.

Stap 3: Converteren van informatie uit de XML-bestanden naar LaTeX

De techniek van de XML-naar-LaTeX conversie is al uitgebreid aan de orde gekomen, we gaan nu kijken hoe dat gedaan is.

De conversie wordt uitgevoerd door een zgn. *XSLT-processor*. Deze leest de XML en XSLT-bestanden en produceert de gewenste uitvoer. Voorbeelden van (freeware) XSLT-processoren zijn MsXML (in Internet Explorer), Saxon en Xalan.

Voor ons systeem hebben we gebruik gemaakt van *Instant-Saxon* [5]. Het voordeel van dit programma is dat het via een *Command Line Interface (CLI)* aan te roepen is, op de volgende manier: `saxon -o apparch.tex clusters.xml genclusterlijst.xslt`, waarbij `apparch.tex` het uitvoerbestand is. Omdat Saxon via een CLI aangeroepen wordt, kunnen meerdere commando's gecombineerd worden in één *batchbestand* dat met een muisklik uitgevoerd wordt. De volgende stappen worden doorlopen voor de conversie naar LaTeX.

Preprocessing. Deze XSLT conversie is nodig voor een aantal taken:

- conversie van de "," in numerieke waarden naar ".". MS Access hanteert de Nederlandse conventie, XML gaat uit van de Amerikaanse conventie[4];
- conversie van *harde returns* in de database informatie naar `\newline` commando's om de formattering van de gegevens in de master-database te behouden;
- conversie van *speciale karakters*, gecodeerd in XML of UTF-8 naar de LaTeX equivalent. Voorbeelden zijn: "`ï`" naar "`\"{\i}`", "`>`" naar "`$>$`" en "`&`" naar "`\&`". Om gebruik van LaTeX commando's in de master-database informatie mogelijk te maken, worden de "\", "{" en "}" *niet* geconverteerd.

Genereren. Ons systeem maakt gebruik van meerdere XSLT conversies. In totaal worden voor het programmaboek vijf XSLT-bestanden gebruikt voor 16 XSLT conversies, die resulteren in evenzovele LaTeX bestanden. Die bestanden bevatten tabellen, annexen etc. XSLT kent een *parameter mechanisme*, waarmee vanuit de CLI extra besturingsparameters aan XSLT meegegeven wordt. Daardoor zijn er "maar" vijf XSLT-bestanden nodig.

Opslaan. Door meerdere batch-bestanden te maken kan de informatie uit de XML-bestanden naar meerdere sets van LaTeX-bestanden geconverteerd worden. In ons geval zijn dat naast het programmaboek ook een studiegids en een opleidingsreglement. Het systeem garandeert *eenduidige informatie over meerdere documenten*. Door de LaTeX-bestanden meteen in de juiste TeX directory te zetten blijft het geheel overzichtelijk.

Stap 4: Compileren van het uiteindelijke LaTeX document
De gegenereerde bestanden zijn voorzien van vooraf gedefinieerde labels en worden met `\include` of `\input` commando's in het LaTeX moederdocument ingevoegd. Dat `.tex` moederdocument bepaalt de structuur van het document en bevat de diverse hoofdstukken met grote delen tekst. Met `pdflatex` wordt het geheel geconverteerd naar een Acrobat `.pdf`-document, waarbij de `hyperref` package wordt gebruikt om een navigeerbare browser versie te verkrijgen. Alle tabellen, gedetailleerde annexen en hyperlinks in het document bevatten *eenduidige informatie*, ze komen immers uit één bron.

Stap 5: Maken van een HTML document
Een extra stap is de conversie van LaTeX naar HTML, via LaTeX2HTML. Hiermee wordt het programmaboek op de in de inleiding genoemde TW website gepubliceerd. Het oorspronkelijke plan was om dit direct uit het XML-bestand te gaan doen met Java-scripts, met een zoek- en selecteerfunctie (stippellijn in Figuur 3). Door de complexe structuur van de opleiding bleek dit moeilijker dan gedacht en is dit wegens tijdgebrek niet verder opgepakt.

Conclusie

De ervaringen met dit documentsysteem zijn als volgt samen te vatten:

□ De benodigde hulpmiddelen voor de opzet van de geautomatiseerde conversie zijn in korte tijd geïdentificeerd (aangekochte software) of ontwikkeld (XSLT en XPath). Nodig zijn een beperkt budget, enige programmeerervaring en wat doorzettingsvermogen; het totale project is door de auteurs binnen enkele maanden in deeltijd gerealiseerd.

□ De doelstellingen zijn gehaald: er wordt gewerkt met eenduidige brongegevens, de gegevens zijn in een open standaard beschikbaar en er is een snel en betrouwbaar documentsysteem tot stand gebracht (van database naar 225 pagina's programmaboek binnen enkele minuten). Het heeft ertoe geleid dat de HDO deze werkwijze nu waarschijnlijk voor al haar documentatie zal overnemen. Ook een aantal TU faculteiten heeft belangstelling getoond.

□ Het vastleggen van layout-details in grote curriculumtabellen via een XSLT style sheet is geen sinecure. Na de conversie kan de layout niet meer gerepareerd worden (dit zou na iedere conversie opnieuw moeten gebeuren) dus moet de layout, kloppend voor alle informatie, in het XSLT-bestand vastgelegd worden. Ver-

der maken keuzemogelijkheden en uitzonderingen in in het curriculum het geheel complexer dan in dit artikel beschreven.

□ De auteurs zijn er niet in geslaagd om andere teamleden (voor zover nog geen gebruiker) over de streep te trekken om LaTeX te gebruiken. De drempel is te hoog gebleven, ondanks demo's in vriendelijkere (WYSIWYG) omgevingen zoals SciWriter of Publicon. Ook het stringente beleid voor software installatie binnen Defensie speelt mee.

□ Dit project was tevens bedoeld om te onderzoeken of ook de curriculuminhoud zelf (het lesmateriaal) eenduidig en gemakkelijk in XML vast te leggen was, om van daaruit automatisch HTML browser files en via LaTeX `pdf` hardcopies te genereren. In dit opzicht is de conclusie negatief: eenduidigheid is maar al te moeizaam te verkrijgen (pakketten gebruiken allerlei definities voor hun XML) en ook betrouwbare hulpmiddelen voor de omzetting van bestaand LaTeX materiaal zijn niet gevonden. Daar is dus voorlopig gekozen voor `.tex` bronfiles.

Noten

1. Geen der auteurs had eerdere ervaring met XML; de eerste auteur bezat enige ervaring met TeX in het kader van de aanmaak van software-documentatie bij de Koninklijke marine, de tweede auteur is sinds 1987 een frequente gebruiker van computer typesetting.
2. Het element `Import` wordt door omzetting naar XML gemaakt en heeft geen betekenis voor de documentgeneratie.
3. XMLSpy Professional Edition voorziet o.a. in een XSLT programmeeromgeving, die gebruikt is voor de ontwikkeling van de XSLT-bestanden
4. Er zijn andere mogelijkheden bekend, maar voor ons was deze manier de meest eenvoudige.

Referenties

[1] Boer, B. de, EuroTeX^{2}001 presentatie "From database to presentation via XML, XSLT and ConTeXt", *MAPS 26* (2001), 27 - 39.
[2] Harold, E.R. en Scott Means, W., *XML in a Nutshell, 2nd Edition*, O-Reilly, 2002.
[3] Sall, K.B., *XML Family of Specifications*, Addison-Wesley, 2002.
[4] *XMLSpy Professional Edition, Version 2004 rel. 3*, http://www.altova.com, 2004.
[5] Kay, M., *Instant Saxon Version 6.5.3*, http://saxon.sourceforge.net, 2003.

Oscar Boot, Frans Absil
Sectie Sensor-, Wapen- en Commandosystemen
Koninklijk Instituut voor de Marine
Den Helder
o.boot@kim.nl

Bundeling van conferentieverslagen

Abstract
In dit artikel wordt beschreven hoe *proceedings* voor een workshop of conferentie gemaakt kunnen worden met behulp van PdfLaTeX en de packages *pdfpages*, *fancyhdr* en *hyperref*.

Keywords
fancyhdr, hyperref, pdflatex, pdfpages, proceedings, conferentieverslag

Inleiding

Al enige jaren maak ik voor verschillende conferenties die binnen mijn leerstoel plaatsvinden de proceedings. Steeds probeer ik dit zo handig en foutloos mogelijk te doen, zodat bij het aanbreken van de deadline voor de drukker (en dus ook voor de auteurs van de artikelen) de kans op fouten zo klein mogelijk wordt.
Er is duidelijk een verandering in de tijd te zien:

- mijn eerste proceedings werden op papier afgedrukt, met een typmachine voorzien van paginanummers en zo afgeleverd bij de drukker;
- de volgende proceedings werden op papier afgedrukt dat ik al van te voren had voorzien van paginanummers;
- toen ik de drukker vroeg of hij direct pdf-files kon afdrukken, was zijn antwoord positief. Daardoor kon een grote verbetering in de afdrukkwaliteit bereikt worden. Met behulp van Adobe Acrobat maakte ik de proceedings door van elk artikel via Adobe Acrobat Distiller een pdf-document (al of niet met een lege bladzijde) in te voegen. De inhoudsopgave maakte ik met de hand. Vrij veel handwerk dat met name als de deadline naderde veel concentratie vroeg en helaas ook wel eens kleine fouten gaf, want het is en blijft dan mensenwerk.

Toch bleef bij mij de gedachte steeds knagen dat dit toch veel handiger moest kunnen.

Eerste poging: combine package

Eerst heb ik geprobeerd het *combine* package te gebruiken. Hiermee is het mogelijk meerdere TeX-documenten met elkaar te combineren tot een groot TeX-document. Combine probeert met allerlei slimme trucs de per artikel gebaseerde settings ook per artikel te laten bestaan en na wat hernoemen van verschillende environments tot één groot LaTeX-document te komen. Maar elke TeX-auteur heeft de vrijheid om eigen environments en dergelijke te maken en te gebruiken. Vooral bij de wiskundige artikelen komt dat veelvuldig voor. Zulke variaties gaan elkaar toch beïnvloeden en wijziging in het ene artikel worden werkzaam in latere artikelen. Dit was

voor mij op dat moment niet werkbaar voor het maken van proceedings zonder de auteurs helemaal in een keurslijf vast te leggen.

Tweede poging: pdfpages package

Toen kwam ik het package *pdfpages* onverwacht tegen en ging daarmee experimenteren. Al gauw werd mij duidelijk dat dit *de* goede weg was om in te slaan. Kort gezegd komt het neer op het maken van een raamwerk dat voor elk samengesteld document (als voorbeeld proceedings) gebruikt kan worden.

- □ titelpagina
- □ cip (pagina voor CIP gegevens Koninklijke Bibliotheek, Den Haag)
- □ voorwoord
- □ inhoudsopgave (toc)
- □ pdf artikel 1
- □ pdf artikel 2
- □ …
- □ pdf artikel n
- □ index van auteurs

Eerst moest er een goede style file komen voor de artikelen zelf. Vele conferenties hebben deze al beschikbaar gemaakt.[1]

Toen werd het tijd voor een meer concrete aanpak. Hoe worden artikelen die gebruik maken van deze style verwerkt tot goede proceedings? In grote lijnen komt het erop neer dat de maker van de proceedings de namen van de auteurs, de titel van het artikel en de filenaam van het pdf-document invult en dat dan de rest van de proceedings goed wordt verwerkt. Naam en titel worden gebruikt voor de inhoudsopgave en als kop boven een artikel.

Omdat de style file voor de artikelen een empty pagestyle gebruikt is het nu mogelijk om bij de kop tevens een paginanummer toe te voegen. Elke pagina uit een artikel aangeleverd in pdf door een auteur wordt via fancyhdr voorzien van titel/naam en paginanummer!

```
\documentclass[twoside]{article}
\usepackage[%showframe,
  a4paper,
  body={150mm,225mm},
  top=37mm,
  inner=35mm
]{geometry}
\usepackage{pdfpages}
\usepackage{fancyhdr}
\usepackage{nextpage}
%%
%% Sommige packages zijn niet meer nodig bij dit voorbeeld
%%
\usepackage{calc}
\usepackage[latin1]{inputenc}
\usepackage{makeidx}
\usepackage[
  bookmarksopen=true,bookmarksopenlevel=2,
  bookmarks=true,plainpages=false,pdfpagelabels=true,
```

```
  colorlinks=true, linkcolor=blue]{hyperref}
%%
%% Afkomstig uit LaTeX Companion (2nd Edition) pagina 236
%%
\newcommand{\clearemptydoublepage}
           {\thispagestyle{empty}\movetooddpage\thispagestyle{empty}}

\renewcommand{\footrulewidth}{0pt}
\renewcommand{\indexname}{List of authors}
%%
%% De argumenten meegegeven aan \includepdfset worden bij elk nieuw
%% pdf-document welke ingelezen wordt met '\includepdf{}' uitgevoerd.
%% Pagecommand wordt zelfs elke pagina uitgevoerd.
%% De offset-parameter brengt de pdf netjes onder de fancy headers
%%
\includepdfset{offset=30pt 0pt,link,pages=-,pagecommand={}}

%%
%% \Inx{level}{title} maakt een index entry
%%
\newcommand{\Inx}[2]{
  \def\level{#1}
  \def\een{1}
  \addtocontents{toc}{\protect%
    \contentsline{\ifx\level\een section\else subsection\fi}
                 {\ifx\level\een \emph{#2}\else#2\fi}
                 {\protect\hyperpage{\thepage}}
                 {}
    }
  \pdfbookmark[#1]{#2}{\thepage}
}
%%
%% \indexpage command
%% Met dit command wordt de indexpagina gemaakt.
%% Eerst wordt fancyhead en toc-entry + pdfbookmark gemaakt
%% Tenslotte de \printindex voor de feitelijke indexpagina
%%
\newcommand\indexpage{
  \renewcommand\plainheadrulewidth{0.4pt}
  \pagestyle{fancyplain}
  \markright{}
  \clearemptydoublepage
  \fancyhead[LO,RE]{\indexname}%
  \Inx{1}{\indexname}
  \printindex
}

%%
%% Newenvironment paperenv: zorgt ervoor dat bij een
%% \begin{paperenv} elk artikel aan de rechterkant van het boek
%% begint.
%%
\newenvironment{paperenv}{\clearemptydoublepage}{}

%%
```

```
%% \paper command:
%% Vijf argumenten:
%% 1: Complete titel van het artikel zoals in de toc moet komen;
%% 2: Titel van artikel (misschien korter) zoals komt te staan in de
%%    header van de bladzijden;
%% 3: Volledige beschrijving van de auteurs zoals int de toc moet
%%    komen;
%% 4: Beschrijving van de auteurs (mogelijk verkort, bijv. Piet Puk et
%%    al.) zoals deze komt te staan in de header van een pagina;
%% 5: Naam van de pdf-file
%%
\newcommand\paper[5]{
  \pagestyle{fancy}
  \markright{}
  \fancyfoot{}
  \fancyhead{}
  \fancyhead[LO,RE]{#2\ --- #4}%
  \fancyhead[RO,LE]{\thepage}%
  \Inx{1}{#1}
  \Inx{2}{#3}
  \includepdf{#5.pdf}%
}

%%
%% Eigen formaat voor Table of Content: zie blz 50 Companion (sec Edt)
%% en code uit article.cls
%%

\makeatletter
\def\@undottedtocline#1#2#3#4#5{\ifnum #1>\c@tocdepth \else           %
  \vskip \z@ plus.2\p@                                               %
  { \leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip %
    \parindent #2\relax\@afterindenttrue                             %
    \interlinepenalty\@M                                             %
    \leavevmode                                                      %
    \@tempdima 0pt \relax \advance\leftskip \@tempdima \hbox{}%      %
    \hskip -\leftskip                                                %
    #4\nobreak\hfill \nobreak                                        %
      \null\par}\fi}                                                 %
\renewcommand\l@section{\@dottedtocline{1}{1.5em}{2.3em}}
\renewcommand\l@subsection{\@undottedtocline{2}{3.8em}{2.3em}}
\renewcommand\@dotsep{2.0}
%%
\makeatother

\parindent=0pt
\parskip=0pt
\makeindex
\begin{document}
  \pagenumbering{roman}
  \include{header}
  \include{cip}
  \include{preface}
  \include{previous-workshops}
  \tableofcontents
```

```
  \input{papers}
  \indexpage
\end{document}
```

De included LaTeX files spreken voor zich en worden niet verder uitgelegd in dit artikel. `header.tex`, `cip.tex`, `preface.tex` en `previous-workshops.tex` bevatten alleen tekst die betrekking heeft op een specifieke conferentie/workshop.

De file `papers.tex` daarentegen verdient het om verder uitgelegd te worden. Deze file bevat alle papers en ziet er als volgt uit:

```
{\newpage{\thispagestyle{empty}\movetooddpage}}
\pagenumbering{arabic}
%%
%% Zet pagina nummer op 1 voor eerste artikel
%%
\setcounter{page}{1}
%%
%% Plaats eventueel een regel 'Invited Speakers' in de toc
%%
\addtocontents{toc}{{\large\bf{Invited Speakers}}\protect\\}

%%
%% Start van eerste paper
%%
\begin{paperenv}
   \index{Jan, Jansen@\textbf{Jan, Jansen} (invited speaker)}
   \index{Piet, Pietersen}
   \paper{Lange titel voor de table-of-content}
         {Korte titel voor header}
        {Jan Jansen (Deparment etc) and Piet Pietersen (Department etc)}
         {Jan Jansen and Piet Pietersen}
         {pdfile1}
\end{paperenv}
...

%%
%% Plaats eventueel een regel 'Regular Speakers' in de toc
%%
\addtocontents{toc}{\mbox{}\protect\\}
\addtocontents{toc}{{\large\bf{Regular Speakers}}\protect\\}

\begin{paperenv}
     \index{...}
     \index{...}
     \paper{...}{...}{... and ... (...)}{... and ...}{pdffile}
  \end{paperenv}
```

Gebruik van hyperref

Omdat er gebruik gemaakt wordt van PdfLaTeX kan het zinvol zijn om de hyperlink opties van verschillende pdf-readers te gebruiken:

- het linken van artikelen aan de auteursnaam in de index kan erg handig zijn voor de lezer. Hyperref zorgt er zelf voor dat de index gelinkt wordt, als de link-optie wordt meegegeven aan het hyperref-package.

- het hebben van bookmarks van artikel en auteurs is een extra optie die te gebruiken is in een pdf-document. Bookmarks in pdf kunnen gemaakt worden door `\pdfbookmark[1]{#1}{\thepage}`, waarin tussen `[]` het level/niveau staat, gevolgd door tekst en paginanummer.

Conclusie

Met behulp van *pdfpages*, *fancyhdr* en *hyperref* is met behulp van PdfLaTeX het maken van proceedings veel eenvoudiger geworden, mits de artikelen van de auteurs voldoen aan de eisen die gesteld worden aan de conferentie (aanleveren in PS of PDF, gebruik van type1 fonts!).

Noten

1. Wij kozen deze keer voor de ICML style file. Deze conferentie wordt ook door de auteurs van onze workshop bezocht en de auteurs zijn redelijk bekend met deze style en door deze style file te gebruiken kunnen de auteurs na aanpassingen dit artikel ook later bij een ICML conferentie gebruiken.

Hendri Hondorp
g.h.w.hondorp@ewi.utwente.nl
Universiteit Twente,
Faculty of EEMCS,
Computer Science, Human Media Interaction group, Enschede

…three, two, one…
a quest for the number of numbers

Keywords
enumerate, numbered items, lists, counting backwards, etaremune

Abstract
This article briefly describes the seemingly trivial task of numbering items in a list with decreasing numbers, starting with the number equal to the amount of items in the list.

Introduction

Some time ago, someone on the TEX-NL mailinglist asked whether it would be possible to have items in an `enumerate` environment numbered with decreasing numbers instead of increasing ones. Hendri presented a quick hack on the LaTEX `enumerate` environment as he didn't know the revnum [3] package yet. At first, his solution required submitting the amount of items to the environment as a parameter, but later versions of his solution didn't need that input anymore.

After the existence of the revnum package was brought to his attention, Hendri looked into the way this package operates. This package consumes numerous TEX counter registers. Hendri found that his own solution, with some extra work, could be an alternative to achieve the goal with a low consumption of counter registers. His solution would evolve to be the etaremune package [1], which is the subject of this small article.

Isn't this trivial?

The original idea does sound trivial indeed, but it really isn't when you realize that once LaTEX has read and typeset input, there is no way to change the typeset version anymore. So, once we have typeset the first item in the list and gave it, for instance, number '1', we can't change that number to a '3' anymore if we find 2 more items in the list. So, the basic problem is to already know the number of items in a list before starting and typesetting this list.

Two possible solutions spring to mind.

2. Extract the entire body of the environment, that should number items backwards, to an external file, parse it to find the number of `\item`'s in the body and insert the body back into the input stream. This is very tricky for several reasons, one being the fact that LaTEX `enumerate` environments can be nested and hence our new environment should also support that. For other possible problems related to this technique, see for instance the extract package [2].
1. Apply a two-step procedure as is usual in LaTEX (for instance to construct the table of contents). In the first LaTEX run, we typeset the item numbers in a default way and at the same time, count the number of items and save this to the `.aux` file. At the second run, we use this number to properly set the starting number (the number of the first item) and all following items.

Technique 1 seems the easiest and hence the most appealing one. This is the technique employed by the revnum and etaremune packages.

How to implement it then?

So now we know what technique we will use. But how do we implement it? First, we need to keep in mind that the environment should be nestable, just as `enumerate` is. Secondly, we need a counter register to do the actual counting of the items in the environment. Unfortunately, we cannot use the fact that the TEX group inside `\begin...\end` will do all counting locally to the environment and not touch the counting of items in an environment in a higher level (in case we are in a nested environment). This doesn't work[1] as the code needed to count items can only be inserted into `\item` via `\@itemlabel` which will be typeset in a box, keeping counting local to a single `\item` and resetting the counter again after leaving the item label box. Adding a `\global` isn't a solution as it makes counting escape all the way to the top level, hence also modifying the counted number of items in higher level lists.

How to proceed? The revnum package tries to work around this problem by using a counter register for every new `revnum` environment that the user starts (plus four to keep track of the number of the envir-

onment in the document etcetera). However, when using a lot of `revnum` environments and of course other packages that consume the necessary counters as well, one might end up asking a TEXpert on the TEX-NL mailinglist to extend TEX.

But we really do not need to use that many counters. We only need a counter for counting the items and one for counting the environments in the document, so we can say in the next LATEX run: "list 113 has 37 items". And when we encounter a nested list, we need to make sure that, when leaving this nested list, we can reset the two counters and continue counting the items in the list in the higher level (which has a lower number than the nested list). We do this by temporarily saving counter values to macros when entering a nested list.

Besides counting items on the first LATEX run, we need to provide some output. The revnum package then starts at 26, so that second and fourth level lists can use the alphabet without having problems of trying to access letter -1. However, when one adds items to an existing list, the package will still try to typeset letter -1 from the alphabet, resulting in problems. The etaremune package typesets an `etaremune` environment as an `enumerate` environment on the first run and makes sure that counters do not get lower than 0 to avoid errors when adding entries to nested lists.

New perspectives

The package opens a hoard of new possibilities. So far, counting backwards has been the 'modus operandi' in just a few organizations. But they use it well and we could all benefit from their expertise. At NASA, people count backwards toward the launching moment of rockets and space shuttles. Engineers operating explosives for demolition of buildings also count backwards toward the toppling of the tower. In general, controlled processes that go 'boom' are counted backwards as they advance. But low-tech destructive and high-tech astro-pyrotechnics aren't the only industries using the countdown method of advancement. In popular music it's customary to count backwards to the presentation of the current best selling ditty on the airwaves. And last but not least, Agatha Christie's play 'Ten Little Indians' about ten strangers in a secluded place uses the thrill-enhancing device of countdown as the characters are knocked off one at a time.

With the basic concept of this new package, countdown can change the way we create documents and even improve the way we read and meet.

For instance, how relevant is it to read a novel and see the page number incline? Unless we know the page number of the last page, there's little use for the page number other than a way to refer to a given section. But if the novel starts at the highest page number, those tiny numbers at the corners of the page indicate how far we are from arriving at the final page. The first page has its own unmistakable prominence at the very start of the book, after the title page and the table of contents, with a high number and the final page is at the very back of the book but it's proudly adorned with the number one! Using the ideas of this article, one could perform this task. It would require some work[2], but it would be worth it. There would even be the option of having a page zero, for 'zeroing in' on the epilogue!

And meetings can have an agenda starting with the highest number, so attendants can work towards a fixed end and not be bothered about a seemingly endless conference. Meetings could get snappier and more concise, making the organization more productive just by counting more sensibly.

This package could change the way we meet, the way we read and the way we think. And we can reserve the count-upwards method for those enigmatic increments towards ends unknown, like our life cycle. Adding one to our age every year does provide us with the mild illusion that in principle we could go on living forever, until the numbers run out. At least we give it a try...

Footnotes

1. Well, it might work if we would redefine `\item` and friends, but that is too risky in relation to other packages.
2. In fact, one would need to change the command `\stepcounter{page}` to `\addtocounter{page}{-1}` in LATEX's output routine additionally to finding the number of pages (which could also be done with the lastpage package).

References

[1] Hendri Adriaens. etaremune package, v1.1, 2005/04/15. `CTAN:/macros/latex/contrib/etaremune`.

[2] Hendri Adriaens. extract package, v1.7, 2005/03/31. `CTAN:/macros/latex/contrib/extract`.

[3] Jörn Wilms. revnum package, v1.0, 1997/05/10. `CTAN:/macros/latex/contrib/revnum`.

Frans Goddijn
`frans[at]goddijn.com`
Hendri Adriaens
`hendri[at]uvt.nl`

Compiling MetaPost figures under ConTEXt

Abstract
To teach yourself MetaPost, the book "Learning MetaPost by Doing" by André Heck is probably unsurpassed. However, the examples therein are processed on Unix using LaTEX. ConTEXt users have a bit of detective work to do before they can have successful compilations. If you are new to ConTEXt, the lines below may help save your a few hours of experimenting.
These instructions were extracted from the MetaFun manual by Hans Hagen (from chapters 1, 2 and 3), and from a small macro that he once gave me that makes it possible to use the `graph` package by John Hobby.

Running MetaPost

1. Prepare a text file with the same example as Figure 1 in Heck:

   ```
   beginfig(1);
   draw (0,0)--(10,0)--(10,10)--(0,10)--(0,0);
   endfig;

   end.
   ```

   Save it as `example.mp`
2. Compile this file under ConTEXt with the following command:

   ```
   texexec --mptex example.mp
   ```

   which produces some temporary files plus the encapsulated Postscript file `example.1`
3. You may remove the temporary files with the command:

   ```
   texutil --purge
   ```

   but this is not necessary since they will be overwritten with the next compile run. Note that sometimes this command will not remove all temporary files, because some of those files are used by ConTEXt to speed up a next compilation run.
4. The `example.1` file may be viewed with Ghostscript. Another way of viewing the results is to prepare a small file as follows:

   ```
   %output=pdftex
   \starttext
   \externalfigure[example.1]
   \stoptext
   ```

   and save it as `sample.tex`. Compile this with the command:

   ```
   texexec sample
   ```

which produces (among other files) the output file `sample.pdf` which can be viewed by Acrobat. At the same time this shows how you can use MetaPost output in a TEX document. Simply put it somewhere in your text file with the `\externalfigure[]` instruction.
5. If they bother you, you can again remove the temporary files by issuing the `texutil -purge` command.

MetaPost embedded in a ConTEXt document

By embedding the MetaPost figure in a ConTEXt document and compiling it, the whole process up to and including the creation of the resultant PDF file is achieved in a single processing step.

1. Type the following TEX file:

```
%output=pdftex          % produces PDF, not DVI output
\noheaderandfooterlines % removes the page number
\starttext
\startuseMPgraphic{one}
  draw (0,0)--(10,0)--(10,10)--(0,10)--(0,0);
\stopuseMPgraphic

\useMPgraphic{one}
\stoptext
```

and save it as `emtest.tex`. Note that the file extension is now `tex` and not `mp`. Note also that the `beginfig()` and `endfig` are replaced by the `\startuseMPgraphic{}` and `\stopuseMPgraphic`. Finally note that the number of the figure has been replaced by a name, which is usually more easily remembered.
2. Compile it with the command:

```
texexec emtest
```

which produces the result file `emtest.pdf` which can be viewed by Acrobat.

Using Hobby's `graph` package

For plotting scientific data the `graph` package offers several facilities, such as plotting data from a file and automatic scaling. With proper labeling good looking graphs can be programmed.

1. Type the following program to produce a figure similar to the one part of the Figure 1 from the recent paper by Maarten Sneep in MAPS 31, page 12:

```
initialize_numbers;
input graph;
Autoform:="@g";

beginfig(0);
draw begingraph(9cm,4cm);
  gdraw"Apples.dat";
  autogrid(itick.bot,itick.lft);
  glabel(btex \ss\tfa Apples etex,1995,1360);
endgraph;
endfig;
end.
```

and save it as `hob.mp`. `initialize_numbers` is probably ConTEXt. I don't know what the `Autoform` is for. The `graph` package must be separately loaded. The `glabel` between the `btex .. etex` pair allows regular ConTEXt commands such as `\ss` for sans serif or font size changing ones such as `\tfa`. Note that only the label is now in sans serif style. The numbers along the axes are still in the regular roman style. The next section shows how to change that too. `Apples.dat` is a column oriented ASCII text file with numbers. The first column is the x-axis, the second column is treated as the y-axis data. An empty line in the file stops the data input to the plot.

2. Compile this file with the command:

```
texexec --mptex hob.mp
```

which produces the result file `hob.0` again as an encapsulated Postscript file. It can be viewed and included in a document as shown before.

The `graph` package in embedded form

This section describes a MetaPost program embedded in a ConTEXt file.

1. Type the following TEX file:

```
%output=pdftex language=en % causes PDF result
\noheaderandfooterlines    % removes page number
\setupbodyfont[12pt]
\setupcolors[state=start]  % no colors if absent

% Somehow needed with the \type{graph} package: ---------------
\forceMPTEXgraphictrue
\appendtoks
  initialize_numbers;
  input graph;
  Autoform:="@g";
\to \MPinitializations

% Gives sansserif numbers text and scale values: ---------------
\startMPinclusions
def do_initialize_numbers =
  if not numbers_initialized :
    init_numbers ( textext.raw("$\hbox{\ss -}$"),
                   textext.raw("$\hbox{\ss 1}$"),
                   textext.raw("${\times}\hbox{\ss 10}$"),
                   textext.raw("${}^{\hbox{\ss -}}$"),
                   textext.raw("${}^{\hbox{\ss 2}}$")) ;
    numbers_initialized := true ;
  fi ;
enddef ;
\stopMPinclusions

\startMPenvironment
  \switchtobodyfont[14pt,ss]
\stopMPenvironment

\startuseMPgraphic{two}
  path HLineone; HLineone:=(0,1)--(1200,1);
  path HLineten; HLineten:=(0,10)--(1200,10);
  draw begingraph(6cm,6cm);
    setcoords(linear,log);
    setrange(0,.1,1200,100);
```

```
    glabel.lft(textext.raw("$\hbox{\ss\tfb 0.1}$"),0,0.1);
    glabel.lft(btex \ss\tfb 1 etex,0,1);
    glabel.lft(btex \ss\tfb 10 etex,0,10);
    glabel.lft(btex \rm\tfb100 etex,0,100);
    glabel.lft(btex \rm 2.24 etex,1200,2.24);
    gdraw HLineone withcolor .65white;
    gdraw HLineten withcolor .65white;
    gdraw "one.g" withcolor .65blue;
    autogrid(otick.bot,);
  endgraph;
\stopuseMPgraphic

\useMPgraphic{two}

\stoptext
```

and save this file as `hobby.tex`.

The paragraphs with `\appendtoks` and `\startMPinclusions` can be placed in a separate file named `m-graph.tex` and input to the program with the instructions:

```
\usemodule[graph]
```

ConTEXt automatically adds the `m-` and loads the proper file.

2. Compile this file with the command:

```
texexec hobby
```

Conclusion

ConTEXt has many more commands and instructions to work with MetaPost files. In addition Hans Hagen has programmed nice and useful extensions to MetaPost that can be found in his MetaFun manual and can be loaded with: `input m-tool`. To compile the examples in Heck's "Learning MetaPost by doing" I believe the above suffices. Programming in MetaPost is often a bit tedious but publication quality graphs can be obtained with patience, that are scalable, colorful, and of very great precision. Have fun.

Karel H. Wesseling

Learning METAPOST by Doing

Introduction

TeX is the well-known typographic programming language that allows its users to produce high-quality typesetting especially for mathematical text. METAPOST is the graphic companion of TeX. It is a graphic programming language developed by John Hobby that allows its user to produce high-quality graphics. It is based on Donald Knuth's METAFONT, but with PostScript output and facilities for including typeset text. This course is only meant as a short, hands-on introduction to METAPOST for newcomers who want to produce rather simple graphics. The main objective is to get students started with METAPOST on a UNIX platform[1]. A more thorough, but also much longer introduction is the Metafun manual of Hans Hagen [Hag02]. For complete descriptions we refer to the METAPOST Manual and the Introduction to METAPOST of its creator John Hobby [Hob92a, Hob92b].

We have followed a few didactical guidelines in writing the course. Learning is best done from examples, learning is done from practice. The examples are often formatted in two columns, as follows:[2]

[1] You can also run METAPOST on a windows platform, e.g., using MikTeX and the WinEdt shell.

[2] On the left is printed the graphic result of the METAPOST code on the right. Here, a square is drawn.

```
beginfig(1);
draw unitsquare scaled 1cm;
endfig;
```

The exercises give you the opportunity to practice METAPOST, instead of only reading about the program. Compare your answers with the ones in the section 'Solutions to the Exercises'.

A Simple Example

METAPOST is not a WYSIWYG drawing tool like `xfig` or `xpaint`. It is a graphic document preparation system. First, you write a plain text containing graphic formatting commands into a file by means of your favourite editor. Next, the METAPOST program converts this text into a PostScript document that you can preview and print. In this section we shall describe the basics of this process.

Running METAPOST

EXERCISE 1 Do the following steps:

1. Create a text file, say `example.mp`, that contains the following very simple METAPOST document:

```
beginfig(1);
draw (0,0)--(10,0)--(10,10)--(0,10)--(0,0);
endfig;

end;
```

   For example, you can use the editor XEmacs:

```
xemacs example.mp
```

   The above UNIX command starts the editor and creates the source file `example.mp`. It is common and useful practice to always give a METAPOST source file a name with extension `.mp`. This will make it easier for you to distinguish the source document from files with other extensions, which METAPOST will create during the formatting.

2. Generate from this file PostScript code. Here the METAPOST program does the job:

```
mpost example
```

   It is not necessary to give the filename extension here. METAPOST now creates some additional files:

   `example.1` a PostScript file that can be printed and previewed;
   `example.log` METAPOST's log file.

3. Check that the file `example.1` contains the following normal Encapsulated PostScript code:[3]

[3] Notice that the bounding box is larger than you might expect, due to the default width of the line drawing the square.

```
%!PS
%%BoundingBox: -1 -1 11 11
%%Creator: MetaPost
%%CreationDate: 2003.05.11:2203
%%Pages: 1
%%EndProlog
%%Page: 1 1
 0 0.5 dtransform truncate idtransform setlinewidth pop [] 0 setdash
 1 setlinecap 1 setlinejoin 10 setmiterlimit
newpath 0 0 moveto
10 0 lineto
10 10 lineto
0 10 lineto
0 10 lineto
0 0 lineto stroke
showpage
%%EOF
```

4. Preview the PostScript document on your computer screen, e.g., by typing:

   ```
   gs example.1
   ```

   You will notice that `gs` does not use the picture's bounding box; alternatively, try `ggv` instead of `gs`, or:

5. Convert the PostScript document into a printable PDF-document:

   ```
   epstopdf example.1
   ```

   It creates the file `example.pdf` that you can you can view on the computer screen with the Adobe Acrobat Reader by entering the command:

   ```
   acroread example.pdf
   ```

   You can print this file in the usual way. The picture should look like the following small square:

Using the Generated PostScript in a LaTeX Document

EXERCISE 2 Do the following steps:

1. Create a file, say `sample.tex`, that contains the following lines of LaTeX commands that will include the image:

```
\documentclass{article}
\usepackage{graphicx}
\DeclareGraphicsRule{*}{mps}{*}{}
\begin{document}
\includegraphics{example.1}
\end{document}
```

   Above, we use the extended `graphicx` package for including the external graphic file that was prepared by METAPOST. The `DeclareGraphicsRule` statement causes all file extensions that are not associated with a well-known graphic format to be treated as Encapsulated PostScript files.

2. Typeset the LaTeX-file:

```
pdflatex sample
```

When typesetting is successful, the device independent file `sample.pdf` is generated.

The Structure of a METAPOST Document

We shall use the above examples to explain the basic structure of a METAPOST document. We start with a closer look at the slightly modified METAPOST code in the file `example.mp` of our first example:

```
beginfig(1); % draw a square
draw (0,0)--(10,0)--(10,10)
  --(0,10)--(0,0);
endfig;
end;
```

This example illustrates the following list of general remarks about regular METAPOST files

- It is recommended to end each METAPOST program in a file with extension `mp` so that this part of the name can be omitted when invoking METAPOST.
- Each statement in a METAPOST program is ended by a semicolon. Only in cases where the statement has a clear endpoint, e.g., in the `end` and `endfig` statement, you may omit superfluous semicolons. We shall not do this in this tutorial. You can put two or more statements on one line as long as they are separated by semicolons. You may also stretch a statement across several lines of code and you may insert spaces for readability.
- You can add comments in the document by placing a percentage symbol % in front of the commentary.
- A METAPOST document normally contains a sequence of `beginfig` and `endfig` pairs with an `end` statement after the last one. The numeric argument to the `beginfig` macro determines the name of the output file that will contain the PostScript code generated by the next graphic statements before the corresponding `endfig` command. In the above case, the output of the `draw` statement between `beginfig(1)` en `endfig` is written in the file `example.1`. In general, a METAPOST document consists of one or more instances of

  `beginfig(`*figure number*`);`
  graphic commands
  `endfig;`

  followed by `end`.
- The `draw` statement with the points separated by two hyphens (`--`) draws straight lines that connect the neighbouring points. In the above case for example, the point `(0,0)` is connected by straight lines with the point `(10,0)` and `(0,10)`. The picture is a square with edges of size 10 units, where a unit is $\frac{1}{72}$ of an inch. We shall refer to this default unit as a 'PostScript point' or 'big point' (`bp`) to distinguish it from the 'standard printer's point' (`pt`), which is $\frac{1}{72.27}$ of an inch. Other units of measure include `in` for inches, `cm` for centimetres, and `mm` for millimetres. For example,

  ```
  draw (0,0)--(1cm,0)--(1cm,1cm)--(0,1cm)--(0,0);
  ```

  generates a square with edges of size 1cm. Here, `1cm` is shorthand for `1*cm`. You may use `0` instead of `0cm` because `cm` is just a conversion factor and `0cm` just multiplies the conversion factor by zero.

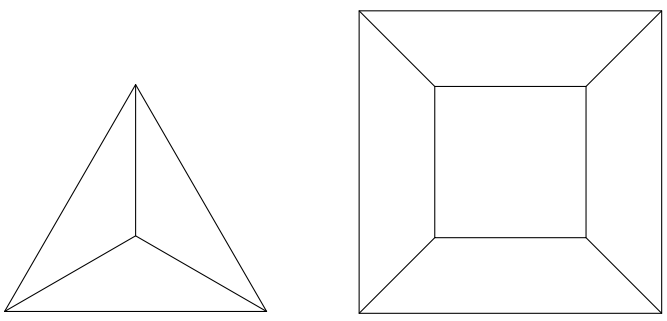
EXERCISE 3 Create a METAPOST file, say `exercise3.mp`, that generates a circle of diameter 2cm using the `fullcircle` graphic object.

EXERCISE 4

1. Create a METAPOST file, say `exercise4.mp`, that generates an equilateral triangle with edges of size 2cm.
2. Extend the METAPOST document such that it generates in a separate file the PostScript code of an equilateral triangle with edges of size 3cm.

EXERCISE 5 Define your own unit, say `0.5cm`, by the statement `u=0.5cm;` and use this unit `u` to generate a regular hexagon with edges of size 2 units.

EXERCISE 6 Create the following two pictures:



Numeric Quantities
Numeric quantities in METAPOST are represented in fixed point arithmetic as integer multiples of $\frac{1}{65536} = 2^{-16}$ and with absolute value less or equal to $4096 = 2^{12}$. Since METAPOST uses fixed point arithmetic, it does not understand exponential notation such as `1.23E4`. It would interpret this as the real number `1.23`, followed by the symbol `E`, followed by the number `4`. Assignment of numeric values can be done with the usual `:=` operator. Numeric values can be shown via the `show` command.

EXERCISE 7

1. Create a METAPOST file, say `exercise7.mp`, that contains the following code

```
numeric p,q,n;
n := 11;
p := 2**n;
q := 2**n+1;
show p,q;
end;
```

   Find out what the result is when you run the above METAPOST program.

2. Replace the value of `n` in the above METAPOST document by 12 and see what happens in this case (Hint: press Return to get processing as far as possible). Explain what goes wrong.
3. Insert at the top of the current METAPOST document the following line and see what happens now when you process the file.

```
warningcheck := 0;
```

The *numeric* data type is used so often that it is the default type of any non-declared variable. This explains why `n := 10;` is the same as `numeric n; n := 10;` and

why you cannot enter `p := (0,0);` nor `p = (0,0);` to define the point, but must use `pair p; p := (0,0);` or `pair p; p = (0,0);` .

If Processing Goes Wrong
If you make a mistake in the source file and METAPOST cannot process your document without any trouble, the code generation process is interrupted. In the following exercise, you will practice the identification and correction of errors.

EXERCISE 8 Deliberately make the following typographical error in the source file `example.mp`. Change the line

```
draw (0,0)--(10,0)--(10,10)--(0,10)--(0,0);
```

into the following two lines

```
draw (0,0)--(10,0)--(10,10)
draw (10,10)--(0,10)--(0,0);
```

1. Try to process the document. METAPOST will be unable to do this and the processing would be interrupted. The terminal window where you entered the `mpost` command looks like:

```
(example.mp
! Extra tokens will be flushed.
<to be read again>
                   addto
draw->addto
           .currentpicture.if.picture(EXPR0):also(EXPR0)else:doubl...
<to be read again>
                   ;
l.3 draw (10,10)--(0,10)--(0,0);

?
```

   In a rather obscure way, the METAPOST program notifies the location where it signals that something goes wrong, viz., at line number 3. However, this does not mean that the error is necessarily there.

2. There are several ways to proceed after the interrupt. Enter a question mark and you see your options:

```
? ?
Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
?
```

3. Press RETURN. METAPOST will continue processing and tries to make the best of it. Logging continues:

```
[1] )
1 output file written: example.1
Transcript written on example.log.
```

4. Verify that only the following path is generated:

```
newpath 0 0 moveto
10 0 lineto
10 10 lineto stroke
```

5. Format the METAPOST document again, but this time enter the character `e`.

Your default editor will be opened and the cursor will be at the location where METAPOST spotted the error. Correct the source file[4] by adding a semicolon at the right spot, and give the METAPOST processing another try.

Basic Graphical Primitives

In this section you will learn how to build up a picture from basic graphical primitives such as points, lines, and text objects.

Pair

The *pair* data type is represented as a pair of numeric quantities in METAPOST. On the one hand, you may think of a pair, say `(1,2)`, as a location in two-dimensional space. On the other hand, it represents a vector. From this viewpoint, it is clear that you can add or subtract two pairs, apply a scalar multiplication to a pair, and compute the dot product of two pairs.

You can render a point `(x,y)` as a dot at the specified location with the statement

```
draw (x,y);
```

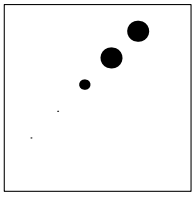
Because the drawing pen has by default a circular shape with a diameter of 1 PostScript point, a hardly visible point is rendered. You must explicitly scale the drawing pen to a more appropriate size, either locally in the current statement or globally for subsequent drawing statements. You can resize the pen for example with a scale factor 4 by

```
draw (x,y) withpen pencircle scaled 4; % temporary change of pen
```

or by

```
pickup pencircle scaled 4; % new drawing pen is chosen
draw (x,y);
```

EXERCISE 9 Explain the following result:



```
beginfig(1)
draw unitsquare scaled 70;
draw (10,20);
draw (10,15) scaled 2;
draw (30,40) withpen pencircle scaled 4;
pickup pencircle scaled 8;
draw (40,50);
draw (50,60);
endfig;
end;
```

Assignment of pairs is often not done with the usual `:=` operator, but with the equation symbol `=`. As a matter of fact, METAPOST allows you to use linear equations to define a pair in a versatile way. A few examples will do for the moment.

- Using a name that consists of the character `z` followed by a number, a statement such as `z0 = (1,2)` not only declares that the left-hand side is equal to the right-hand side, but it also implies that the variables `x0` and `y0` exist and are equal to `1` and `2`, respectively. Alternatively, you can assign values to the numeric variable `x1` and `y1` with the result that the pair `(x1,y1)` is defined and can be referred to by the name `z1`.
- A statement like

```
z1 = -z2 = (3,4);
```

[4] If you have not specified another editor in the EDITOR environment variable, then the `vi`-editor will be started. You can leave this editor by entering `ZZ`. In the c-shell you can add in the file `.cshrc` the line `setenv MPEDIT 'xemacs +%d %s'` so that XEMACS is used.

is equivalent to

```
z1 = (3,4);
z2 = -(3,4);
```

□ If two pairs, say `z1` and `z2`, are given, you can define the pair, say `z3`, right in the middle between these two points by the statement `z3 = 1/2[z1,z2]`.

□ When you have declared a pair, say `P`, then `xpart P` and `ypart P` refer to the first and second coordinate of `P`, respectively. For example,

```
pair P;  P = (10,20);
```

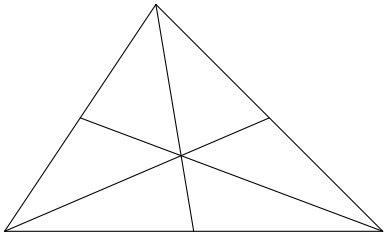
is the same as

```
pair P;
xpart P = 10;
ypart P = 20;
```

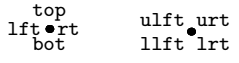
EXERCISE 10 Verify that when you use a name that begins with `z.` followed by a sequence of alphabetic characters and/or numbers, a statement such as `z.P = (1,2)` not only declares that the left-hand side is equal to the right-hand side, but it also implies that the variables `x.P` and `y.P` exist and are equal to 1 and 2, respectively. Alternatively, you can assign values to the numeric variable `x.P` and `y.P`, with the result that the pair `(x.P,y.P)` is defined and can be referred to by the name `z.P`.

EXERCISE 11

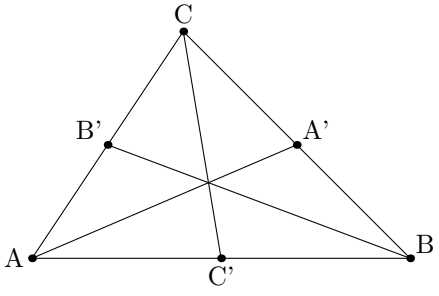
1. Create the following geometrical picture of an acute-angled triangle together with its three medians[5]:



2. The `dotlabel` command allows you to mark a point with a dot and to position some text around it. For instance, `dotlabel.lft("A",(0,0));` generates a dot with the label `A` to the left of the point. Other `dotlabel` suffixes and their meanings are shown in the picture below:



Use the `dotlabel` command to put labels in the picture in part 1, so that it looks like

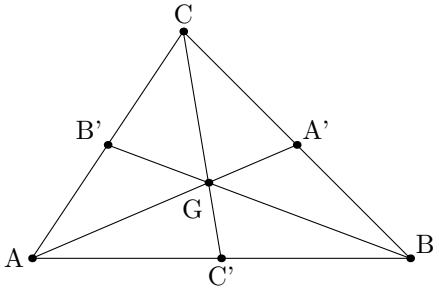


[5]The A-median of a triangle ABC is the line from A to the midpoint of the opposite edge BC.

3. Recall that `1/2[z1,z2]` denotes the point halfway between the points `z1` and `z2`. Similarly, `1/3[z1,z2]` denotes the point on the line connecting the points `z1` and `z2`, one-third away from `z1`. For a numeric variable, which is possibly unknown yet, `c[z1,z2]` is `c` times of the way from `z1` to `z2`. If you do not want to waste a name for a variable, use the special name `whatever` to specify a general point on a line connecting two given points:

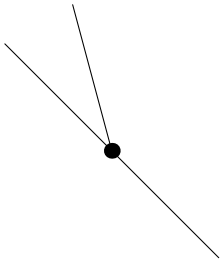
   ```
   whatever[z1,z2];
   ```

   denotes some point on the line connecting the points `z1` and `z2`. Use this feature to define the intersection point of the medians, also known as the centre of gravity, and extend the above picture to the one below. Use the `label` command, which is similar to the `dotlabel` command except that it does not draw a dot, to position the character `G` around the centre of gravity. If necessary, assign `labeloffset` another value so that the label is further away from the centre of gravity.



EXERCISE 12 The `dir` command is a simple way to define a point on the unit circle. For example, `dir(30);` generates the pair `(0.86603,0.5)` $\left(=(\frac{1}{2}\sqrt{3},\frac{1}{2})\right)$. Use the `dir` command to generate a regular pentagon.

EXERCISE 13 Use the `dir` command to draw a line in northwest direction through the point $(1,1)$ and a line segment through this point that makes an angle of 30 degrees with the line. Your picture should look like



Path

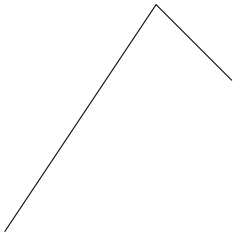
Open and Closed Curves. METAPOST can draw straight lines as well as curved ones. You have already seen that a `draw` statement with points separated by `--` draws straight lines connecting one point with another. For example, the result of

```
draw p0--p1--p2;
```

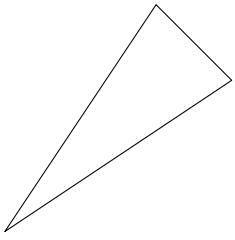
after defining three points by

```
pair p[];  p0 = (0,0);  p1 = (2cm,3cm);  p2 = (3cm,2cm);
```

is the following picture.



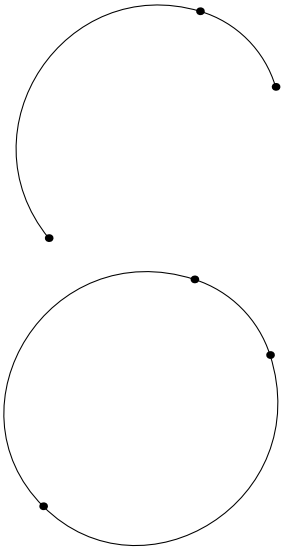
Closing the above path is done either by extending it with `--p0` or by connecting the first and last point via the `cycle` command. Thus, the path `p0--p1--p2--cycle`, when drawn, looks like



The difference between these two methods is that the path extension with the starting point only has the optical effect of closing the path. This means that only with the `cycle` extension it really becomes a closed path.

EXERCISE 14 Verify that you can only fill the interior of a closed curve with some colour or shade of gray, using the `fill` command, when the path is really closed with the `cycle` command. The gray shading is obtained by the directive `withcolor c*white`, where `c` is a number between 0 and 1.

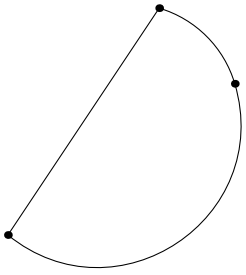
Straight and Curved Lines. Compare the pictures from the previous subsection with the following ones, which show curves[6] through the same points.



```
u := 1cm;
pair p[];
p0 = (0,0); p1 = (2u,3u); p2 = (3u,2u);

beginfig(1);
draw p0..p1..p2; % draw open curve
for i=0 upto 2:
  draw p[i] withpen pencircle scaled 3;
endfor; % draw defining points
endfig;

beginfig(2);
draw p0..p1..p2..cycle; % draw closed curve
for i=0 upto 2:
  draw p[i] withpen pencircle scaled 3;
endfor; % draw defining points
endfig;
end;
```

[6] the curves through the three points are a circle or a part of the circle

Just use -- where you want straight lines and .. where you want curves.

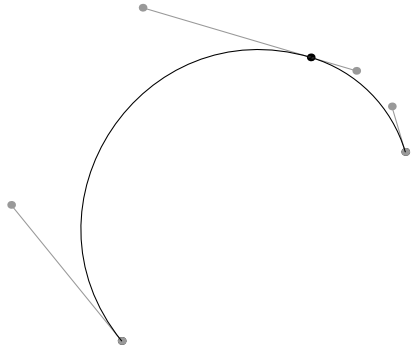


```
beginfig(1);
u := 1cm;
pair p[];
p0 = (0,0); p1 = (2u,3u); p2 = (3u,2u);
draw p0--p1..p2..cycle;
for i=0 upto 2:
  draw p[i] withpen pencircle scaled 3;
endfor;
endfig;
end;
```

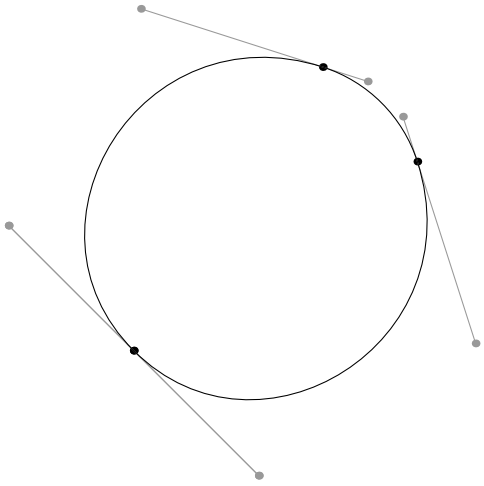
Construction of Curves. When METAPOST draws a smooth curve through a sequence of points, each pair of consecutive points is connected by a cubic Bézier curve, which needs, in order to be determined, two intermediate control points in addition to the end points. The points on the curved segment from points p_0 to p_1 with post control point c_0 and pre control point c_1 are determined by the formula

$$p(t) = (1-t)^3 p_0 + 3(1-t)^2 t c_0 + 3(1-t) t^2 c_1 + t^3 p_1 \,,$$

where $t \in [0,1]$. METAPOST automatically calculates the control points such that the segments have the same direction at the interior knots. In the figure below, the additional control points are drawn as gray dots and connected to their parent point with gray line segments. The curve moves from the starting point in the direction of the post control point, but possibly bends after a while in another direction. The further away the post control point is, the longer the curve keeps this direction. Similarly, the curve arrives at a point coming from the direction of the pre control point. The further away the pre control point is, the earlier the curve gets this direction. It is as if the control points pull their parent point in a certain direction and the further away a control point is, the stronger it pulls. By default in METAPOST, the incoming and outgoing direction at a point on the curve are the same so that the curve is smooth.



```
u := 1.25cm;
color gray; gray := 0.6white;
pair p[];
p0 = (0,0); p1 = (2u,3u); p2 = (3u,2u);
def drawpoint(expr z, c) = draw z
  withpen pencircle scaled 3 withcolor c;
enddef;

beginfig(1);
path q; q := p0..p1..p2;
for i=0 upto length(q):
  drawpoint(point i of q, black);
  p3 := precontrol i of q;
  p4 := postcontrol i of q;
  draw p3--p4 withcolor gray;
  drawpoint(p3, gray);
  drawpoint(p4, gray);
endfor;
draw q;
endfig;
```



```
beginfig(2);
path q; q := p0..p1..p2..cycle;
for i=0 upto length(q):
  drawpoint(point i of q, black);
  p3 := precontrol i of q;
  p4 := postcontrol i of q;
  draw p3--p4 withcolor gray;
  drawpoint(p3, gray);
  drawpoint(p4, gray);
endfor;
draw q;
endfig;

end;
```

Do not worry when you do not understand all details of the above METAPOST program. It contains features and programming constructs that will be dealt with later in the tutorial.

There are various ways of controlling curves:

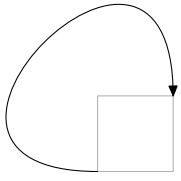
- Vary the angles at the start and end of the curve with one of the keywords `up`, `down`, `left`, and `right`, or with the `dir` command.
- Specify the requested control points manually.
- Vary the inflection of the curve with `tension` and `curl`. `tension` influences the curvature, whereas `curl` influences the approach of the starting and end points.

```
pair p[];  p0:=(0,0);  p1:=(1cm,1cm);
def drawsquare = draw unitsquare
    scaled 1cm withcolor 0.7white;
enddef;
```

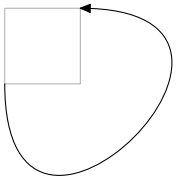
```
beginfig(1);
drawsquare;  drawarrow p0..p1;
endfig;
```

```
beginfig(2);
drawsquare;  drawarrow p0{right}..p1;
endfig;
```

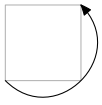
```
beginfig(3);
drawsquare;  drawarrow p0{up}..p1;
endfig;
```



```
beginfig(4);
drawsquare;  drawarrow p0{left}..p1;
endfig;
```

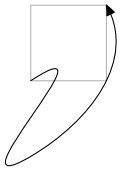


```
beginfig(5);
drawsquare;  drawarrow p0{down}..p1;
endfig;
```

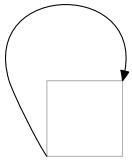


```
beginfig(6);
drawsquare;  drawarrow p0{dir(-45)}..p1;
endfig;
```

```
beginfig(7);
drawsquare;  drawarrow p0..
  controls (0,1cm) and (1cm,0) ..p1;
endfig;
```



```
beginfig(8);
drawsquare;  drawarrow p0{curl 80}..
  (0,-1cm)..{curl 8}p1;
endfig;
```



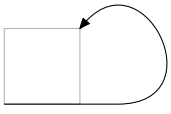
```
beginfig(9);
drawsquare;  drawarrow p0..tension(2)
  ..(0,1cm)..p1;
endfig;

end;
```

The METAPOST operators `--`, `---`, and `...` have been defined in terms of `curl` and `tension` directives as follows:

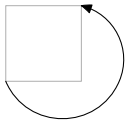
```
def --  = {curl 1}..{curl 1}      enddef;
def --- = .. tension infinity ..  enddef;
def ... = .. tension atleast 1 .. enddef;
```

The meaning of `...` is "choose an inflection-free path between the points unless the endpoint directions make this impossible". The meaning of `---` is "get a smooth connection between a straight line and the rest of the curve".

```
pair p[];  p0:=(0,0);  p1:=(1cm,1cm);
def drawsquare = draw unitsquare scaled 1cm
    withcolor 0.7white;
enddef;
```



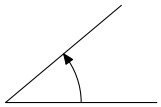
```
beginfig(1);
drawsquare;  drawarrow p0---(1.5cm,0)..p1;
endfig;
```



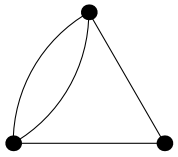
```
beginfig(2);
drawsquare;  drawarrow p0...(1.5cm,0)..p1;
endfig;

end;
```

The above examples were also meant to give you the impression that you can draw in METAPOST almost any curve you wish.

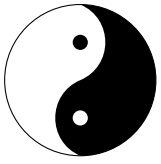
EXERCISE 15 Draw an angle of 40 degrees that looks like



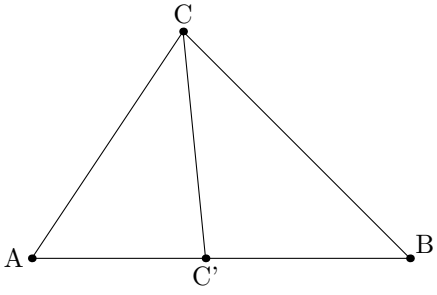
EXERCISE 16 Draw a graph that looks like



EXERCISE 17 Draw the Yin-Yang symbol[7] that looks like

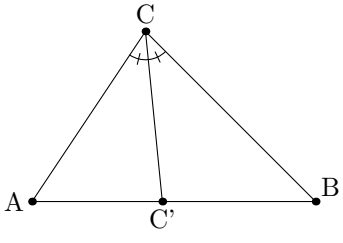


Angle and Direction Vector
In a previous exercise you have already seen that the `dir` command generates a pair that is a point on the unit circle at a given angle with the horizontal axis. The inverse of `dir` is `angle`, which takes a pair, interprets it as a vector, and computes the two-argument arctangent, i.e., it gives the angle corresponding with the vector. In the example below we use it to draw a bisector of a triangle.



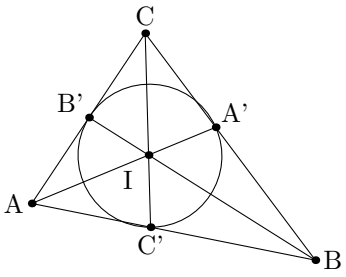

```
pair A, B, C, C';
u := 1cm; A=(0,0); B=(5u,0); C=(2u,3u);
C' = whatever[A,B] = C + whatever*dir(
       1/2*angle(A-C)+1/2*angle(B-C));
beginfig(1)
draw A--B--C--cycle;  draw C--C';
dotlabel.lft("A",A); dotlabel.urt("B",B);
dotlabel.top("C",C); dotlabel.bot("C'",C');
endfig;
end;
```

EXERCISE 18 Change the above picture to the following geometrical diagram, which illustrates better that a bisector is actually drawn for the acute-angled triangle.



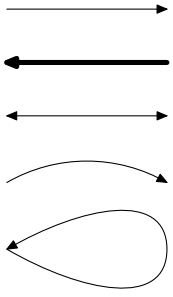
EXERCISE 19 Draw a picture that shows all the bisectors of a acute-angled triangle. Your picture should look like

[7] See `www.chinesefortunecalendar.com/YinYang.htm` for details about the symbol.



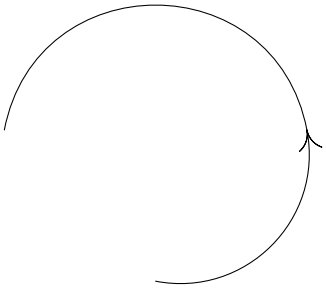
In this way, it illustrates that the bisectors of a triangle go through one point, the so-called incenter, which is the centre of the inner circle of the triangle.

Arrow
The `drawarrow` command draws the given path with an arrowhead at the end. For double-headed arrows, simply use the `drawdblarrow` command. A few examples:



```
beginfig(1);
drawarrow (0,0)--(60,0);
drawarrow reverse((0,-20)--(60,-20))
  withpen pencircle scaled 2;
drawdblarrow (0,-40)--(60,-40);
drawarrow (0,-65){dir(30)}..
  {dir(-30)}(60,-65);
drawarrow (0,-90){dir(-30)}..{up}(60,-90)..
  {dir(-150)}cycle;
endfig;
end;
```

If you want arrowheads of different size, you can change the arrowhead length through the variable `ahlength` (4bp by default) and you can control the angle at the tip of the arrowhead with the variable `ahangle` (45 degrees by default). You can also completely change the definition of the `arrowhead` procedure. In the example below, we draw a curve with an arrow symbol along the path. As a matter of fact, the path is drawn in separate pieces that are joined together with the `&` operator.

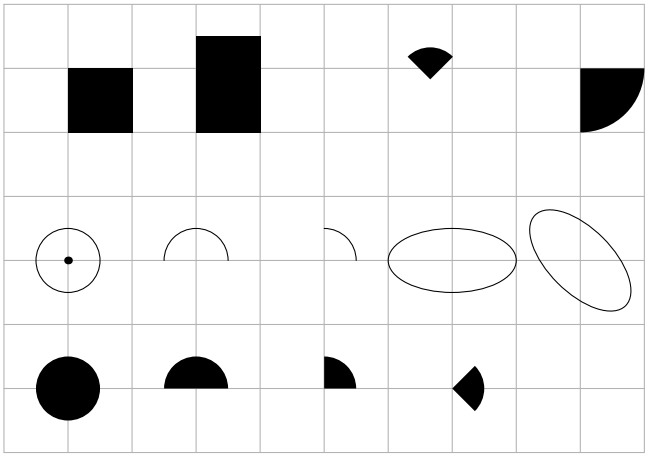


```
beginfig(1);
save arrowhead;
vardef arrowhead(expr p) =
  save A,u,a,b; pair A,u; path a,b;
  A := point length(p)/2 of p;
  u := unitvector(direction length(p)/2 of p);
  a := A{-u}..(A - ahlength*u rotated 30);
  b := A{-u}..(A - ahlength*u rotated -30);
  (a & reverse(a) & b & reverse(b))--cycle
enddef;
u:=2cm; ahlength:=0.3cm;
drawarrow (0,0)..(u,u)..(-u,u);
endfig; end;
```

Circle, Ellipse, Square, and Rectangle
You have already seen that you can draw a circle through three points `z0`, `z1`, and `z2`, that do not lie on a straight line with the statement `draw z0..z1..z2..cycle;`. But METAPOST also provides predefined paths to build circles and circular disks or parts of them. Similarly, you can draw a rectangle once the four corner points, say `z0`, `z1`, `z2`, and `z3`, are known with the statement `draw z0--z1--z2--z3--cycle;`. The path `(0,0)--(1,0)--(1,1)--(0,1)--cycle` is in METAPOST predefined as `unitsquare`.

Path	*Definition*
`fullcircle`	circle with diameter 1 and centre $(0,0)$.
`halfcircle`	upper half of `fullcircle`
`quartercircle`	first quadrant of `fullcircle`
`unitsquare`	`(0,0)--(1,0)--(1,1)--cyle`

You can construct from these basic paths any circle, ellipse, square, or rectangle by rotating (`rotated` operator), by scaling (operators `scaled`, `xscaled`, and `yscaled`), and/or by translating the graphic object (`shifted` operator). Keep in mind that the ordering of operators has a strong influence on the final shape. But pictures say more than words. The diagram



is drawn with the following METAPOST code.

```
beginfig(1);
u := 24;  % 24 = 24bp = 1/3 inch
for i=-1 upto 9: draw (i*u,4u)--(i*u,-3u) withcolor 0.7white; endfor;
for i=-3 upto 4: draw (-u,i*u)--(9u,i*u)  withcolor 0.7white; endfor;
dotlabel("",origin);  % the grid with reference point (0,0) has been drawn
draw fullcircle scaled u;
draw halfcircle scaled u shifted (2u,0);
draw quartercircle scaled u shifted (4u,0);
draw fullcircle xscaled 2u yscaled u shifted (6u,0);
draw fullcircle xscaled 2u yscaled u rotated -45 shifted (8u,0);
fill fullcircle scaled u shifted (0,-2u);
fill halfcircle--cycle scaled u shifted (2u,-2u);
path quarterdisk;  quarterdisk := quartercircle--origin--cycle;
fill quarterdisk scaled u shifted (4u,-2u);
fill quarterdisk scaled u rotated -45 shifted (6u,-2u);
fill quarterdisk scaled u shifted (6u,-2u) rotated 45;
fill quarterdisk rotated -90 scaled 2u shifted (8u,3u);
fill unitsquare scaled u shifted (0,2u);
fill unitsquare xscaled u yscaled 3/2u shifted (2u,2u);
endfig;
end;
```

Text

You have already seen how the `dotlabel` command can be used to draw a dot and a label in the neighbourhood of the dot. If you do not want the dot, simply use the

`label` command:

`label.`*suffix*`(`*string expression*, *pair*`)` ;

It uses the same suffixes as the `dotlabel` command to position the label relative to the given pair. No suffix means that the label is printed with its centre at the specified location Available directives for the specification of the label relative to the given pair are (see also page 63):

`top` : *top*	`ulft` : *upper left*
`lft` : *left*	`urt` : *upper right*
`rt` : *right*	`lrt` : *lower right*
`bot` : *bot*	`llft` : *lower left*

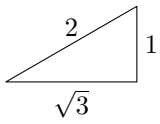
The distance from the pair to the label is set by the numeric variable `labeloffset`.

The commands `label` and `dotlabel` both use a string expression for the label text and typeset it in the default font, which is likely to be `"cmr10"` and which can changed through the variables `defaultfont` and `defaultscale`. For example,

```
defaultfont := "ptmr";
defaultscale := 12pt/fontsize(defaultfont);
```

makes labels come out as Adobe Times-Roman at about 12 points.

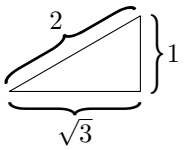
Until now the string expression in a text command has only been a string delimited by double quotes (optionally joined to another string via the concatenation operator `&`). But you can also bracket the text with `btex` and `etex` (do not put it in quotes this time) and pass it to TEX for typesetting. This allows you to use METAPOST in combination with TEX for building complex labels. Let us begin with a simple example:




```
beginfig(1);
z0 = (0,0);  z1 = (sqrt(3)*cm,0);
z2 = (sqrt(3)*cm,1cm);
draw z0--z1--z2--cycle;
label.bot(btex $\sqrt{3}$ etex, 1/2[z0,z1]);
label.rt(btex 1 etex, 1/2[z1,z2]);
label.top(btex 2 etex, 1/2[z0,z2]);
endfig;
end;
```

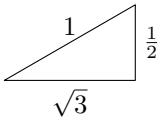
Whenever the METAPOST program encounters `btex` *typesetting commands* `etex`, it suspends the processing of the input in order to allow TEX to typeset the commands and the `dvitomp` preprocessor to translate the typeset material into a picture expression that can be used in a `label` or `dotlabel` statement. The generated low level METAPOST code is placed in a file with extension `.mpx`. Hereafter METAPOST resumes its work.

We speak about a picture expression that is created by typesetting commands because it is a graphic object to which you can apply transformation. This is illustrated by the following example, in which we use diagonal curly brackets and text.




```
beginfig(1);
z0 = (0,0);  z1 = (sqrt(3)*cm,0);
z2 = (sqrt(3)*cm,1cm);
draw z0--z1--z2--cycle;
label.bot(btex $\lbrace$ etex rotated 90
  xscaled 5 yscaled 1.4, 1/2[z0,z1]);
label.rt((btex $\rbrace$ etex) xscaled 1.3
  yscaled 3, 1/2[z1,z2]);
label(btex $\lbrace$ etex xscaled 1.5 yscaled 5.7
  rotated -60, 1/2[z0,z2] + dir(120)*2mm);
labeloffset:=3.5mm;
label.bot(btex $\sqrt{3}$ etex, 1/2[z0,z1]);
label.rt(btex 1 etex, 1/2[z1,z2]);
label(btex 2 etex, 1/2[z0,z2]+dir(120)*5mm);
endfig;
end;
```

Until now we have only used plain TEX commands. But what if you want to run another TEX-version? The following example shows how you can use a `verbatimtex.....etex` block to specify that LATEX is used and which style and/or packages are chosen.

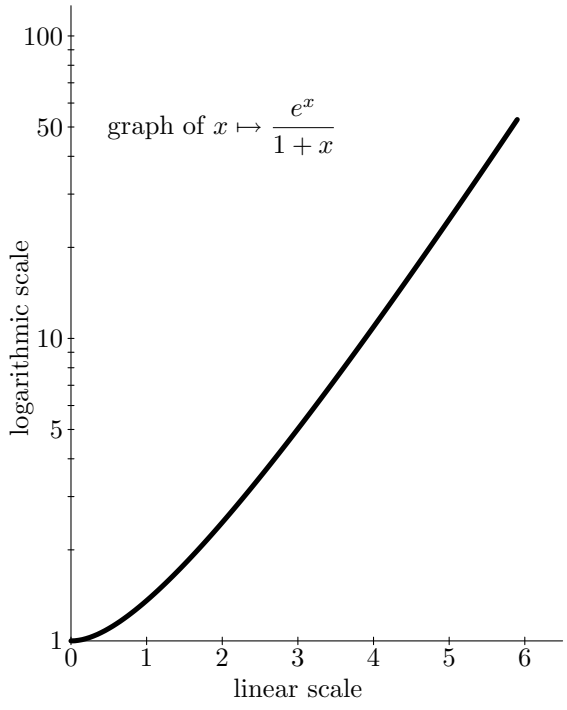



```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

beginfig(1);
z0 = (0,0); z1 = (sqrt(3)*cm,0);
z2 = (sqrt(3)*cm,1cm);
draw z0--z1--z2--cycle;
label.bot(btex $\sqrt{3}$ etex, 1/2[z0,z1]);
label.rt(btex $\frac{1}{2}$ etex, 1/2[z1,z2]);
label.top(btex 1 etex, 1/2[z0,z2]);
endfig;

end;
```

One last remark about using LATEX: Between `btex` and `etex`, you cannot use displayed math mode such as `$$\frac{x}{x+1}$$`. You must use `$\displaystyle \frac{x}{x+1}$` instead.

Let us use what we have learned so far in this chapter in a more practical example: drawing the graph of the function $x \mapsto \dfrac{e^x}{1+x}$ from 0 to 5 with the vertical axis in a logarithmic scale. The picture is generated by the following METAPOST code:

```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

% some function definitions
vardef exp(expr x) = (mexp(256)**x) enddef;
vardef ln(expr x) = (mlog(x)/256) enddef;
vardef log(expr x) = (ln(x)/ln(10)) enddef;
vardef f(expr x) = (exp(x)/(1+x)) enddef;
ux := 1cm; uy := 4cm;
```




```
beginfig(1)
numeric xmin, xmax, ymin, ymax;
xmin := 0; xmax := 6;
ymin := 0; ymax := 2;
% draw axes
draw (xmin,0)*ux -- (xmax+1/2,0)*ux;
draw (0,ymin)*uy -- (0,ymax+1/10)*uy;
% draw tickmarks and labels on horizontal axis
for i=0 upto xmax:
  draw (i,-0.05)*ux--(i,0.05)*ux;
  label.bot(decimal(i),(i,0)*ux);
endfor;
% draw tickmarks and labels on vertical axis
for i=2 upto 10:
  draw (-0.01,log(i))*uy--(0.01,log(i))*uy;
  draw (-0.01,log(10*i))*uy--(0.01,log(10*i))*uy;
endfor;
for i=0 upto 2: label.lft(decimal(10**i), (0,i)*uy); endfor;
for i=0 upto 1: label.lft(decimal(5*(10**i)),
  (0,log(5*(10**i)))*uy); endfor;
% compute and draw the graph of the function
xinc := 0.1;
path pts_f;
pts_f := (xmin*ux,log(f(xmin))*uy)
  for x=xmin+xinc step xinc until xmax:
    .. (x*ux,log(f(x))*uy)
  endfor;
draw pts_f withpen pencircle scaled 2;
% draw title
label(btex graph of $\displaystyle x\mapsto\frac{e^x}{1+x}$
  etex, (2ux,1.7uy));
% draw axis explanation
labeloffset := 0.5cm;
label.bot(btex linear scale etex, (3,0)*ux);
label.lft(btex logarithmic scale etex rotated(90),
  (0,1)*uy);
endfig;

end;
```

The above code needs some explanation.

First of all, METAPOST does not know about the exponential or logarithmic function. But you can easily define these functions with the help of the built-in functions `mexp(x)` = exp(x/256) and `mlog(x)` = 256 ln x. Note that we have reserved the name `log` for the logarithm with base 10 in the above program.

As you will see later in this tutorial, METAPOST has several repetition control structures. Here we apply the `for` loop to draw tick marks and labels on the axes and to compute the path of the graph. The basic form is:

`for` *counter* = *start* `step` *stepsize* `until` *finish* :
    loop text
`endfor;`

Instead of `step 1 until`, you may use the keyword `upto`. `downto` is another word for `step -1 until`.

In the following code snippet

```
for i=0 upto xmax:
  draw (i,-0.05)*ux--(i,0.05)*ux;
  label.bot(decimal(i),(i,0)*ux);
endfor;
```

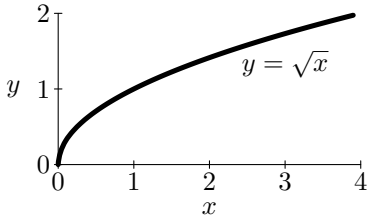
the input lines are two statements: one to draw tick marks and the other to put a label. We use the `decimal` command to convert the numeric variable `i` into a string

so that we can use it in the `label` statement. The following code snippet

```
pts_f := (xmin*ux,log(f(xmin))*uy)
    for x=xmin+xinc step xinc until xmax:
      .. (x*ux,log(f(x))*uy)
    endfor;
```

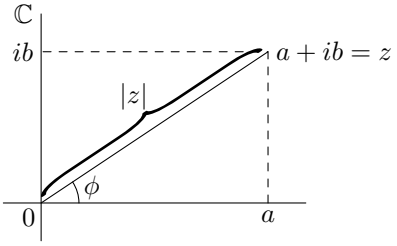
shows that you can also use the `for` loop to build up a single statement. The input lines within the `for` loop are pieces of a path definition. This mode of creating a statement may look strange at first sight, but it is an opportunity given by the fact that METAPOST consists more or less of two parts: a preprocessor and a PostScript generator. The preprocessor only reads from the input stream and prepares input for the PostScript generator.

EXERCISE 20 Draw the graph of the function $x \mapsto \sqrt{x}$ on the interval $(0, 2)$. Your picture should look like



Your METAPOST code should be such that only a minimal change in the code is required to draw the graph on a different domain, say $[0, 3]$.

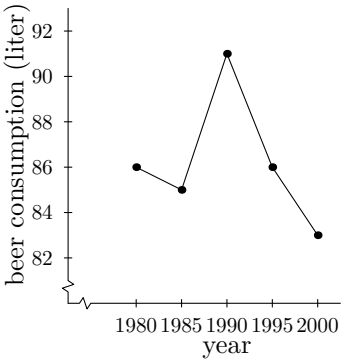
EXERCISE 21 Draw the following picture in METAPOST. The dashed lines can be drawn by adding `dashed evenly` at the end of the `draw` statement.



EXERCISE 22 The annual beer consumption in the Netherlands in the period 1980–2000 is listed below.

year	1980	1985	1990	1995	2000
litre	86	85	91	86	83

Draw the following graph in METAPOST.

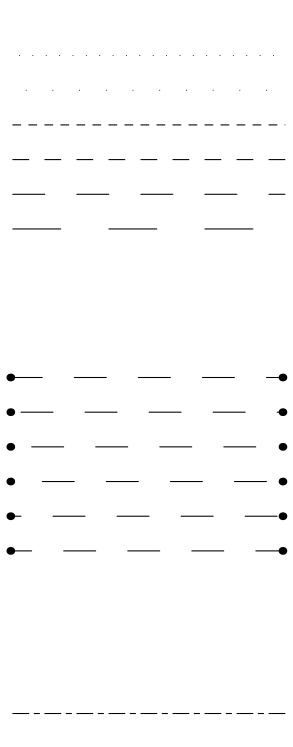


Style Directives

In this section we explain how you can alter the appearance of graphics primitives, e.g., allowing certain lines to be thicker and others to be dashed, using different colours, and changing the type of the drawing pen.

Dashing

Examples show you best how the specify a dash pattern when drawing a line or curve.



```
beginfig(1);
path p; p := (0,0)--(102,0);
def drawit (suffix p)(expr pattern) =
  draw p dashed pattern;
  p := p shifted (0,-13);
enddef;
drawit(p, withdots);
drawit(p, withdots scaled 2);
drawit(p, evenly);
drawit(p, evenly scaled 2);
drawit(p, evenly scaled 4);
drawit(p, evenly scaled 6);

p := (0,-150)--(102,-150);
def shiftit (suffix p)(expr s) =
  draw p dashed evenly scaled 4 shifted s;
  dotlabel("",point 0 of p);
  dotlabel("",point 1 of p);
  p := p shifted (0,-13);
enddef;
shiftit(p, (0,0));
shiftit(p, (4bp,0));
shiftit(p, (8bp,0));
shiftit(p, (12bp,0));
shiftit(p, (16bp,0));
shiftit(p, (20bp,0));

picture dd; dd :=
dashpattern(on 6bp off 2bp on 2bp off 2bp);
draw (0,-283)--(102,-283) dashed dd;
draw (0,-296)--(102,-296) dashed dd scaled 2;
endfig;

end;
```

In general, the syntax for dashing is

`draw` *path* `dashed` *dash pattern* `;`

You can define a dash pattern with the `dashpattern` function whose argument is a sequence of `on`/`off` distances. Predefined patterns are:

```
evenly   = dashpattern(on 3 off 3); % equal length dashes
withdots = dashpattern(off 2.5 on 0 off 2.5); % dotted lines
```

Colouring

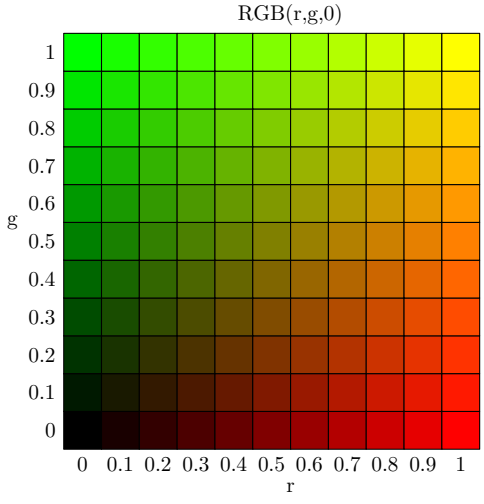
The `color` data type is represented as a triple (r, g, b) that specifies a colour in the RGB colour system. Each of r, g, and b must be a number between 0 and 1, inclusively, representing fractional intensity of red, green, or blue, respectively. Predefined colours are:

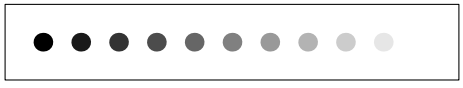
```
red   = (1,0,0);  green = (0,1,0);  blue = (0,0,1);
black = (0,0,0);  white = (1,1,1);
```

A shade of gray can be specified most conveniently by multiplying the `white` colour with some scalar between 0 and 1. The syntax of using a colour in a graphic statement is:

`withcolor` *colour expression* ;

Let us draw two colour charts:





```
beginfig(1);
u := 1/2cm;
defaultscale := 10pt/fontsize(defaultfont);
beginfig(1);
path sqr; sqr := unitsquare scaled u;
for i=0 upto 10:
  label.bot(decimal(i/10), ((i+1/2)*u,0));
  label.lft(decimal(i/10), (0,(i+1/2)*u));
  for j=0 upto 10:
    fill sqr shifted (i*u,j*u)
         withcolor (i*0.1, j*0.1,0);
    draw sqr shifted (i*u,j*u); % draw grid
  endfor;
endfor;
label.bot("r",(6u,-2/3u));
label.lft("g",(-u,6u));
label.top("RGB(r,g,0)", (6u,11u));
endfig; end;
```

```
beginfig(2);
draw (-u,-u)--(11u,-u)--(11u,u)--(-u,u)
     --cycle; % bounding box
pickup pencircle scaled 1/2u;
for i=0 upto 10:
  draw (i*u,0) withcolor i*0.1*white;
endfor;
endfig;

end;
```

EXERCISE 23 Compare the linear conversion from colour to gray, defined by the function

$$(r, g, b) \mapsto \frac{(r + g + b)}{3} \times (1, 1, 1)$$

with the following conversion formula used in black and white television:

$$(r, g, b) \mapsto (0.30r + 0.59g + 0.11b) \times (1, 1, 1)\,.$$

Specifying the Pen
In METAPOST you can define your drawing pen for specifying the line thickness or for calligraphic effects. The statement

`draw` *path* `withpen` *pen expression*;

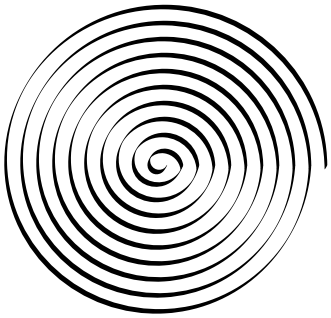
causes the chosen pen to be used to draw the specified path. This is only a temporary pen change. The statement

`pickup` *pen expression*;

causes the given pen to be used in subsequent `draw` statements. The default pen is circular with a diameter of 0.5 bp. If you want to change the line thickness, simply use the following pen expression:

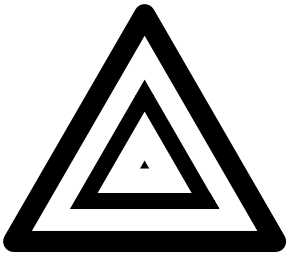
`pencircle scaled` *numeric expression*;

You can create an elliptically shaped and rotated pen by transforming the circular pen. An example:



```
beginfig(1);
pickup pencircle xscaled 2bp yscaled 0.25bp
  rotated 60 withcolor red;
for i=10 downto 1:
  draw 5(i,0)..5(0,i)..5(-i,0)
       ..5(0,-i+1)..5(i-1,0);
endfor;
endfig;
end;
```

In the following example we define a triangular shaped pen. It can be used to plot data points as triangles instead of dots. For comparison we draw a large triangle with both the triangular and the default circular pen.



```
beginfig(1);
path p; p := dir(-30)--dir(90)--dir(210)--cycle;
pen pentriangle;
pentriangle := makepen(p);
draw origin withpen pentriangle scaled 2;
draw (p scaled 1cm) withpen pentriangle scaled 4;
draw (p scaled 2cm) withpen pencircle scaled 8;
endfig;
end;
```

Setting Drawing Options
The function `drawoptions` allows you to change the default settings for drawing. For example, if you specify

```
drawoptions(dashed evenly withcolor red);
```

then all `draw` statements produce dashed lines in red colour, unless you overrule the drawing setting explicitly. To turn off `drawoptions` all together, just give an empty list:

```
drawoptions();
```

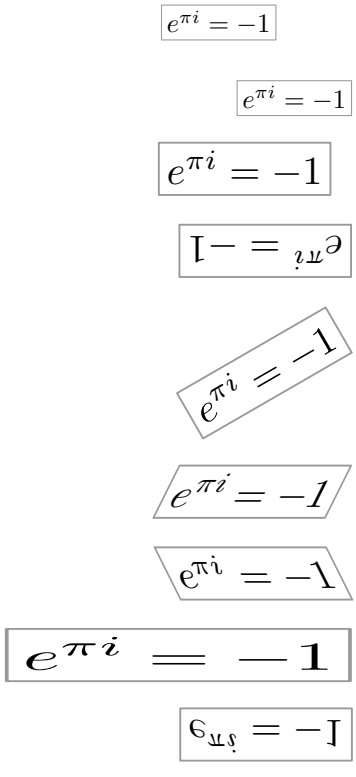
As a matter of fact, this is done automatically by the `beginfig` macro.

Transformations

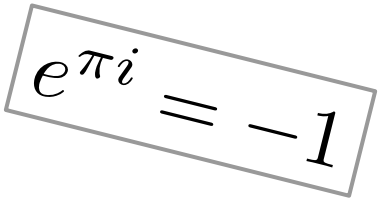
A very characteristic technique with METAPOST, which we applied already in many of the previous examples, is creating a graphic and then using it several times with different transformations. METAPOST has the following built-in operators for scaling, rotating, translating, reflecting, and slanting:

$$
\begin{aligned}
(x,y)\ \texttt{shifted}\ (a,b) &= (x+a, y+b)\,;\\
(x,y)\ \texttt{rotated}\ \theta &= (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)\,;\\
(x,y)\ \texttt{rotatedaround}\ \big((a,b),\theta\big) &= (x\cos\theta - y\sin\theta + a(1-\cos\theta) + b\sin\theta,\\
&\qquad x\sin\theta + y\cos\theta + b(1-\cos\theta) - a\sin\theta)\,;\\
(x,y)\ \texttt{slanted}\ a &= (x+ay, y)\,;\\
(x,y)\ \texttt{scaled}\ a &= (ax, ay)\,;\\
(x,y)\ \texttt{xscaled}\ a &= (ax, y)\,;\\
(x,y)\ \texttt{yscaled}\ a &= (x, ay)\,;\\
(x,y)\ \texttt{zscaled}\ (a,b) &= (ax-by, bx+ay)\,.
\end{aligned}
$$

The effect of the translation and most scaling operations is obvious. The following playful example, in which the formula $e^{\pi i} = -1$ is drawn in various shapes, serves as an illustration of most of the listed transformations.



```
beginfig(1);
pair s; s=(0,-2cm);
def drawit(expr p) =
  draw p shifted s;  s := s shifted (0,-2cm);
enddef;

picture pic;
draw btex $e^{\pi i}=-1$ etex;
draw bbox currentpicture withcolor 0.6white;
pic := currentpicture;
draw pic shifted (1cm, -1cm);
pic := pic scaled 1.5; drawit(pic);
% work with the enlarged base picture

drawit(pic scaled -1);

drawit(pic rotated 30);

drawit(pic slanted 0.5);

drawit(pic slanted -0.5);

drawit(pic xscaled 2);

drawit(pic yscaled -1);
```




```
drawit(pic zscaled (2, -0.5));
endfig;
end;
```

The effect of `rotated` θ is rotation of θ degrees about the origin counter-clockwise. The transformation `rotatedaround(p,`θ`)` rotates θ degrees counter-clockwise around point `p`. Accordingly, it is defined in METAPOST as follows:

```
def rotatedaround(expr p, theta) =  % rotates theta degrees around p
  shifted -p rotated theta shifted p enddef;
```

When you identify a point (x, y) with the 3-vector $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$, each of the above operations is described by an affine matrix. For example, the rotation of θ degrees around the origin counter-clockwise and the translation with (a, b) have the following matrices:

$$\texttt{rotated}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \texttt{translated}(a, b) = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix}.$$

It is easy to verify that

$$\texttt{rotatedaround}\big((a, b), \theta\big) = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -a \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{pmatrix}.$$

The matrix of `zscaled(a,b)` is as follows:

$$\texttt{zscaled}(a, b) = \begin{pmatrix} a & -b & 0 \\ b & a & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Thus, the effect of `zscaled(a,b)` is to rotate and scale so as to map `(1,0)` into `(a,b)`. The operation `zscaled` can also be thought of as multiplication of complex numbers. The picture on the next page illustrates this.

The general form of an affine matrix T is

$$T = \begin{pmatrix} T_{xx} & T_{xy} & T_x \\ T_{yx} & T_{yy} & T_y \\ 0 & 0 & 1 \end{pmatrix}.$$

The corresponding transformation in the two-dimensional space is

$$(x, y) \mapsto (T_{xx}x + T_{xy}y + T_x, T_{yx}x + T_{yy}y + T_y).$$

This mapping is completely determined by the sextuple $(T_x, T_y, T_{xx}, T_{xy}, T_{yx}, T_{yy})$. The information about the mapping can be stored in a variable of data type `transform` and then be applied in a `transformed` statement. There are three ways to define a transform:

□ *In terms of basic transformations*. For example,

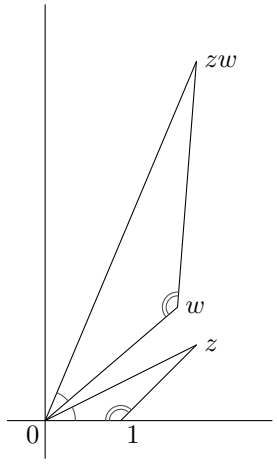
```
transform T; T = identity shifted (-1,0) rotated 60 shifted (1,0);
```

defines the transformation T as a composition of translating with vector $(-1,0)$, rotating around the origin over 60 degrees, and translating with a vector $(1,0)$.

□ *Specifying the sextuple* $(T_x, T_y, T_{xx}, T_{xy}, T_{yx}, T_{yy})$. The six parameters than define a transformation T can be referred to directly as `xpart T`, `ypart T`, `xxpart T`, `xypart T`, `yxpart T`, and `yypart T`. Thus,

```
transform T;
xpart T = ypart T = 1;
xxpart T = yypart T = 0;
xypart T = yxpart T = -1;
```

defines a transformation, viz., the reflection in the line through $(1,0)$ and $(0,1)$.




```
beginfig(1);
pair z; z := (2,1)*cm;
pair w; w := (7/4,3/2)*cm;
pair zw; zw := (z zscaled w) / cm;

draw (-0.5,0)*cm--(3,0)*cm;
draw (0,-0.5)*cm--(0,5.5)*cm;
draw (1,0)*cm--z; draw (0,0)--z; draw (0,0)--w;
draw (0,0)--zw; draw w--zw;

def drawangle(
   expr endofa, endofb, common, length) =
   save tn; tn :=
   turningnumber(common--endofa--endofb--cycle);
   draw (unitvector(endofa-common){(endofa-common)
   rotated (tn*90)} .. unitvector(endofb-common))
   scaled length shifted common withcolor 0.3white;
enddef;
drawangle((1,0)*cm, z, (0,0), 0.4cm);
drawangle(w, zw, (0,0), 0.4cm);
drawangle((0,0), z, (1,0)*cm, 0.2cm);
drawangle((0,0), z, (1,0)*cm, 0.15cm);
drawangle((0,0), zw, w, 0.2cm);
drawangle((0,0), zw, w, 0.15cm);
label.llft(btex $0$ etex,(0cm,0cm));
label.lrt(btex $1$ etex,(1cm,0cm));
label.rt(btex $z$ etex, z);
label.rt(btex $w$ etex, w);
label.rt(btex $zw$ etex, zw);
endfig;

end;
```

□ *Specifying the images of three points*. It is possible to apply an unknown transform to a known pair and use the result in a linear equation. For example,

```
transform T;
(1,0) transformed T = (1,0);
(0,1) transformed T = (0,1);
(0,0) transformed T = (1,1);
```

defines the reflection in the line through $(1,0)$ and $(0,1)$.

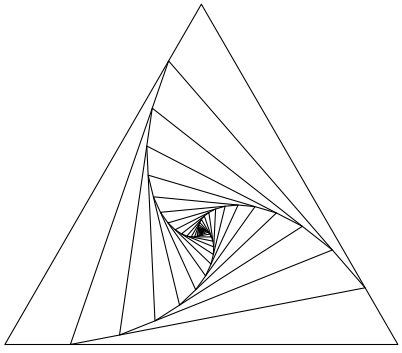
The built-in transformation `reflectedabout(p,q)`, which reflects about the line connecting the points p and q, is defined by a combination of the last two techniques:

```
def reflectedabout(expr p,q) = transformed
  begingroup
    transform T_;
```

```
    p transformed T_ = p;  q transformed T_ = q;
   xxpart T_ = -yypart T_; xypart T_ = yxpart T_; % T_ is a reflection
    T_
  endgroup
enddef;
```

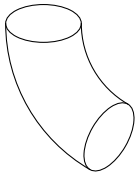
Given a transformation `T`, the inverse transformation is easily defined by `inverse(T)`.

We end with another playful example of an iterative graphic process.



```
beginfig(1);
pair A,B,C; u:=3cm;
A=u*dir(-30); B=u*dir(90); C=u*dir(210);

transform T;
A transformed T = 1/6[A,B];
B transformed T = 1/6[B,C];
C transformed T = 1/6[C,A];

path p; p = A--B--C--cycle;
for i=0 upto 60:
  draw p; p:= p transformed T;
endfor;
endfig;

end;
```

EXERCISE 24 Using transformations, construct the following picture:

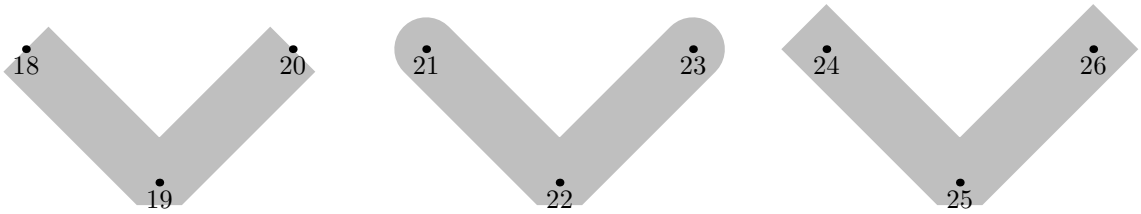


Advanced Graphics

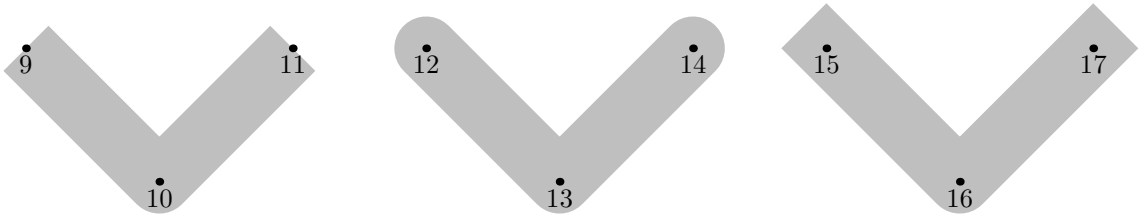
In this section we deal with fine points of drawing lines and with more advanced graphics. This will allow you to create more professional-looking graphics and more complicated pictures.

Joining Lines

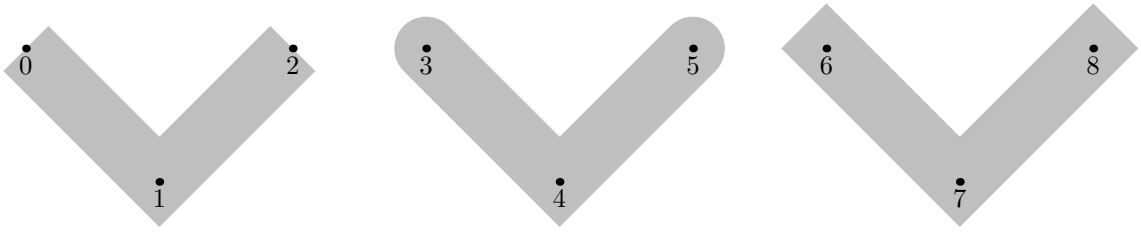
In the last example of the section on pen styles you may have noticed that lines are joined by default such that line joints are normally rounded. You can influence the appearances of the lines by the two internal variables `linejoin` and `linecap`. The picture below shows the possibilities.



linejoin=beveled linecap=butt | linejoin=beveled linecap=rounded | linejoin=beveled linecap=squared



linejoin=rounded linecap=butt | linejoin=rounded linecap=rounded | linejoin=rounded linecap=squared



linejoin=mitered linecap=butt | linejoin=mitered linecap=rounded | linejoin=mitered linecap=squared

This picture can be produced by the following METAPOST code

```
beginfig(1);
for i=0 upto 2:
  for j=0 upto 2:
    z[3i+9j]=(150i, 150j);
    z[3i+9j+1]=(150i+50,150j-50);
    z[3i+9j+2]=(150i+100,150j);
  endfor;
endfor;
drawoptions(withpen pencircle scaled 24 withcolor 0.75white);
linejoin := mitered; linecap := butt;    draw z0--z1--z2;
linejoin := mitered; linecap := rounded; draw z3--z4--z5;
linejoin := mitered; linecap := squared; draw z6--z7--z8;
linejoin := rounded; linecap := butt;    draw z9--z10--z11;
linejoin := rounded; linecap := rounded; draw z12--z13--z14;
```

```
linejoin := rounded; linecap := squared; draw z15--z16--z17;
linejoin := beveled; linecap := butt;    draw z18--z19--z20;
linejoin := beveled; linecap := rounded; draw z21--z22--z23;
linejoin := beveled; linecap := squared; draw z24--z25--z26;
%
drawoptions();
for i=0 upto 26: dotlabel.bot(decimal(i), z[i]); endfor;
labeloffset := 25pt; label.bot("linejoin=mitered", z1);
labeloffset := 40pt; label.bot("linecap=butt",     z1);
labeloffset := 25pt; label.bot("linejoin=mitered", z4);
labeloffset := 40pt; label.bot("linecap=rounded",  z4);
labeloffset := 25pt; label.bot("linejoin=mitered", z7);
labeloffset := 40pt; label.bot("linecap=squared",  z7);
%
labeloffset := 25pt; label.bot("linejoin=rounded", z10);
labeloffset := 40pt; label.bot("linecap=butt",     z10);
labeloffset := 25pt; label.bot("linejoin=rounded", z13);
labeloffset := 40pt; label.bot("linecap=rounded",  z13);
labeloffset := 25pt; label.bot("linejoin=rounded", z16);
labeloffset := 40pt; label.bot("linecap=squared",  z16);
%
labeloffset := 25pt; label.bot("linejoin=beveled", z19);
labeloffset := 40pt; label.bot("linecap=butt",     z19);
labeloffset := 25pt; label.bot("linejoin=beveled", z22);
labeloffset := 40pt; label.bot("linecap=rounded",  z22);
labeloffset := 25pt; label.bot("linejoin=beveled", z25);
labeloffset := 40pt; label.bot("linecap=squared",  z25);
%
endfig;
end;
```

By setting the variable `miterlimit`, you can influence the mitering of joints. The next example demonstrates that the value of this variable acts as a trigger, i.e. when the value of miterlimit gets smaller than some threshhold value, then the mitered join is replaced by a beveled value.

```
beginfig(1);
for i=0 upto 2:
  z[3i]=(150i,0);  z[3i+1]=(150i+50,-50); z[3i+2]=(150i+100,0);
endfor;
drawoptions(withpen pencircle scaled 24pt);
labeloffset:= 25pt;
linejoin := mitered; linecap:=butt;
for i=0 upto 2:
  miterlimit := i;
  draw z[3i]--z[3i+1]--z[3i+2];
  label.bot("miterlimit=" & decimal(miterlimit), z[3i+1]);
endfor;
endfig;
end;
```

miterlimit=0 miterlimit=1 miterlimit=2

Building Cycles

In previous examples you have seen that intersection points of straight lines can be specified by linear equations. A more direct way to deal with path intersection is via the operator `intersectionpoint`. So, given four points `z1`, `z3`, `z3`, and `z4` in general position, you can specify the intersection point `z5` of the line between `z1` and `z2` and the line between `z3` and `z4` by

```
z5 = z1--z2 intersectionpoint z3--z4;
```

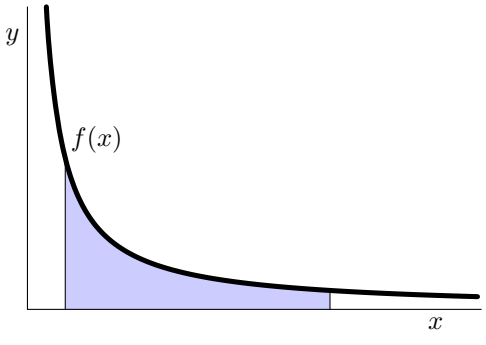
You do not need to rely on setting up linear equations with

```
z5 = whatever[z1,z2] = whatever[z3,z4];
```

The strength of the `intersection` operator is that it also works for curved lines. We use this operator in the next example of a filled area beneath the graph of a function. The closed curve that forms the border of the filled area is constructed with the `buildcycle` command. When given two or more paths, the `buildcycle` macro tries to piece them together so as to form a cyclic path. In case there are more intersection points between paths, the general rule is that

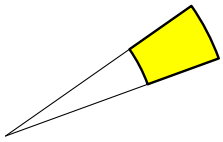
$$\texttt{buildcycle}(p_1, p_2, \ldots, p_n)$$

chooses the intersection between each p_i and p_{i+1} to be as late as possible on the path p_i and as early as possible on p_{i+1}. In practice, it is more convenient to choose the path arguments such that consecutive ones have a unique intersection.




```
beginfig(1);
numeric xmin, xmax, ymin, ymax;
xmin := 1/4; xmax := 6; ymax := 1/xmin; u := 1cm;
% compute the graph of the function
vardef f(expr x) = 1/x enddef;
xinc := 0.1;
path pts_f;
pts_f := (xmin,f(xmin))*u
    for x=xmin+xinc step xinc until xmax:
      .. (x,f(x))*u
    endfor;
path hline[], vline[];
hline0 = (0,0)*u -- (xmax,0)*u;
vline0 = (0,0)*u -- (0,ymax)*u;
vline0.5 = (0.5,0)*u -- (0.5,ymax)*u;
vline4 = (4,0)*u -- (4,ymax)*u;
fill buildcycle(hline0, vline0.5, pts_f, vline4)
  withcolor 0.8[blue,white];
draw hline0; draw vline0; % draw axes
draw (0.5,0)*u -- vline0.5 intersectionpoint pts_f;
draw (4,0)*u -- vline4 intersectionpoint pts_f;
draw pts_f withpen pencircle scaled 2;
label.bot(btex $x$ etex, (0.9xmax,0)*u);
label.lft(btex $y$ etex, (0,0.9ymax)*u);
label.urt(btex $f(x)$ etex, (0.5,f(0.5))*u);
endfig;

end;
```

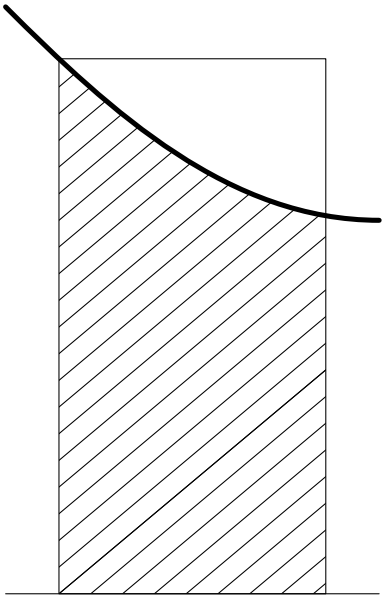
EXERCISE 25 Create the following picture:



Clipping
Clipping is a process to select just those parts of a picture that lie inside an area that is determined by a cyclic path and to discard the portion outside this is area. The command to do this in METAPOST is

`clip` *picture variable* `to` *path expression*;

You can use it to shade a picture element:



```
beginfig(1);
pair p[]; path c[];
c0 = -500*dir(40) -- 500*dir(40);
for i=0 upto 25:
  draw c0 shifted (0,10*i);
  draw c0 shifted (0,-10*i);
endfor;
p1 = (100,0);
c1 = (-20,0) -- (120,0);
c2 = p1--(100,infinity);
c3 = (-20,220){dir(-45)}..(120,140){right};
c4 = (0,0)--(0,infinity);
c5 = buildcycle(c1,c2,c3,c4);
clip currentpicture to c5;
p3 = c4 intersectionpoint c3;
p2 = (100, ypart p3);
draw (0,0)--p1--p2--p3--cycle;
draw c1; draw c3 withpen pencircle scaled 2;
endfig;

end;
```

Dealing with Paths Parametrically
In METAPOST, a path is a continuous curve that is composed of a chain of segments. Each segment is a cubic Bézier curve, which is determined by 4 control points. The points on the curved segment from points p_0 to p_1 with post control point c_0 and pre control point c_1 are determined by the formula

$$p(t) = (1-t)^3 p_0 + 3t(1-t)^2 c_0 + 3t^2(1-t)c_1 + t^3 p_1\,,$$

where $t \in [0,1]$. If the path consists of two arcs, i.e., consists of three points p_0, p_1, and p_2, then the time parameter t runs from 0 to 2. If the path consists of n curve segments, then t runs normally from 0 to n. At $t = 0$ it starts at point p_0 and at intermediate time $t = 1$ the second point p_1 is reached; a third point p_2 in the path, if present, is reached at $t = 2$, and so on. You can get the point on a *path* at any time t with the construction

`point` t `of` *path*;

For a cyclic path with n arcs through the points p_0, $p_1, \ldots, p_{n-1}$, the normal parameter range is $0 \leqslant t < n$, but `point` t `of` *path* can be computed for any t by first

reducing t modulo n. The number of arcs in a path is available through

`length(`*path*`);`

The correspondence between the time parameter and a point on the curve is also used to create a subpath of the curve. The command has the following syntax

`subpath` *pair expression* `of` *path expression*`;`

If the value of the pair expression is `(t1,t2)` and the path expression equals `p`, then the result is a path that follows `p` from `point t1 of p` to `point t2 of p`. If `t1` $>$ `t2`, then the subpath runs backward along `p`.

Based on the `subpath` operation are the binary operators `cutbefore` and `cutafter`. For intersecting paths `p1` and `p2`,

```
p1 cutbefore p2;
```

is equivalent to

```
subpath(xpart(p1 intersectiontimes p2), length(p1)) of p1;
```

except that it also sets the path variable `cuttings` to the parts of `p1` that gets cut off. With multiple intersections, it tries to cut off as little as possible. Similarly,

```
p1 cutafter p2;
```

tries to cut off the part of `p1` after its last intersection with `p2`.

We have seen that for a time parameter `t` we can find the corresponding point on the curve `p` by the statement `point t of p;` Another statement, of the general form

`direction t of` *path*`;`

allows you to obtain a direction vector at the point of the *path* that corresponds with time `t`. The magnitude of the direction vector is somewhat arbitrary. The `directiontime` operation is the inverse of the `direction of` operation. Given a direction vector (a pair) and a path,

`directiontime` *direction vector* `of` *path*`;`

return a numeric value that gives the first time `t` when the path has the indicated direction.

`directionpoint` *direction vector* `of` *path*`;`

returns the first point on the path where the given direction is achieved.

The more familiar concept of arc length is also provided for in METAPOST:

`arclength(`*path*`);`

returns the arc length of the given path. If `p` is a path and `a` is a number between 0 and `arclength(p)`, then

```
arctime a of p;
```

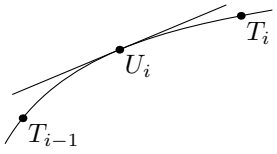
gives the time `t` such that

```
arclength(subpath(0,t) of p) = a;
```

A summary of the path operators is listed in the table below:

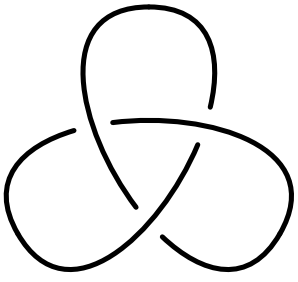
Name	*Arguments and Result*			*Meaning*
	left	right	result	
`arclength`	–	path	numeric	arc length of a path.
`arctime of`	numeric	path	numeric	time on a path where arc length from the start reaches a given value.
`cutafter`	path	path	path	left argument with part after the intersection dropped.
`cutbefore`	path	path	path	left argument with part before the intersection dropped.
`direction of`	numeric	path	path	the direction of a path at a given time.
`directionpoint of`	pair	path	pair	point where a path has a given direction.
`directiontime of`	pair	path	numeric	time when a path has a given direction.
`intersectionpoint`	path	path	pair	an intersection point.
`intersectiontimes`	path	path	numeric	times (t_1, t_2) on paths p_1 and p_2 when the paths intersect.
`length`	–	path	numeric	number of arcs in a path.
`point of`	numeric	path	pair	point on a path given a time value.
`subpath`	pair	path	path	portion of a path for given range of time values times.

Let us apply what we have learned in this subsection to a couple of examples. In the first example, we draw a tangent line at a point of a curve.



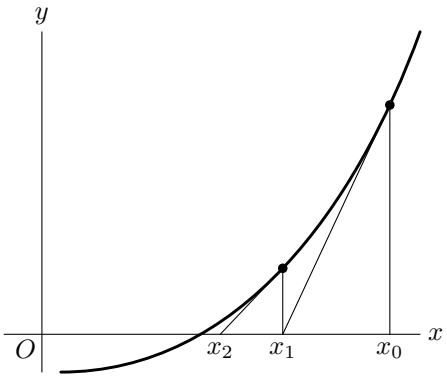

```
beginfig(1);
path c;
c = (0,0){dir(60)}..(100,50){dir(10)};
pair p[];
p0 = point(0.1) of c;
p1 = point(0.5) of c;
p2 = point(0.9) of c;
p3 = direction(0.5) of c;
dotlabel.lrt(btex $T_{i-1}$ etex, p0);
dotlabel.lrt(btex $U_i$ etex, p1);
dotlabel.lrt(btex $T_i$ etex, p2);
draw c;
draw (-p3 -- p3) scaled (40/xpart(p3))
  shifted p1;
endfig;
end;
```

The second example is the trefoil knot, i.e, the torusknot of type (1,3). The METAPOST code has been written such that assigning $n = 5$ and $n = 7$ draws the torusknot of type (1,5) and (1,7), respectively, in a nice way.



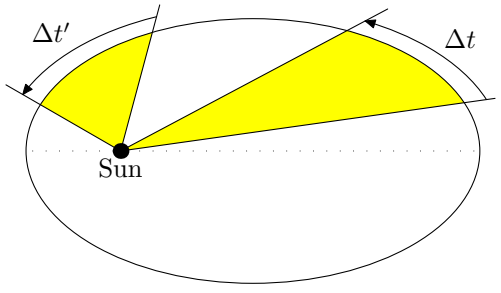
```
beginfig(1);
pair A,B; path p[];
numeric n; n:=3;
A = (0,2cm); B = A rotated (2*360/n);
p0 = A{dir(180)}..tension((n+1)/2)..
   B{dir(180+2*360/n)};
numeric a;
(a,whatever) = p0 intersectiontimes
   (p0 rotated (360/n));
p1 = subpath(0,a-.04) of p0;
p2 = subpath(a+.04,1) of p0;
drawoptions(withpen pencircle scaled 2);
for i=0 upto n-1:
  draw p1 rotated (i*360/n);
  draw p2 rotated (i*360/n);
endfor;
endfig;
end;
```

The third example shows a few steps in the Newton iterative method of finding zeros.




```
beginfig(1);
u := 1cm;
draw (-.5u,0)--(5u,0);
draw (0,-.5u)..(0,4u);
label.llft(btex $0$ etex, (0,0));
label.rt(btex $x$ etex, (5u,0));
label.top(btex $y$ etex, (0,4u));

path f;
f = (.25u,-.5u){right}..(5u,4u){dir(70)};
draw f withpen pencircle scaled 1.2pt;
x0 = 4.6*u;
numeric t[];
for i=0 upto 1:
  (t[i],whatever) = f intersectiontimes
    ((x[i],-infinity)--(x[i],infinity));
  z[i] = point t[i] of f;
  (x[i+1],0) = z[i] +
           whatever*direction t[i] of f;
  draw (x[i],0)--z[i]--(x[i+1],0);
  fill fullcircle scaled 3.6pt shifted z[i];
endfor;
label.bot(btex $x_0$ etex, (x0,0));
label.bot(btex $x_1$ etex, (x1,0));
label.bot(btex $x_2$ etex, (x2,0));
endfig;
end;
```

EXERCISE 26 Create the following picture, which has to do with Kepler's law of areas.



Control Structures

In this section we shall look at two commonly used control structures of imperative programming languages: condition and repetition.

Conditional Operations

The basic form of a conditional statement is

`if` *condition*`:` *balanced tokens* `else:` *balanced tokens* `fi`

where *condition* represents an expression that evaluates to a boolean value (i.e., true or false) and the *balanced tokens* normally represent any valid METAPOST statement or sequence of statements separated by semicolons. The `if` statement is used for branching: depending on some condition, one sequence of METAPOST statements is executed or another. The keyword `fi` is the reverse of `if` and marks the end of the conditional statement. The keyword `fi` separates the statement sequence of the preceding command clause so that the semicolon at the end of the last command in the `else:` part can be omitted. It also marks the end of the

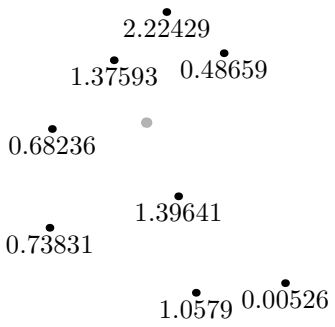
conditional statement so that you do not need a semicolon after the keyword to separate the conditional statement from then next statement.

Other forms of conditional statements are obtained from the basic form by:

- Omitting the `else:` part when there is nothing to be said.
- Nesting of conditional operations. The following shortcut can be used: `else: if` can be replaced by `elseif`, in which case the corresponding `fi` must be omitted. For example, nesting of two basic `if` operations looks as follows:

  `if` *1st condition*`:` *1st tokens* `elseif` *2nd condition*`:` *2nd tokens* `fi`

Let us give an example: computing the centre of gravity (also called barycentre) of a number of objects with randomly generated weight and position. The example contains many more programming constructs, some of which will be covered in sections later on in the tutorial; so you may ignore them if you wish. The nested conditional statement is easily found in the METAPOST code below. With the commands `numeric(x)` and `pair(x)` we test whether `x` is a number or a point, respectively.




```
beginfig(1);
vardef centerofgravity(text t) =
  save x, wght, G, X;
  pair G,X; numeric wght, w;
  G := origin; wght:=0;
  for x=t:
    if numeric(x):
       show("weight = "& decimal(x));
       G:= G + x*X;
       wght := wght + x;
    elseif pair(x):
       show("location = (" &
         decimal(xpart(x)) & ", " &
         decimal(ypart(x)) & ")");
       X:=x; % store pair
    else:
      errmessage("should not happen");
    fi;
  endfor;
  G/wght
enddef;

numeric w[]; pair A[];
n:=8;
for i=1 upto n:
  A[i] = 1.5cm*
    (normaldeviate, normaldeviate);
  w[i] = abs(normaldeviate);
  dotlabel.bot(decimal(w[i]), A[i]);
endfor;
draw centerofgravity(A[1],w[1]
  for i=2 upto n: ,A[i],w[i] endfor)
  withpen pencircle scaled 4bp
  withcolor 0.7white;
endfig;

end;
```

The `errmessage` command is for displaying an error message if something goes wrong and interrupting the program at this point. The `show` statement is used here for debugging purposes. When you run the METAPOST program from a shell, `show` puts its results on the standard output device. In our example, the shell window looked like:

```
(heck@remote 1) mpost barycenter
This is MetaPost, Version 0.641 (Web2C 7.3.1)
(barycenter.mp
>> "location = (-13.14597, -80.09227)"
>> "weight = 1.0579"
>> "location = (-19.7488, -43.93861)"
>> "weight = 1.39641"
>> "location = (-43.89838, 7.07126)"
>> "weight = 1.37593"
>> "location = (-2.69252, 9.70473)"
>> "weight = 0.48659"
>> "location = (-24.17944, 25.14096)"
>> "weight = 2.22429"
>> "location = (-67.98569, -55.73247)"
>> "weight = 0.73831"
>> "location = (20.28859, -76.48691)"
>> "weight = 0.00526"
>> "location = (-67.07672, -18.69904)"
>> "weight = 0.68236" [1] )
1 output file written: barycenter.1
Transcript written on barycenter.log.
(heck@remote 2)
```

The boolean expression that forms the condition can be built up with the following relational and logical operators.

Relational Operators	
Operator	*Meaning*
=	equal
<>	unequal
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
Logical Operators	
Operator	*Meaning*
`and`	test if all conditions hold
`or`	test if one of many conditions hold
`not`	negation of condition

One final remark on the use of semicolons in the conditional statement. Where the colons after the `if` and `else` part are obligatory, semicolons are optional, depending on the context. For example, the statement

```
if cycle(p): fill p;
elseif path(p): draw p;
else: errmessage("what?");
fi;
```

fills or draws a path depending on the path `p` being cyclic or not. You may omit whatever semicolon in this example and rewrite it even as

```
if cycle(p): fill p
elseif path(p): draw p
else: errmessage("what?")
fi
```

However, when you use the conditional clause to build up a single statement, then you must be more careful with placing or omitting semicolons. In

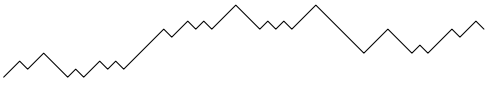
```
draw p withcolor if cycle(p): red else: blue fi withpen pensquare;
```

you cannot add semicolons after the colour specifications, nor omit the final semicolon that marks the end of the statement (unless it is a statement that is recognised as finished because of another keyword, e.g., `endfor`).

Repetition
Numerous examples in previous section have used the `for` loop of the form

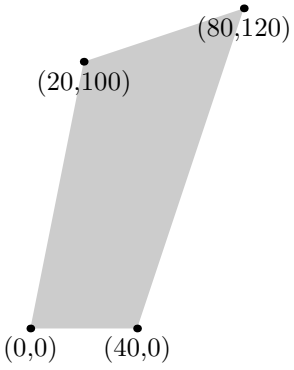
```
for counter = start step stepsize until finish :
    loop text
endfor;
```

where *counter* is a counting variable with initial value *start*. The counter is incremented at each step in the repetition by the value of *stepsize* until it passes the value of *finish*. Then the repetition stops and METAPOST continues with what comes after the `endfor` part. The loop text is usually any valid METAPOST statement or sequence of statements separated by semicolons that are executed at each step in the repetition. Instead of `step 1 until`, we can also use the keyword `upto`. `downto` is another word for `step -1 until`. This counted `for` loop is an example of an *unconditional repetition*, in which a predetermined set of actions are carried out. Below, we give another example of a counted `for` loop: generating a Bernoulli walk. We use the `normaldeviate` operator to generate a random number with the standard normal distribution (mean 0 and standard deviation 1).



```
beginfig(1);
n := 60; pair p[]; p0 = (0,0);
for i=1 upto n:
  p[i] = p[i-1] +
  (3,if normaldeviate>0: -3 else: 3 fi);
endfor;
draw p0 for i=1 upto n: --p[i] endfor;
endfig;
end;
```

The last example in the previous section, in which we computed the centre of gravity of randomly generated weighted points, contained another unconditional repetition, viz., the `for` loop over a sequence of zero or more expressions separated by commas. Another example of this kind is:

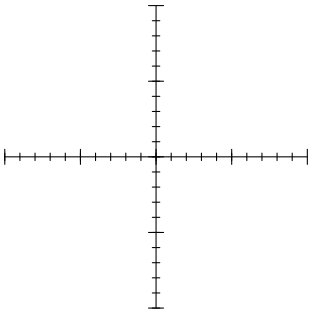



```
beginfig(1);
fill for p=(0,0),(40,0),(80,120),(20,100):
  p-- endfor cycle withcolor 0.8white;
for p=(0,0),(40,0),(80,120),(20,100):
  dotlabel.bot("(" & decimal(xpart(p)) &
    "," & decimal(ypart(p)) & ")", p);
endfor;
endfig;

end;
```

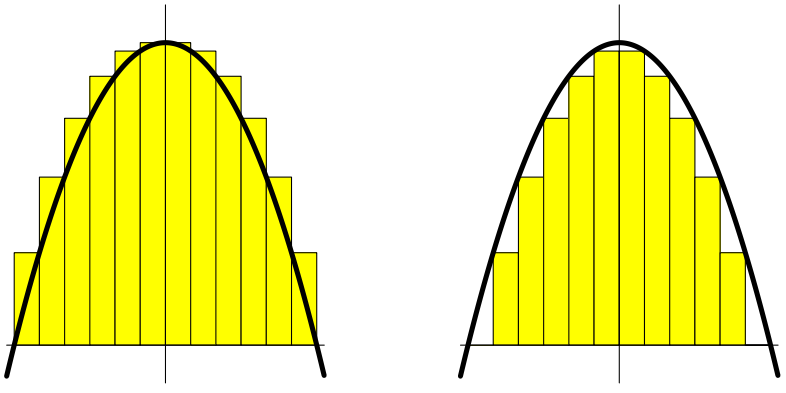
Nesting of counted `for` loops is of course possible, but there are no abbreviations: balancing with respect to `for` and `endfor` is obligatory. You must use both `endfor` keywords in

```
for i=0 upto 10:
  for j=0 upto 10:
    show("i = " & decimal(i) & ", j = " & decimal(j));
  endfor
endfor
```

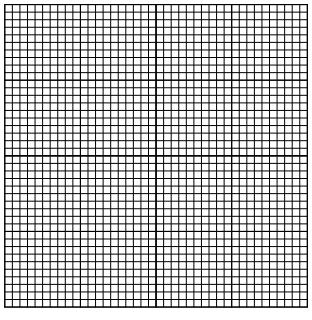
EXERCISE 27 Create the following picture:



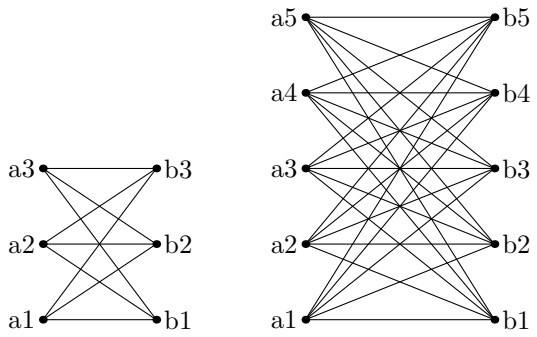
EXERCISE 28 Create the picture below, which illustrates the upper and lower Riemann sum for the area enclosed by the horizontal axis and the graph of the function $f(x) = 4-x^2$.



EXERCISE 29 Create the following piece of millimetre paper.



EXERCISE 30 A graph is bipartite when its vertices can be partitioned into two disjoint sets A and B such that each of its edges has one endpoint in A and the other in B. The most famous bipartite graph is $K_{3,3}$ show below to the left. Write a program that draws the $K_{n,n}$ graph for any natural number $n > 1$. Show that your program indeed creates the graph $K_{5,5}$, which is shown below to the right.



Another popular type of repetition is the *conditional loop*. METAPOST does not have a pre- or post conditional loop (`while` loop or `until` loop) built in. You must create one by an endless loop and an explicit jump outside this loop. First the endless loop: this is created by

`forever:` *loop text* `endfor;`

To terminate such a loop when a boolean condition becomes true, use an exit clause:

`exitif` *boolean expression*`;`

When the exit clause is encountered, METAPOST evaluates the boolean expression and exits the current loop if the expression is true. Thus, METAPOST's version of a `until` loop is:

```
forever:
    loop text;
    exitif boolean expression;
endfor;
```

If it is more convenient to exit the loop when an expression becomes false, then use the predefined macro `exitunless` Thus, METAPOST's version of a `while` loop is:

```
forever: exitunless boolean expression;
    loop text
endfor
```

Macros

Defining Macros

In the section about the repetition control structure we introduced `upto` as a shortcut of `step 1 until`. This is also how it is is internally defined in METAPOST:

```
def upto = step 1 until enddef;
```

It is a definition of the form

`def` *name* = *replacement text* `enddef;`

It calls for a macro substitution of the simplest kind: subsequent occurrences of the token *name* will be replaced by the *replacement text*. The name in a macro is a variable name; the replacement text is arbitrary and may for example consist of a sequence of statements separated by semicolons.

It is also possible to define macros with arguments, so that the replacement text can be different for different calls of the macro. An example of a built-in, parametrised macro is:

```
def rotatedaround(expr z, d) =    % rotates d degrees around z
  shifted -z rotated d shifted z
enddef;
```

Although it looks like a function call, a use of `rotatedaround` expands into in-line code. The `expr` in this definition means that a formal parameter (here `z` or `d`) can be an arbitrary expression. Each occurrence of a formal parameter will be replaced by the corresponding actual argument (this is referred to as 'call-by-value'). Thus the line

```
rotatedaround(p+q, a+b);
```

will be replaced by the line

```
shifted -(p+q) rotated (a+b) shifted (p+q);
```

Macro parameters need not always be expressions. Another argument type is `text`, indicating that the parameters are just past as an arbitrary sequence of tokens.

Grouping and Local Variables

In METAPOST, all variables are global by default. But you may want to use in some piece of METAPOST code a variable that has temporarily inside that portion of code a value different from the one outside the program block. The general form of a program block is

`begingroup` *statements* `endgroup`

where *statements* is a sequence of one or more statements separated by semicolons. For example, the following piece of code

```
x := 1; y := 2;
begingroup x:=3; y:=x+1; show(x,y); endgroup
show(x,y);
```

will reveal that the `x` and `y` values are 3 and 4, respectively, inside the program block. But right after this program block, the values will be 1 and 2 as before.

The program block is used in the definition of the `hide` macro:

```
def hide(text t) = exitif numeric begingroup t; endgroup; enddef;
```

It takes a text parameter and interprets it as a sequence of statements while ultimately producing an empty replacement text. In other words, this command allows you to run code silently.

Grouping often occurs automatically in METAPOST. For example, the `beginfig` macro starts with `begingroup` and the replacement text for `endfig` ends with `endgroup`. `vardef` macros are always grouped automatically, too.

You may want to go one step further: not only treating values of a variable locally, but also its name. For example, in a macro definition you may want to use a so-called *local variable*, i.e., a variable that only has meaning inside that definition and does not interfere with existing variables. In general, variables are made local by the statement

`save` *name sequence*`;`

For example, the macro `whatever` has the replacement text[8]

```
begingroup save ?; ? endgroup
```

This macro returns an unknown. If the `save` statement is used outside of a group, the original values are simply discarded. This explains the following definition of the built-in macro `clearxy`:

[8] in fact, `save` is a `vardef` macro, which has the `begingroup` and `begingroup` automatically placed around the replacement text. Thus, the `begingroup` and `endgroup` are superfluous here.

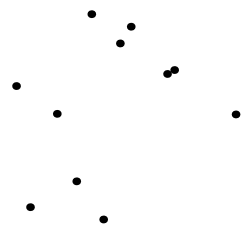
```
def clearxy = save x,y enddef
```

Vardef Definitions
Sometimes we want to return a value from a macro, as if it is a function or subroutine. In this case we want to make sure that the calculations inside the macro do not interfere with the expected use of the macro. This is the main purpose of the `vardef` definition of the form

`def` *name* = *replacement text*; *returned text* `enddef;`

By using `vardef` instead of `def` we hide the replacement text but the last statement, which returns a value.

Below we given an example of a macro that generates a random point in the region `[l,r]`×`[b,u]`. We use the `uniformdeviate` to generate a random number with the uniform distribution between 0 and the given argument. A validity test on the actual arguments is carried out; in case a region is not defined properly, we use `errmessage` to display some text and exit from the macro call, returning the origin as default point when computing is continued.



```
beginfig(1);
vardef randompoint(expr l,r,b,u) =
  if (r<=l) or (u<=b):
     errmessage("not a proper region");
     origin
  else:

  numeric x, y;
  x = l+uniformdeviate(r-l);
  y = b+uniformdeviate(u-b);
  (x,y)
  fi
enddef;
for i=0 upto 10:
  dotlabel("",randompoint(10,100,10,100));
endfor;
endfig;
end;
```

Do not place a semicolon after `origin` or `(x,y)`. In that case, the statement becomes part of the hidden replacement text and an empty value is returned. This causes a runtime error.

Defining the Argument Syntax
In METAPOST, you can explicitly define the argument syntax and construct unary, binary, or tertiary operators. Let us look at the code of a predefined unary operator:

```
vardef unitvector primary z = z/abs z enddef;
```

As the example suggests, the keyword `primary` is enough to specify the macro as a unary `vardef` operator. Other keywords are `secondary` and `tertiary`. The advantage of specifying an n-ary operator is that you do not need to place brackets around arguments in compound statements; METAPOST will sort out which tokens are the arguments. For example

```
unitvector v rotated angle v;
```

is understood to be equivalent to

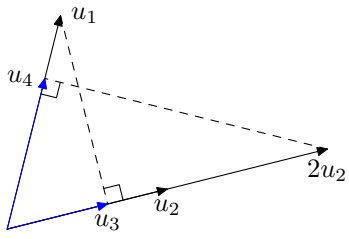
```
(unitvector(v)) rotated(angle(v));
```

You can also define a macro to play the role of an `of` operation. For example, the `direction of` macro is predefined by

```
vardef direction expr t of p =
  postcontrol t of p - precontrol t of p
enddef;
```

Precedence Rules of Binary Operators

METAPOST provides the classifiers `primarydef`, `secondarydef`, and `tertiarydef` (for `def` macros, not for `vardef` macros) to set the level of priority of a binary operator. In the example below, the orthogonal projection of a vector `v` along another vector `w` is defined as a secondary binary operator.

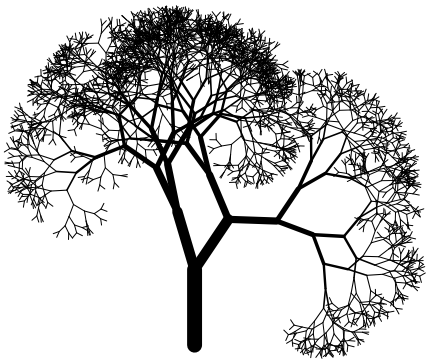



```
beginfig(1);
secondarydef v projectedalong w =
 if pair(v) and pair(w):
    (v dotprod w) / (w dotprod w) * w
 else:
    errmessage "arguments must be vectors"
 fi
enddef;
pair u[]; u1 = (20,80); u2 = (60,15);
drawarrow origin--u1;
drawarrow origin--u2;
drawarrow origin--2*u2;
u3 = u1 projectedalong u2;
u4 = 2*u2 projectedalong u1;
drawarrow origin--u3 withcolor blue;
draw u1--u3 dashed withdots;
draw ((1,0)--(1,1)--(0,1))
  zscaled (6pt*unitvector(u2)) shifted u3;
drawarrow origin--u4 withcolor blue;
draw 2*u2--u4 dashed withdots;
draw ((1,0)--(1,1)--(0,1))
  zscaled (6pt*unitvector(-u1)) shifted u4;
labeloffset := 4pt;
label.rt(btex $u_1$ etex, u1);
label.bot(btex $u_2$ etex, u2);
label.bot(btex $2u_2$ etex, 2*u2);
label.bot(btex $u_3$ etex, u3);
label.lft(btex $u_4$ etex, u4);
endfig;
end;
```

Recursion

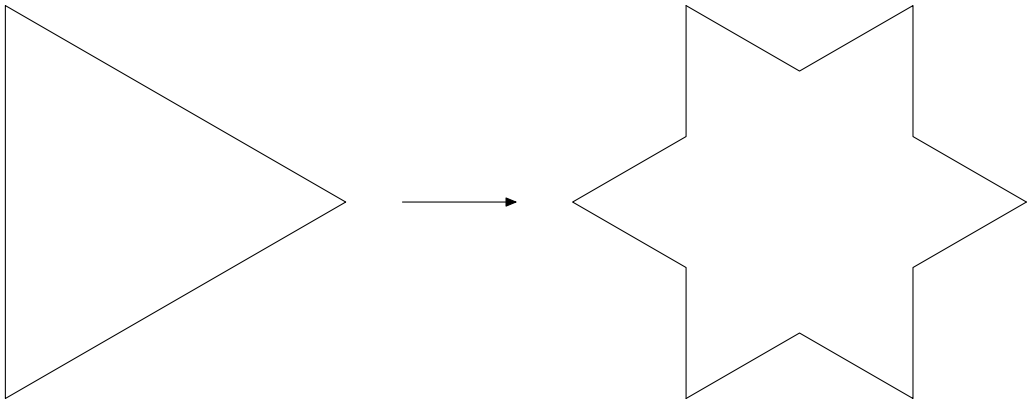
A macro is defined recursively if in its definition, it makes a call to itself. Recursive definition of a macro is possible in METAPOST. We shall illustrate this with the computation of a Pythagorean tree.

```
beginfig(1)
u:=1cm; branchrotation := 60;
offset := 180-branchrotation;
thinning := 0.7;
shortening := 0.8;
def drawit(expr p, linethickness) =
  draw p withpen pencircle scaled linethickness;
enddef;
vardef tree(expr A,B,n,size) =
    save C,D,thickness; pair C,D;
    thickness := size;
```

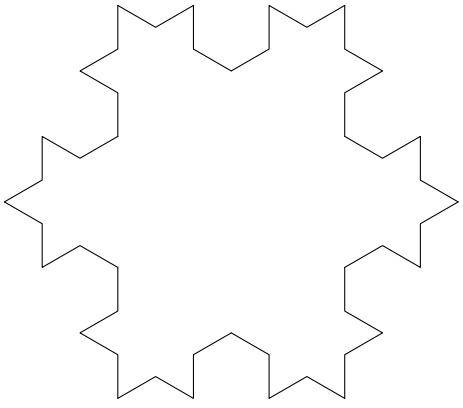


```
    C := shortening[B, A rotatedaround(B,
      offset+uniformdeviate(branchrotation))];
    D := shortening[B, A rotatedaround(B,
      -offset-uniformdeviate(branchrotation))];
    if n>0:
      drawit(A--B, thickness);
      thickness := thinning*thickness;
      tree(B, C, n-1, thickness);
      tree(B, D, n-1, thickness);
    else:
      drawit(A--B,thickness);
      thickness := thinning*thickness;
      drawit(B--C, thickness);
      drawit(B--D, thickness);
    fi;
enddef;
tree((0,0), (0,u), 10, 2mm);
endfig;
end;
```

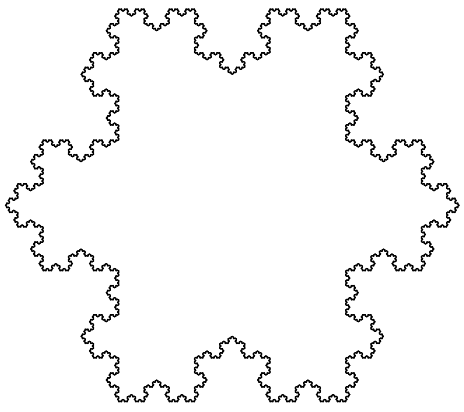
EXERCISE 31 The Koch snowflake is constructed as follows: start with an equilateral triangle. Break each edge into four straight pieces by adding a bump a shown below.



You can then repeat the process of breaking a line segment into four pieces of length one-fourth of the segment that is broken up. Below you see the next iteration.



Write a program that can compute the picture after n iterations. After six iteration, the Koch snowflake should look like



Using Macro Packages

METAPOST comes with built-in macro packages, which are nothing more than files containing macro definitions and constants. The most valuable macro package is `graph`, which contains high-level utility macros for easy drawing graphs and data plots. The `graph` package is loaded by the statement

```
input graph
```

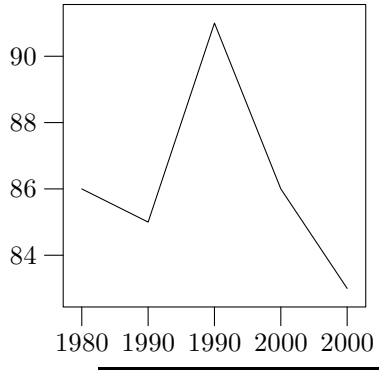
A detailed description, including examples, of the `graph` package can be found in [Hob92b, GRS94]. Here, we just show one example of its use. We represent the following data about the annual beer consumption in the Netherlands in the period 1980–2000 graphically[9].

year	1980	1985	1990	1995	2000
litre	86	85	91	86	83

Suppose that the data are stored columnwise in a file, say `consumption.dat`, as

```
1980 86
1985 85
1990 91
1995 86
2000 83
```

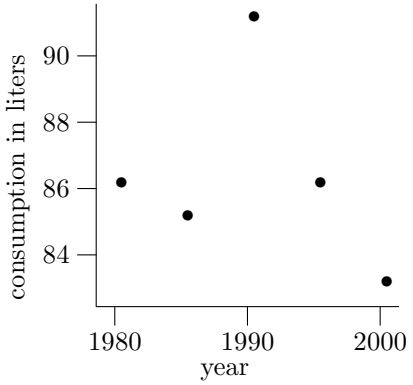
The following piece of code produces a line plot of the data. The only drawback of the `graph` package appears: years are by default marked by decade. This explains the densely populated labels.




```
input graph;
beginfig(1);
draw begingraph(4cm,4cm);
gdraw "consumption.dat";
endgraph;
endfig;

end;
```

[9] Compare this example with your own code in exercise 22.

A few changes in the code draws the data point as bullets, changes the frame style, limits the number of tick marks on the horizontal axis, and puts labels near the axes.




```
beginfig(1);
draw begingraph(4cm,4cm);
gdraw "consumption.dat" plot btex $\bullet$ etex;
Gmarks := 3; % limit number of ticks
frame.llft;
glabel.lft(btex consumption in litres etex
  rotated 90, OUT);
glabel.bot(btex year etex, OUT);
endgraph;
endfig;

end;
```

With the `graph` package, you can easily change in a plot the ticks, the scales, the grid used an/or displayed, the title of the plot, and so on.

Mathematical functions

The following METAPOST code defines some mathematical functions that are not built-in. Note that METAPOST contains two trigonometric functions, `sind` and `cosd`, but they expect their argument in degrees, not in radians.

```
vardef sqr  primary x = (x*x)                                    enddef;
vardef log  primary x = (if x=0: 0 else: mlog(x)/mlog(10) fi) enddef;
vardef ln   primary x = (if x=0: 0 else: mlog(x)/256 fi)      enddef;
vardef exp  primary x = ((mexp 256)**x)                          enddef;
vardef inv  primary x = (if x=0: 0 else: x**-1 fi)            enddef;
vardef pow (expr x,p) = (x**p)                                   enddef;
% trigonometric functions
numeric pi; pi := 3.1415926;
numeric radian;  radian := 180/pi; % 2pi*radian = 360 ;
vardef tand     primary x = (sind(x)/cosd(x))     enddef;
vardef cotd     primary x = (cosd(x)/sind(x))     enddef;
vardef sin      primary x = (sind(x*radian))      enddef;
vardef cos      primary x = (cosd(x*radian))      enddef;
vardef tan      primary x = (sin(x)/cos(x))       enddef;
vardef cot      primary x = (cos(x)/sin(x))       enddef;
% hyperbolic functions
vardef sinh     primary x = save xx ; xx = exp xx ; (xx-1/xx)/2 enddef ;
vardef cosh     primary x = save xx ; xx = exp xx ; (xx+1/xx)/2 enddef ;
vardef tanh     primary x = (sinh(x)/cosh(x))     enddef;
vardef coth     primary x = (cosh(x)/sinh(x))     enddef;
% inverse trigonometric and hyperbolic functions
vardef arcsind  primary x = angle((1+-+x,x))      enddef;
vardef arccosd  primary x = angle((x,1+-+x))      enddef;
vardef arcsin   primary x = ((arcsind(x))/radian) enddef;
vardef arccos   primary x = ((arccosd(x))/radian) enddef;
vardef arccosh  primary x = ln(x+(x+-+1))         enddef;
vardef arcsinh  primary x = ln(x+(x++1))          enddef;
```

Most definitions speak for themselves, except that you may not be familiar with Pythagorean addition (++) and subtraction (+−+):

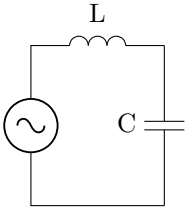
$$++(a,b) = \sqrt{a^2+b^2}, \quad +-+(a,b) = \sqrt{a^2-b^2}.$$

More Examples

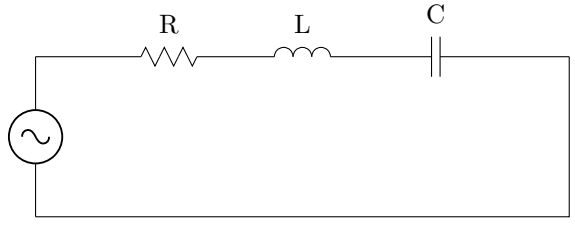
The examples in this section are meant to give you an idea of the strength of METAPOST.

Electronic Circuits

`mpcirc` is a macro package for drawing electronic circuits, developed by Tomasz Cholewo and downloadable from `http://ci.uofl.edu/tom/software/LaTeX/mpcirc/`. Let us use it to create some diagrams. We show the diagrams and the code to create them. The basic idea of `mpcirc` is that you have at your disposal a set of predefined electronic components such as resistor, capacity, diode, and so on. Each component has some connection points, referred to by `a`, `b`, ..., for wires. You place the elements, using predefined orientations, and then connect them with wires. This mode of operating with `mpcirc` is referred to as turtle-based. Another programming style for drawing diagrams is node-based. In this approach, the node locations are determined first and then the elements are put between them using `betw.x` macros. We shall use the turtle-mode in our examples and hope that the comments speak for themselves.

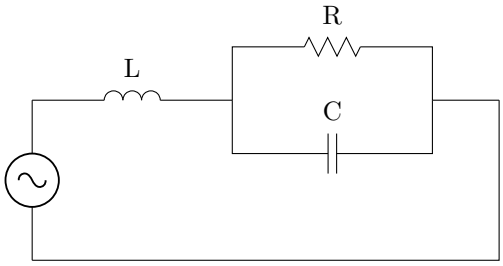



```
u:=10bp; % unit of length
input mpcirc;
beginfig(1);
prepare(L,C,Vac); % mention your elements
z0=(10u,10u);  % lower right node
ht:=6u;        % height of circuit
z1=z0+(0,ht);  % upper right node
C=.5[z0,z1];   % location of capacitor
L.t=T.r;       % use default orientation
C.t=Vac.t=T.u; % components rotated 90 degrees
% set the distance between Voltage and Capacitor
equally_spaced(5u,0) Vac, C;
L=z1-0.5(C-Vac); % location of spool
edraw; % draw components of the circuit
% draw wires connecting components
% the first ones rotated 90 degrees
wire.v(Vac.a,z0);
wire.v(Vac.b,L.a);
wire.v(L.b,z1);
wire(C.a,z0);
wire(C.b,z1);
endfig;
end;
```




```
u:=10bp; % unit of length
input mpcirc;
beginfig(1);
prepare (L,R,C,Vac); % mention your elements
z0=(0,0);      % lower left node
ht:=6u;        % height of circuit
z1=z0+(0,ht);  % upper left node
Vac=.5[z0,z1]; % location of voltage
Vac.t=T.u;     % rotated 90 degrees
L.t=R.t=C.t=T.r; % default orientation
% set equal distances
equally_spaced(5u,0) z1,R,L,C,z2;
```

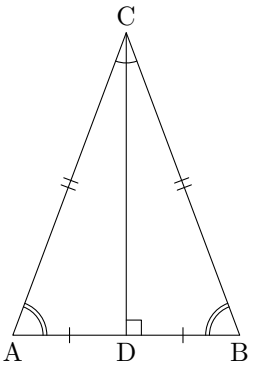
```
edraw; % draw elements of circuits
% draw wires connecting nodes
% the first ones rotated 90 degrees
wire.v(Vac.a,z0);
wire.v(Vac.b,z1);
wire.v(z2,z0);
wire(z1,R.a);
wire(R.b,L.a);
wire(L.b,C.a);
wire(C.b,z2);
endfig;
end;
```




```
u:=10bp; % unit of length
input mpcirc;
beginfig(1);
prepare (L,R,C,Vac); % mention your elements
z0=(0,0);      % lower left node
ht:=6u;        % height of circuit
z1=z0+(0,ht);  % upper left node
Vac=.5[z0,z1]; % location of voltage
Vac.t=T.u;     % rotated 90 degrees
L.t=R.t=C.t=T.r; % default orientation
% set equal distances
equally_spaced(7.5u,0) z1,z2,z3;
L=0.5[z1,z2]; % location of spool
C=0.5[z2,z3]-(0,2u);
R=0.5[z2,z3]+(0,2u);
z4 = z3+(2.5u,0);
edraw; % draw elements of circuits
% draw wires connecting nodes
wire.v(Vac.a,z0);
wire.v(Vac.b,z1);
wire(z1,L.a);
wire(L.b,z2);
wire.v(z2,C.a);
wire.v(z2,R.a);
wire.v(z3,C.b);
wire.v(z3,R.b);
wire(z3,z4);
wire.v(z4,z0);
endfig;
end;
```

Marking Angles and Lines

In geometric pictures, line segments of equal length are often marked by an equal number of ticks and equal angles are often marked the same, too. In the following example, the macros `tick`, `mark_angle`, and `mark_right_angle` mark lines and angles. When dealing with angles, we use the macro `turningnumber` to find the direction of a cyclic path: 1 means counter-clockwise, -1 means clockwise. We use it to make our macros `mark_angle`, and `mark_right_angle` independent of the order in which the non-common points of the angle are specified.

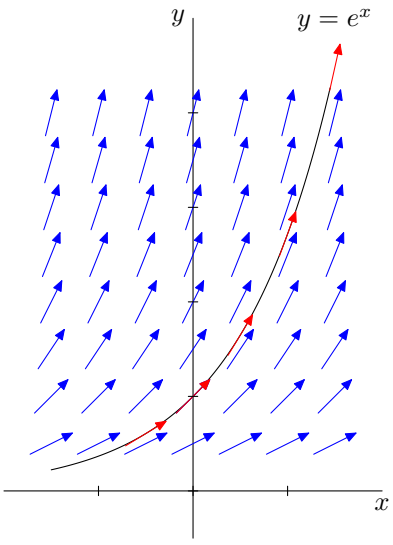



```
% set some user-adjustable constants
angle_radius := 4mm;
angle_delta := 0.5mm;
mark_size := 2mm;

def mark_angle(expr A, common, B, n) =
  % draw 1, 2, 3 or 4 arcs
  draw_angle(A, common, B, angle_radius);
  if n>1: draw_angle(A, common, B,
    angle_radius+angle_delta); fi;
  if n>2: draw_angle(A, common, B,
    angle_radius-angle_delta); fi;
  if n>3: draw_angle(A, common, B,
    angle_radius+2*angle_delta); fi;
enddef;

def draw_angle(expr endofa, common, endofb, r) =
   begingroup
   save tn;
   tn := turningnumber(common--endofa--endofb--cycle);
   draw (unitvector(endofa-common){(endofa-common)
   rotated(tn*90)} .. unitvector(endofb-common))
     scaled r shifted common;
   endgroup
enddef;

def mark_right_angle(expr endofa, common, endofb) =
   begingroup
   save tn; tn :=
   turningnumber(common--endofa--endofb--cycle);
   draw ((1,0)--(1,1)--(0,1)) zscaled(mark_size*
     unitvector((1+tn)*endofa+(1-tn)*endofb-2*common))
     shifted common;
   endgroup
enddef;

def tick(expr p, n) =
   begingroup
   save midpnt;
   midpnt = 0.5*arclength(p);
   % find the time when half-way the path
   for i=-(n-1)/2 upto (n-1)/2:
     draw_mark(p, midpnt+mark_size*i/2);
     % place n tick marks
   endfor;
   endgroup
enddef;

def draw_mark(expr p, m) =
   begingroup
   save t, dm; pair dm;
   t = arctime m of p;
   % find a vector orthogonal to p at time t
   dm = mark_size*unitvector(direction t of p rotated 90);
   draw(-1/2dm..1/2dm) shifted (point t of p);
   % draw tick mark
   endgroup
enddef;

beginfig(1);
pair A, B, C, D;
A := (0,0);  B := (3cm,0);
C := (1.5cm,4cm);  D := (1.5cm,0);
draw A--B--C--cycle;  draw C--D;
% draw triangle and altitude
```

```
label.bot("A", A);  label.bot("B", B);
label.top("C", C); label.bot("D", D);
tick(A--D,1);  tick(D--B,1);  tick(A--C,2);  tick(B--C,2);
mark_angle(C,A,B,2);  mark_angle(A,B,C,2);
mark_angle(B,C,A,1); mark_right_angle(C,D,B);
endfig;

end;
```

Vectorfields

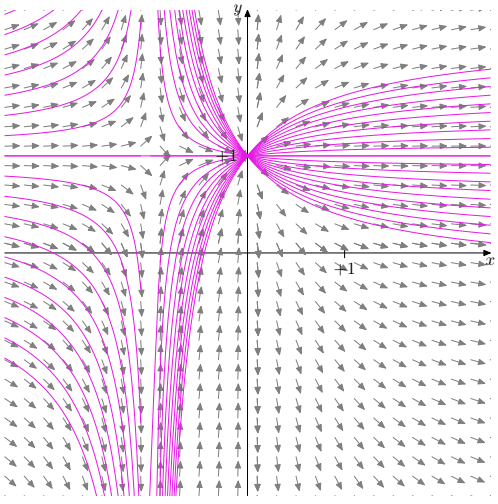
In the first example below we show the directional field corresponding with the ordinary differential equation $y' = y$. For clarity, we show a vectorfield instead of a directional field with small line segments.




```
beginfig(1);
% some constants
numeric xmin, xmax, ymin, ymax, xinc, u;
xmin := -1.5; xmax := 1.5; ymin := 0; ymax := 4.5;
xinc := 0.05; u := 1cm;

% draw axes
draw (xmin-0.5,0)*u -- (xmax+0.5,0)*u;
draw (0,ymin-0.5)*u -- (0,ymax+0.5)*u;

% define f making up the ODE y' = f(x,y).
% Here we take y' = y with the exponential curve
% as solution curve
vardef f(expr x,y) = y enddef;

% define routine to compute function values
def compute_curve(suffix g)(expr xmin, xmax, xinc) =
  ( (xmin,g(xmin))
  for x=xmin+xinc step xinc until xmax: .. (x,g(x)) endfor )
enddef;

% compute and draw exponential curve
vardef exp(expr x) =  (mexp 256)**x enddef;
path p;  p := compute_curve(exp, xmin, xmax, xinc) scaled u;
draw p;

% draw direction field
pair vec; path v;
for x=xmin step 0.5 until xmax:
  for y=ymin+0.5 step 0.5 until ymax-0.5:
    vec := unitvector( (1,f(x,y)) ) scaled 1/2u;
    v := ((0,0)--vec) shifted -1/2vec;
    drawarrow v shifted (x*u,y*u) withcolor blue;
  endfor;
endfor;
% draw directions along the exponential curve
for x=-0.5 step 0.5 until xmax:
  vec := unitvector( (1,f(x,exp(x))) ) scaled 1/2u;
  v := ((0,0)--vec) shifted -1/2vec;
  drawarrow v shifted (x*u,exp(x)*u) withcolor red;
endfor;
% draw ticks and labels
for x=round(xmin) upto xmax:
  draw (x,-0.05)*u--(x,0.05)*u;
endfor;
for y=round(ymin) upto ymax:
  draw (-0.05,y)*u--(0.05,y)*u;
endfor;
label.bot(btex $x$ etex, (xmax+0.5,0)*u);
label.lft(btex $y$ etex, (0,ymax+0.5)*u);
```

```
label(btex $y=e^x$ etex, (xmax, exp(xmax)+0.5)*u);
endfig;
end;
```

Now we shall show some other examples of directional fields corresponding with ODE's and solution curves using the macro package `courbe` from Jean-Michel Sarlat, which we downloaded from `http://melusine.eu.org/syracuse/metapost/courbes/`.

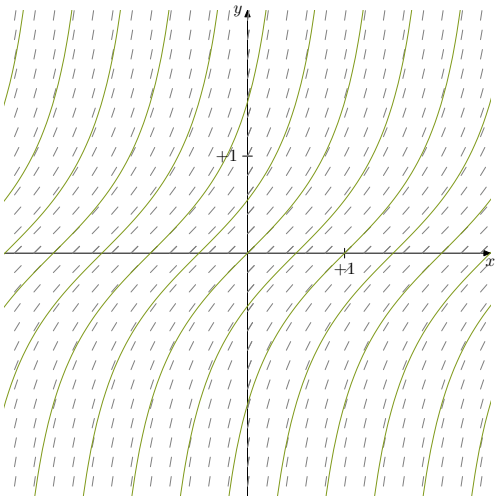


$(x + x^2)y' - y = -1$

```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

input courbes;
vardef fx(expr t) = t enddef;
vardef fy(expr t) = 1+a*t/(1+t) enddef;

beginfig(1);
repere(10cm,10cm,5cm,5cm,2cm,2cm);
trace.axes(0.5pt);
marque.unites(1mm);
%% Champs de vecteurs
vardef F(expr x,y) = (y-1)/(x+x**2) enddef;
champ.vecteurs(0.1,0.1,0.2,0.15,0.5white);
%% Courbes intégrales
color la_couleur;
la_couleur = (0.9,0.1,0.9);
for n = 0 upto 20:
  a := (n/8) - 1.25;
  draw ftrace(-0.995,2.5,50) en_place withcolor la_couleur;
  draw ftrace(-2.5,-1.1,50) en_place withcolor la_couleur;
endfor;
%
draw rpoint(r_xmin,1)--rpoint(r_xmax,1)
  withcolor la_couleur;
decoupe.repere;
etiquette.axes;
etiquette.unites;
label(btex $(x+x^2)y'-y=-1$ etex scaled 2.5,rpoint(0,-3));
endfig;
end;
```

In the next example, we added a macro to the `courbes.mp` package for drawing a directional field with line segments instead of arrows.

```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

input courbes;
% ======================================== fonctions
vardef fx(expr t) = t enddef;
vardef fy(expr t) = tan(t+a) enddef;
% ======================================== figure
```

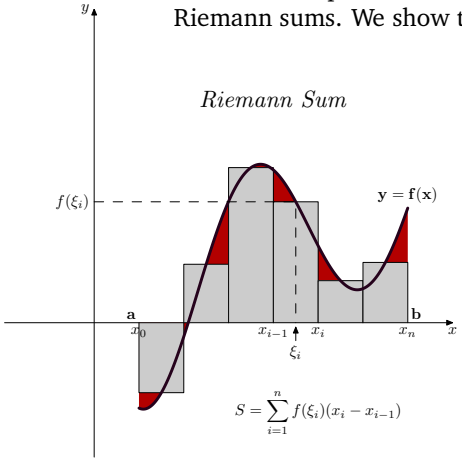


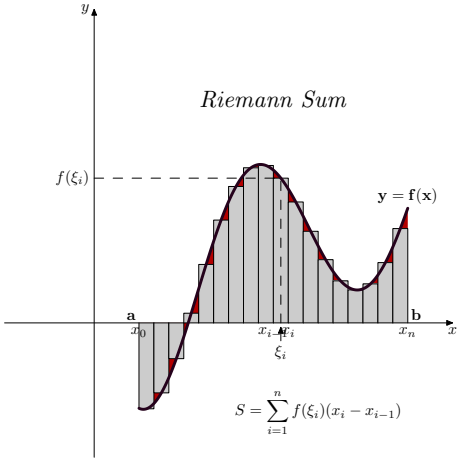
$y' = 1 + y^2$

```
beginfig(1);
repere(10cm,10cm,5cm,5cm,2cm,2cm);
trace.axes(0.5pt);
marque.unites(1mm);

%% Champs de directions
vardef F(expr x,y) = 1+y**2 enddef;
champ.segments(0,0,0.2,0.1,0.5white);

%% Courbes intégrales
for n = 0 upto 16:
  a := (n/2) - 4;
  draw ftrace(-1.5-a,1.5-a,50) en_place
    withcolor (0.5,0.6,0.1);
endfor;

decoupe.repere;
etiquette.axes;
etiquette.unites;
label(btex $y'=1+y^2$ etex scaled 2.5,rpoint(0,-3));
endfig;

end;
```

Riemann Sums

Another example of the `courbes.mp` macro package is the following illustration of Riemann sums. We show two pictures for different number of segments.






```
verbatimtex
%&latex
\documentclass{article}
\everymath{\displaystyle}
\begin{document}
etex

input courbes;
vardef fx(expr t) = t enddef;
vardef fy(expr t) = (t-5)*sin(t)-cos(t)+2 enddef;

beginfig(1);

numeric a,b,n,h;
a = 1;  b = 7; n = 6; h = (b-a)/n;
color aubergine;  aubergine = (37/256,2/256,29/256);
repere(10cm,10cm,2cm,3cm,1cm,1cm);
fill ((a,0)--ftrace(a,b,200)--(b,0)--cycle) en_place
  withcolor 0.7red;

for i=1 upto n:
  path cc;
  aa := a + (i-1) * h;
  bb := aa + h;
  ff := fy(aa + h/2);
  cc := rpoint(aa,0)--rpoint(aa,ff)--rpoint(bb,ff)--
        rpoint(bb,0);
  fill cc--cycle withcolor 0.8white;
  draw cc;
endfor;

trace.axes(0.5pt);
trace.courbe(a,b,200,2pt,aubergine);
decoupe.repere;
etiquette.axes;
```

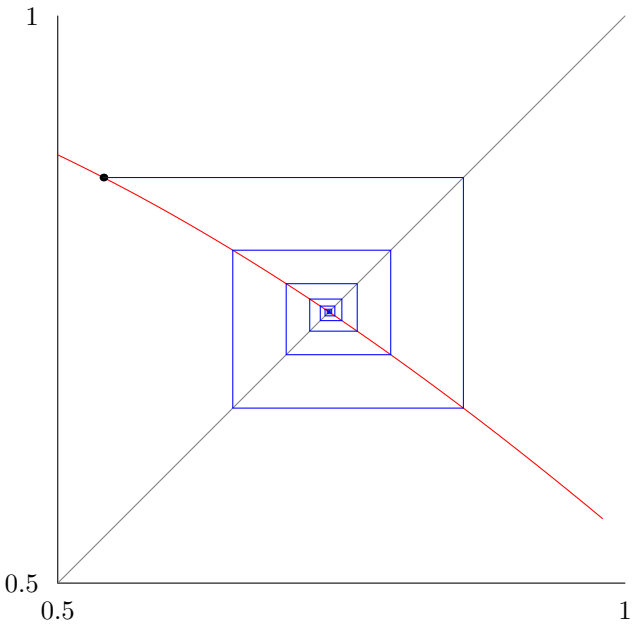
```
label.bot(btex $x_{i-1}$ etex, rpoint(a+n/2*h,0));
label.bot(btex $x_{0}$ etex, rpoint(a,0));
label.bot(btex $x_{n}$ etex, rpoint(b,0));
label.ulft(btex $\mathbf{a}$ etex, rpoint(a,0));
label.urt(btex $\mathbf{b}$ etex, rpoint(b,0));
label.top(btex $\mathbf{y=f(x)}$ etex, f(b) en_place);
label.bot(btex $x_{i}$ etex, rpoint(a+n/2*h+h,0));
projection.axes(f(a+(n+1)/2*h),0.5pt,2);

label.lft(btex $f(\xi_i)$ etex, rpoint(0,fy(a+(n+1)/2*h)));
label.bot(btex $\xi_i$ etex,rpoint(a+(n+1)/2*h,-0.4));
drawarrow rpoint(a+(n+1)/2*h,-0.4)--
  rpoint(a+(n+1)/2*h,-0.1);
label(btex \textit{Riemann Sum} etex scaled 2,rpoint(4,5));
label(btex $S = \sum_{i=1}^n f(\xi_i)(x_i - x_{i-1})$ etex
  scaled 1.5,rpoint(5,-2));
endfig;

end;
```

Iterated Functions

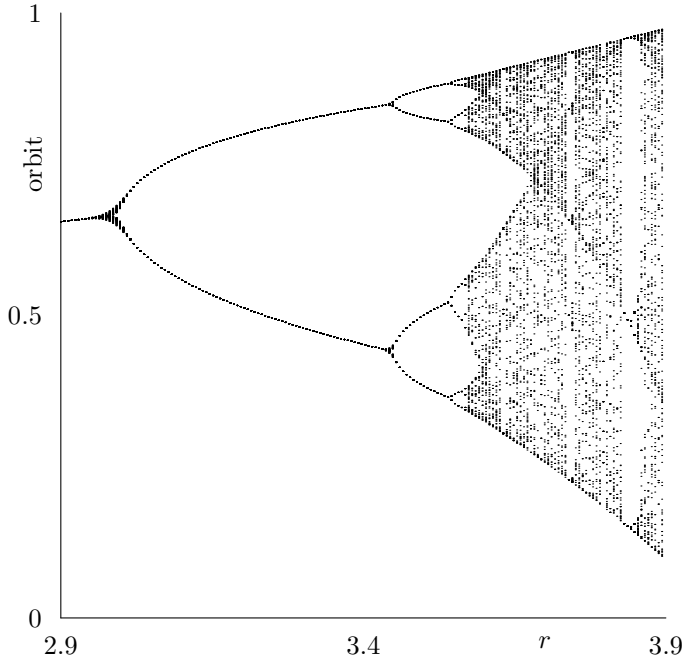
The following diagrams are 'standard' in the theory of iterative processes:

- The cobweb-graph of applying the cosine function iteratively.
- The bifurcation diagram of the logistic function, $f(x) = rx(1-x)$ for $0 < r < 4$.

The code that produced these diagrams is shown below.



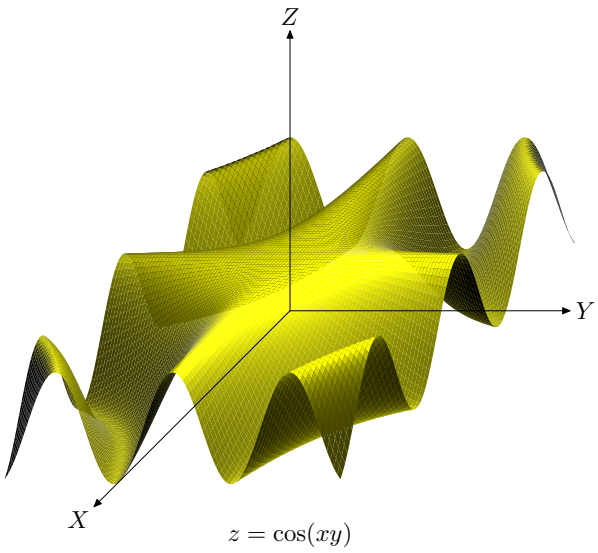

```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

beginfig(1)
% some constants
u := 10cm;
numeric xmin, xmax, ymin, ymax, xinc;
xmin := 0.5; xmax := 1.0;
ymin := xmin; ymax := xmax;
xinc := 0.02;

% draw axes
draw (xmin,ymin)*u -- (xmax,ymin)*u;
draw (xmin,ymin)*u -- (xmin,ymax)*u;

% define routine to compute function values
def compute_curve(suffix g)(expr xmin, xmax, xinc) =
    ( (xmin,g(xmin))
    for x=xmin+xinc step xinc until xmax:
      .. (x,g(x))
    endfor )
enddef;

% compute and draw cosine curve
numeric pi; pi := 3.1415926;
numeric radian;  radian := 180/pi; % 2pi*radian = 360 ;
vardef cos primary x = (cosd(x*radian)) enddef;
path p;
p := compute_curve(cos, xmin, xmax, xinc) scaled u;
draw p;
% draw identity graph
draw (xmin,ymin)*u -- (xmax,xmax)*u withcolor 0.5white;
```




```
% compute the orbit starting from some point
numeric x, initial, orbitlength;
x := 1.0; % the starting point
initial := 1; % some initial iterations
orbitlength := 15; % number of iterations
for i=1 upto initial: % do initial iterations
  x := cos(x);
endfor;
dotlabel("", (x,cos(x))*u); % mark starting point
for i=1 upto orbitlength:  % draw hooks
  draw (x,cos(x))*u -- (cos(x),cos(x))*u --
    (cos(x),cos(cos(x)))*u withcolor blue;
  x := cos(x); % next value
endfor;

% draw axis labels
labeloffset := 0.25cm;
label.bot(decimal(xmin), (xmin,ymin)*u);
label.bot(decimal(xmax), (xmax,ymin)*u);
label.lft(decimal(ymin), (xmin,ymin)*u);
label.lft(decimal(ymax), (xmin,ymax)*u);
endfig;
```

```
beginfig(2)
numeric rmin, rmax, r, dr, n, ux, uy;
rmin := 2.9; rmax := 3.9;
r := rmin; n := 175;
dr := (rmax - rmin)/n;
ux := 8cm; uy := 8cm;
for i = 1 upto n:
  x := 0.5; % our starting point
  for j=1 upto 75: % initial iterations
    x := r*x*(1-x);
  endfor
    for j=1 upto 150: % the next 100 iterations
        x := r*x*(1-x);
        draw (r*ux,x*uy) withpen pencircle scaled .5pt;
    endfor
    r := r+dr;
endfor;

% draw axes and labels
draw (rmin*ux,0) -- (rmax*ux,0);
draw (rmin*ux,0) -- (rmin*ux,uy);
labeloffset := 0.25cm;
label.bot(decimal(rmin), (rmin*ux,0));
label.bot(decimal((rmin+rmax)/2), ((rmin+rmax)/2*ux,0));
label.bot(decimal(rmax), (rmax*ux,0));
label.lft(decimal(0), (rmin*ux,0));
label.lft(decimal(0.5), (rmin*ux,0.5*uy));
label.lft(decimal(1), (rmin*ux,uy));
label.bot(btex $r$ etex, ((rmax-0.2)*ux,0));
label.lft(btex orbit etex rotated 90, (rmin*ux,0.75*uy));
endfig;

end;
```

A Surface Plot
You can draw surface plots from basic principles. We give one example.

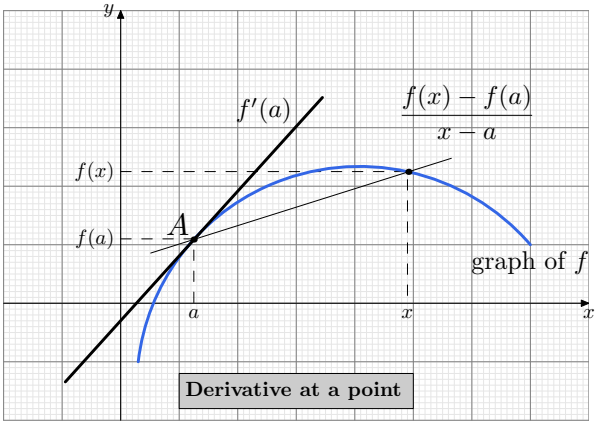



```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

% u: dimensional unit
% xp, yp, zp: coordinates of light source
% bf : brightness factor
% base_color : base color
numeric u,xp,yp,zp,bf;
color base_color;
u = 1cm;
xp := 3;  yp := 3; zp := 5;
bf := 30;
base_color := red+green;

% O, Xr, Yr, Zr : reference frame
pair O,Xr,Yr,Zr;
O = (0,0);
Xr = (-.7,-.7) scaled u;
Yr = (1,0) scaled u;
Zr = (0,1) scaled u;

% for drawing the reference frame
vardef frameXYZ(expr s) =
  drawarrow O--Xr scaled s;
  drawarrow O--Yr scaled s;
  drawarrow O--Zr scaled s;
  label.llft(btex $X$ etex scaled 1.25, (Xr scaled s));
  label.rt(btex $Y$ etex scaled 1.25, (Yr scaled s));
  label.top(btex $Z$ etex scaled 1.25, (Zr scaled s));
enddef;

% from 3D to 2D coordinates
vardef project(expr x,y,z) = x*Xr + y*Yr + z*Zr enddef;

% numerical derivatives by central differences
vardef diffx(suffix f)(expr x,y) =
  numeric h; h := 0.01;
  (f(x+h,y)-f(x-h,y))/(2*h)
enddef;
vardef diffy(suffix f)(expr x,y) =
 numeric h; h := 0.01;
 (f(x,y+h)-f(x,y-h))/(2*h)
enddef;

% Compute brightness factor at a point
vardef brightnessfactor(suffix f)(expr x,y,z) =
 numeric dfx,dfy,ca,cb,cc;
 dfx := diffx(f,x,y);
 dfy := diffy(f,x,y);
 ca := (zp-z)-dfy*(yp-y)-dfx*(xp-x);
 cb := sqrt(1+dfx*dfx+dfy*dfy);
 cc := sqrt((z-zp)*(z-zp)+(y-yp)*(y-yp)+(x-xp)*(x-xp));
 bf*ca/(cb*cc*cc*cc)
enddef;

% compute the colors and draw the patches
vardef
 z_surface(suffix f)(expr xmin,xmax,ymin,ymax,nx,ny) =
 numeric dx,dy,xt,yt,zt,factor[][];
 pair Z[][];
```

```
 dx := (xmax-xmin)/nx;
 dy := (ymax-ymin)/ny;
 for i=0 upto nx:
   xt := xmin+i*dx;
   for j=0 upto ny:
     yt := ymin+j*dy;
     zt := f(xt,yt);
     Z[i][j] = project(xt,yt,zt);
     factor[i][j] := brightnessfactor(f,xt,yt,zt);
   endfor
 endfor
 for i = 0 upto nx-1:
   for j= 0 upto ny-1:
    fill Z[i][j]--Z[i][j+1]--Z[i+1][j+1]--Z[i+1][j]--cycle
          withcolor factor[i][j]*base_color;
   endfor
 endfor
enddef;

beginfig(1);
xp := 3;
yp := 3;
zp := 10;
bf := 100;

numeric pi; pi := 3.14159;
vardef cos primary x = cosd(x/pi*180) enddef;
vardef f(expr x,y) = cos(x*y) enddef;
z_surface(f,-3,3,-3,3,100,100);
frameXYZ(5);
label(btex $z = \cos(xy)$ etex scaled 1.25,(0,-4cm));
endfig;
end;
```

Miscellaneous

We adopt another example from Jean-Michel Sarlat that uses his macro package `courbe` and another one called `grille`. The example has been downloaded and slightly adapted from `http://melusine.eu.org/syracuse/metapost/cours/sarlat/derivation/`.




```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

input courbes;
input grille;

vardef droite(expr a,b,t) =
 (t[a,b])--(t[b,a])
enddef;
path c,cartouche;
c=(2.3cm,1cm)..(4.5cm,4cm)..(9cm,3cm);
cartouche = (3cm,2mm)--(7cm,2mm)--(7cm,8mm)--(3cm,8mm)--
  cycle;
pair A,M;
A = point 0.6 of c;
M = point 1.5 of c;

vardef tangente(expr t,x) =
 pair X,Y;
 X := point (t-.05) of c;
 Y := point (t+.05) of c;
```

```
 droite(X,Y,x)
enddef;
beginfig(1);
grille(1cm,0,10cm,0,7cm);
repere(10cm,7cm,2cm,2cm,1cm,1cm);
trace.axes(.5pt);
marque.unites(0.1);

%% lectures sur la grille
numeric xa,ya,xm,ym;
xa = 1.25; ya = 1.1 ; xm = 4.9 ; ym = 2.25;
pair AA,MM;
AA = (xa,ya) ; MM = (xm,ym) ;
projection.axes(AA,0.5,1.7);
projection.axes(MM,0.5,1.7);
label.bot(btex $a$ etex,rpoint(xa,0));
label.bot(btex $x$ etex,rpoint(xm,0));
label.lft(btex $f(a)$ etex, rpoint(0,ya));
label.lft(btex $f(x)$ etex, rpoint(0,ym));
%% fin des lectures

draw c withpen pencircle scaled 1.5pt withcolor (.2,.4,.9);

draw droite(A,M,1.2);

draw tangente(0.6,9) withpen pencircle scaled 1.5pt;

dotlabel.ulft(btex $A$ etex scaled 1.5,A);
dotlabel("",M);
label(btex $\displaystyle\frac{f(x)-f(a)}{x-a}$ etex
  scaled 1.25,
M shifted (1cm,1cm));
label.ulft(btex $f'(a)$ etex scaled 1.25, (5cm,5cm));
label.bot(btex graph of $f$ etex scaled 1.25, point 2 of c);

%% Cartouche
fill cartouche withcolor .8white;
draw cartouche;
label.rt(btex \textbf{Derivative at a point}
  etex,(3cm,5mm));
%% fin du cartouche

decoupe.repere;
etiquette.axes;
endfig;
end;
```

References

[GRS94] Michel Goossens, Sebastian Rahtz, Frank Mittelbach. *The LaTeX Graphics Companion*, Addison-Wesley (1994), ISBN 0-201-85469-4.

[Hag02] Hans Hagen. *The Metafun Manual*, 2002. downloadable as `www.pragma-ade.com/general/manuals/metafun-p.pdf`

[Hob92a] John D. Hobby: *A User's manual for MetaPost*, AT&T Bell Laboratories Computing Science Technical Report 162, 1992.

[Hob92b] John D. Hobby: *Drawing Graphs with MetaPost*, AT&T Bell Laboratories Computing Science Technical Report 164, 1992.

Solutions to the Exercises

EXERCISE 3

```
beginfig(1);
draw fullcircle scaled 2cm;
endfig;
end;
```

EXERCISE 4

```
beginfig(1);
path p;
p := (0,0)--(2cm,0)--(1cm,sqrt(3)*cm)--(0,0);
draw p;
endfig;

beginfig(2);
draw p scaled 1.5;
endfig;

end;
```

EXERCISE 5

```
beginfig(1);
u := 0.5cm;
draw (2u,0)--(u,sqrt(3)*u)--(-u,sqrt(3)*u)--(-2u,0)
   --(-u,-sqrt(3)*u)--(u,-sqrt(3)*u)--(2u,0);
endfig;
end;
```

EXERCISE 6

```
beginfig(1);
u=1cm;
draw (0,0)--(2*sqrt(3)*u,0)--(sqrt(3)*u,3u)--(0,0);
draw (0,0)--(sqrt(3)*u,u)--(2*sqrt(3)*u,0);
draw (sqrt(3)*u,u)--(sqrt(3)*u,3u);
endfig;

beginfig(2);
draw unitsquare scaled 2u shifted (-u,-u);
draw unitsquare scaled 4u shifted (-2u,-2u);
draw (u,u)--(2u,2u);
draw (-u,u)--(-2u,2u);
draw (-u,-u)--(-2u,-2u);
draw (u,-u)--(2u,-2u);
endfig;

end;
```

EXERCISE 7

```
warningcheck := 0;
numeric p, q, n;
n := 12;
p := 2**n;
q := 2**n+1;
show p,q;
end;
```

EXERCISE 9

```
beginfig(1)
draw unitsquare scaled 70;
draw (10,20);
draw (10,15) scaled 2;
draw (30,40) withpen pencircle scaled 4;
pickup pencircle scaled 8;
draw (40,50);
draw (50,60);
endfig;
end;
```

EXERCISE 10

```
beginfig(1)
pickup pencircle scaled 6bp;
z.P = (1cm,2cm);
draw z.P;
draw 2(x.P,y.P);
endfig;

end;
```

EXERCISE 11

```
pair A,B,C,A',B',C';
u := 1cm;
A=(0,0);
B=(5u,0);
C=(2u,3u);
A'=1/2[B,C];
B'=1/2[A,C];
C'=1/2[A,B];

beginfig(1)
draw A--B--C--A;
draw A--A';
draw B--B';
draw C--C';
endfig;

beginfig(2)
draw A--B--C--A;
draw A--A';
draw B--B';
draw C--C';
dotlabel.lft("A",A);
dotlabel.urt("B",B);
dotlabel.top("C",C);
dotlabel.urt("A'",A');
dotlabel.ulft("B'",B');
dotlabel.bot("C'",C');
endfig;

beginfig(3)
pair G;
G = whatever[A,A'] = whatever[B,B'];
draw A--B--C--A;
draw A--A';
draw B--B';
draw C--C';
dotlabel.lft("A",A);
dotlabel.urt("B",B);
dotlabel.top("C",C);
dotlabel.urt("A'",A');
dotlabel.ulft("B'",B');
dotlabel.bot("C'",C');
dotlabel.llft("",G);
label.llft("G",G-(0,1.5mm));
endfig;
end;
```

EXERCISE 12

```
beginfig(1)
s := 2cm;
z0 = s*dir(0);
z1 = s*dir(72);
z2 = s*dir(2*72);
z3 = s*dir(3*72);
z4 = s*dir(4*72);
draw z0--z1--z2--z3--z4--z0;
endfig;
end;
```

EXERCISE 13

```
beginfig(1);
z = (1cm,1cm);
draw z withpen pencircle scaled 6;
z1 = z - 2cm*dir(135);
z2 = z + 2cm*dir(135);
z3 = z + 2cm*dir(105);
draw z1--z2;
draw z--z3;
endfig;
end;
```

EXERCISE 14

```
pair p[];  p0 = (0,0);  p1 = (2cm,3cm);  p2 = (3cm,2cm);

beginfig(1);
fill p0--p1--p2--p0;
endfig;

beginfig(2);
fill p0--p1--p2--cycle withcolor 0.5white;
endfig;

end;
```

EXERCISE 15

```
beginfig(1);
draw origin--2*cm*dir(0);
draw origin--2*cm*dir(40);
drawarrow 1cm*dir(0){dir(90)}..
          1cm*dir(40){dir(130)}
endfig;

end;
```

EXERCISE 16

```
beginfig(1);
u := 1cm;
z0 = origin;  z1 = (2u,0);  z2 = (u,sqrt(3)*u);
draw z0--z1--z2;
draw z0{up}...z2;
draw z2{down}...z0;
pickup pencircle scaled 6;
draw z0; draw z1; draw z2;
endfig;

end;
```

EXERCISE 17

```
beginfig(1)
path p;
p = (0,1cm)..(1cm,0)...(0,-1cm);
fill p{dir(157)}..(0,0){dir(23)}..{dir(157)}cycle;
draw p..(-1cm,0)..cycle;
fill (0,-0.6cm)..(0.1cm,-0.5cm)..(0,-0.4cm)..
  (-0.1cm,-0.5cm)..cycle withcolor white;
fill (0,0.6cm)..(0.1cm,0.5cm)..(0,0.4cm)..(-0.1cm,0.5cm)..
  cycle;
endfig;
end;
```

EXERCISE 18

```
pair A,B,C,C';
path arc,mark[];
numeric AC, BC; % directional angle of AC and BC
u := 0.75cm; A=(0,0); B=(5u,0); C=(2u,3u);
AC = angle(A-C); BC = angle(B-C);
C' = whatever[A,B] = C + whatever*dir(1/2*AC+1/2*BC);
arc = (C+0.5u*dir(AC)){dir(AC+90)}..
      {dir(BC+90)}(C+0.5u*dir(BC));
mark[1] = C+0.4u*dir(3/4*AC+1/4*BC)--
          C+0.6u*dir(3/4*AC+1/4*BC);
mark[2] = C+0.4u*dir(1/4*AC+3/4*BC)--
          C+0.6u*dir(1/4*AC+3/4*BC);
beginfig(1)
draw A--B--C--cycle; draw C--C';
dotlabel.lft("A",A); dotlabel.urt("B",B);
dotlabel.top("C",C); dotlabel.bot("C'",C');
draw arc; draw mark[1]; draw mark[2];
endfig;
end;
```

EXERCISE 19

```
pair A,B,C,A',B',C',I;
u := 0.75cm; A=(0,0); B=(5u,-u); C=(2u,3u);
A' = whatever[B,C] = A + whatever*dir(
       1/2*angle(B-A)+1/2*angle(C-A));
B' = whatever[A,C] = B + whatever*dir(
       1/2*angle(A-B)+1/2*angle(C-B));
C' = whatever[A,B] = C + whatever*dir(
       1/2*angle(A-C)+1/2*angle(B-C));
I = whatever[A,A']=whatever[B,B'];
beginfig(1)
draw A--B--C--cycle;
draw A--A'; draw B--B'; draw C--C';
draw A'..B'..C'..cycle;
dotlabel.lft("A",A); dotlabel.rt("B",B);
dotlabel.top("C",C); dotlabel.urt("A'",A');
dotlabel.ulft("B'",B'); dotlabel.bot("C'",C');
labeloffset := 0.3cm;
dotlabel.llft("I",I);
endfig;
end;
```

EXERCISE 20

```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex
```

```
beginfig(1)
u := 1cm;
numeric xmin, xmax, ymin, ymax, xinc;
xmin := 0; xmax := 4;
ymin := 0; ymax := 2;
% draw axes
draw (xmin,0)*u -- (xmax,0)*u;
draw (0,ymin)*u -- (0,ymax)*u;
% compute and draw graph of function
path p;
xinc := 0.1;
p := (xmin,sqrt(xmin))*u
    for x=xmin+xinc step xinc until xmax:
      .. (x,sqrt(x))*u
    endfor;
draw p withpen pencircle scaled 2;
% draw tickmarks and labels
for i=0 upto xmax:
  label.bot(decimal(i), (i,0)*u);
  draw (i,-0.05)*u--(i,0.05)*u;
endfor;
for i=0 upto ymax:
  label.lft(decimal(i), (0,i)*u);
  draw (-0.05,i)*u--(0.05,i)*u;
endfor;
\labeloffset := 0.5u;
label.bot(btex $x$ etex, ((xmin+xmax)/2,0)*u);
label.lft(btex $y$ etex, (0,(ymin+ymax)/2)*u);
label(btex $y=\sqrt{x}$ etex,
  ((xmin+3xmax)/4,(ymin+2ymax)/3)*u);
endfig;
end;
```

EXERCISE 21

```
verbatimtex
%&latex
\documentclass{article}
\usepackage{amsmath,amssymb}
\begin{document}
etex

beginfig(1)
a := 3cm; b := 2cm;
phi := angle(a,b);
draw (-1/2cm,0)--(a+1/2cm,0); % horizontal axis
draw (0,-1/2cm)--(0,b+1/2cm); % vertical axis
draw (a,0)--(a,b)--(0,b) dashed evenly;
draw origin--(a,b);
label.llft(btex $0$ etex, (0,0));
label.bot(btex $a$ etex, (a,0));
label.lft(btex $ib$ etex, (0,b));
label.lft(btex $\mathbb{C}$ etex, (0,b+1/2cm));
label.rt(btex $a+ib=z$ etex, (a,b));
label(btex $\lbrace$ etex xscaled 1.5 yscaled 10
  rotated(phi-90), 1/2(a,b) + dir(90+phi)*1.5mm);
label(btex $|z|$ etex, 1/2(a,b) + dir(90+phi)*5mm);
draw (1/2cm,0){up}..1/2cm*dir(phi){dir(90+phi)};
label(btex $\phi$ etex, 7mm*dir(phi/2));
endfig;
end;
```

EXERCISE 22

```
verbatimtex
%&latex
\documentclass{article}
\begin{document}
etex

beginfig(1)
u := 0.6cm;
labeloffset := 1/3u;
defaultscale := 8pt/fontsize(defaultfont);
% the data
z1 = (1980,86); z2 = (1985,85); z3 = (1990,91);
z4 = (1995,86); z5 = (2000,83);
yoff := 80; % vertical offset
% draw axes
draw (0,0)--(1/4u,0)--(1/3u,1/8u)--(5/12u,-1/8u)
   --(1/2u,0)--(6u,0);
draw (0,0)--(0,1/4u)--(-1/8u,1/3u)--(1/8u,5/12u)
   --(0,1/2u)--(0,6.5u);
% draw horizontal axis and data points
for i=1 upto 5:
  draw (1/2+i,-1/12)*u--(1/2+i,1/12)*u; %ticks
  label.bot(decimal(x[i]), ((1/2+i)*u,0)); % labels
  dotlabel("", (1/2+i,(y[i]-yoff)/2)*u); % data point
endfor
% draw line graph
draw (3/2,(y[1]-yoff)/2)*u
  for i=2 upto 5: --(1/2+i, (y[i]-yoff)/2)*u endfor;
% draw vertical axis
for i=1 upto 6:
  draw (-1/12,i)*u--(1/12,i)*u; % ticks
  label.lft(decimal(yoff+2i), (0,i*u)); % labels
endfor
% draw horizontal and vertical texts
label.bot(btex year etex, (7/2u,-1/2u));
label.lft(btex beer consumption (liter) etex
  rotated 90, (-1/2u,7/2u));
endfig;

end;
```

EXERCISE 23

```
u := 1/2cm; defaultscale := 8pt/fontsize(defaultfont);
beginfig(1);
path sqr; sqr := unitsquare scaled u;
for i=0 upto 10:
  label.bot(decimal(i/10), ((i+1/2)*u,0));
  label.lft(decimal(i/10), (0,(i+1/2)*u));
  for j=0 upto 10:
    fill sqr shifted (i*u,j*u) withcolor
      (i*0.1+j*0.1)/3*white;
    draw sqr shifted (i*u,j*u); % for drawing the grid
  endfor;
endfor;
label.bot("r", (6u,-2/3u));
label.lft("g", (-u,6u));
label.top("RGB(r,g,0)", (6u,11u));
endfig;

beginfig(2);
path sqr; sqr := unitsquare scaled u;
for i=0 upto 10:
  label.bot(decimal(i/10), ((i+1/2)*u,0));
  label.lft(decimal(i/10), (0,(i+1/2)*u));
  for j=0 upto 10:
    fill sqr shifted (i*u,j*u) withcolor
      (0.3*i*0.1+0.59*j*0.1)*white;
    draw sqr shifted (i*u,j*u); % for drawing the grid
  endfor;
```

```
endfor;
label.bot("r", (6u,-2/3u));
label.lft("g", (-u,6u));
label.top("RGB(r,g,0)", (6u,11u));
endfig;

end;
```

EXERCISE 24

```
beginfig(1);
path O[], l[];  pair A[];
O1 = fullcircle xscaled 1cm yscaled 1/2cm shifted (0,1cm);
O2 = O1 rotated -120;
A1 = (-1/2cm,1cm);
A2 = A1 xscaled -1;
l1 = A1{down}..(A2 rotated -120){down rotated 60};
l2 = A2{down}..(A1 rotated -120){down rotated 60};
draw O1; draw O2; draw l1; draw l2;
endfig;
end;
```

EXERCISE 25

```
beginfig(1);
r := 3cm;
path C[], p[], a;
C1 = fullcircle scaled 2r;
C2 = fullcircle scaled (2/3*2r);
p1 = origin -- r*dir(20);
p2 = origin -- r*dir(35);
a := buildcycle(p1,C2,p2,C1);
fill a withcolor red+green;
draw a withpen pencircle scaled 1bp;
draw p1; draw p2;
endfig;
end;
```

EXERCISE 26

```
beginfig(1);
u := 1cm; a := 6u; b := 3.5u;
pair sun; sun := (-1.75u,0);
path E[], p[], area[];
E1 = fullcircle xscaled a yscaled b;
E2 = E1 scaled 1.1;
p1 = sun -- (5u*dir(8) shifted sun);
p2 = sun -- (4u*dir(28) shifted sun);
p3 = sun -- (2u*dir(75) shifted sun);
p4 = sun -- (1.75u*dir(150) shifted sun);
area1 = buildcycle(p1,E1,p2);
area2 = buildcycle(p3,E1,p4);
fill area1 withcolor red+green;
fill area2 withcolor red+green;
draw p1; draw p2; draw p3; draw p4; draw E1;
draw (-a/2,0)--(a/2,0) dashed withdots;
draw sun withpen pencircle scaled 6bp;
label.bot(btex Sun etex, sun);
numeric t[]; % intersection times
t1 = ypart (p1 intersectiontimes E2);
t2 = ypart (p2 intersectiontimes E2);
t3 = ypart (p3 intersectiontimes E2);
t4 = ypart (p4 intersectiontimes E2);
drawarrow subpath (t1,t2) of E2;
drawarrow subpath (t3,t4) of E2;
label.urt(btex $\Delta t$ etex, point((t1+t2)/2) of E2);
label.ulft(btex $\Delta t'$ etex, point((t3+t4)/2) of E2);
endfig;

end;
```

EXERCISE 27

```
beginfig(1);
u:=1cm;
draw (-2u,0)--(2u,0);
draw (0,-2u)--(0,2u);
for i=-2u step u until 2u:
  draw (i,u/10)--(i,-u/10);
  draw (u/10,i)--(-u/10,i);
endfor;
for i=-2u step u/5 until 2u:
  draw (i,u/20)--(i,-u/20);
  draw (u/20,i)--(-u/20,i);
endfor;
endfig;
end;
```

EXERCISE 28

```
beginfig(1);
u:=1cm;
numeric xmin, xmax, ymin, ymax;
xmin := -2.1; xmax := 2.1;
ymin := -0.5; ymax := 4.5;
% draw axes
path xaxis, yaxis;
xaxis = (xmin,0)*u -- (xmax,0)*u;
yaxis = (0,ymin)*u -- (0,ymax)*u;
% compute the graph of f
def f(expr x) = (4-x**2) enddef;
inc := 0.01;
path pts_f;
pts_f := (xmin*u,f(xmin)*u)
    for x=xmin+inc step inc until xmax:
      .. (x*u,f(x)*u)
    endfor;
% compute and draw rectangles
n := 12; % number of rectangles
x0 := -2; x1 := 2;
inc := (x1-x0)/n;
for i=x0 step inc until x1-inc:
      path p;
      p = (i,0)--(i+inc,0)--(i+inc,max(f(i),f(i+inc)))
          --(i, max(f(i),f(i+inc)))--cycle;
      p := p scaled u;
      fill p withcolor red+green;
      draw p;
    endfor;
draw pts_f withpen pencircle scaled 2;
draw xaxis;
draw yaxis;
pair t; % translation vector
t := (6u,0);
for i=x0 step inc until x1-inc:
      path p;
      p = (i,0)--(i+inc,0)--(i+inc,min(f(i),f(i+inc)))
          --(i, min(f(i),f(i+inc)))--cycle;
      p := p scaled u shifted t;
      fill p withcolor red+green;
      draw p;
    endfor;
draw pts_f shifted t withpen pencircle scaled 2;
```

```
draw xaxis shifted t;
draw yaxis shifted t;
endfig;

end;
```

EXERCISE 29

```
beginfig(1)
u:=1cm;
draw (-2u,0)--(2u,0);
draw (0,-2u)--(0,2u);
for i=-2u step u until 2u:
    draw(i,2u)--(i,-2u);
    draw (2u,i)--(-2u,i);
endfor;
for i=-2u step u/10 until 2u:
    draw (i,2u)--(i,-2u) withpen pencircle scaled .1bp;
    draw (2u,i)--(-2u,i) withpen pencircle scaled .1bp;
endfor;
endfig;
end;
```

EXERCISE 30

```
beginfig(1)
u:=1cm;
pair A[], B[];
n := 3;
for i=1 upto n:
  A[i] = (0,i*u);
  B[i] = (n/2*u,i*u);
endfor;
for i=1 upto n:
  for j=1 upto n:
    draw A[i]--B[j];
  endfor;
endfor
for i=1 upto n:
  dotlabel.lft("a" & decimal(i), A[i]);
  dotlabel.rt("b" & decimal(i), B[i]);
endfor;
endfig;
end;
```

EXERCISE 31

```
u:=3cm;
vardef koch(expr A,B,n) =
  save C; pair C;
  C = A rotatedaround(1/3[A,B], 120);
  if n>1:
     koch( A,        1/3[A,B], n-1);
     koch( 1/3[A,B], C,        n-1);
     koch( C,        2/3[A,B], n-1);
     koch( 2/3[A,B], B,        n-1);
  else:
     draw A--1/3[A,B]--C--2/3[A,B]--B;
  fi;
enddef;

beginfig(1)
z0=(u,0);
z1=z0 rotated 120;
z2=z1 rotated 120;
draw z0--z1--z2--cycle shifted (-3u,0);
drawarrow (-1.75u,0)--(-1.25u,0);
koch( z0, z1, 1);
koch( z1, z2, 1);
koch( z2, z0, 1);
endfig;

beginfig(2)
z0=(u,0);
z1=z0 rotated 120;
z2=z1 rotated 120;
koch( z0, z1, 2);
koch( z1, z2, 2);
koch( z2, z0, 2);
endfig;

beginfig(3)
z0=(u,0);
z1=z0 rotated 120;
z2=z1 rotated 120;
koch( z0, z1, 6);
koch( z1, z2, 6);
koch( z2, z0, 6);
endfig;

end;
```

André Heck

Antykwa Toruńska
ver. 2.03

Van de redactie
Bij deze Maps vindt u een klein boekje dat dient ter promotie van het font Antykwa Toruńska, en de volgende pagina's van de Maps bevatten een aantal pagina's uit de documentatie die hoort bij de zojuist verschenen uitgebreidere versie van deze fontfamilie.
De tekst hieronder is een vertaling van de eerste pagina van de documentatie die volgt. Gezien de kwaliteit van het zetwerk vonden we het leuker om de documentatie op te nemen als fascimile, in plaats van het opnieuw te zetten in de Maps-stijl.

Antykwa Toruńska (wat 'Antieke van Torun' betekent) is een tweedelige fontfamilie ontworpen door Zygfryd Gardzielewski, een typograaf uit Toruń, Polen. Een paar mensen hebben samengewerkt aan het font gedurende de ontwerpfase in metaal. Het font wordt hoofdzakelijk gebruikt voor het zetten van kleine drukken. Zijn kenmerkende eigenschappen zijn het verwijden van de verticale stammen bij de bovenkant en de golvende vorm van enkele horizontale en diagonale lijnen alsook de schreven. Antykwa Toruńska werd voor het eerst gegoten in metaal in 1960, door de Grafmasz zetterij te Warschau. Het werd geproduceerd in een gewone, een halfvette en een schuine variant, in groottes van 6 tot 48 dd.

Ik begon met het digitalizeren van Antykwa Toruńska in begin 1995. De eerste, zeer slechte versie was gebaseerd op een fontcatalogus. Dat was een zeer onvolmaakte bron. De volgende versie was gebaseerd op fotokopieën van de originele ontwerptekeningen die mij door Zygfryd Gardzielewski ter beschikking werden gesteld, die ik daarvoor zeer dankbaar ben. Andrzej Tomaszewski, een bekende Poolse typograaf, maakte me ervan bewust dat ook deze versie van slechte kwaliteit was, omdat het te nauwgezet probeerde om een origineel te volgen dat in de pre-digitale era werd gecreëerd, dertig jaar geleden, gebruikmakend van een traditionele benadering.

Deze versie (1.0) mag gedigitaliseerd worden genoemd in zoverre dat zij exact was en bouwelementen in het gehele font hergebruikte. Ze werd vrijgegeven in 1998 en bestond (net als het origineel) in drie varianten: rechtop, vet en cursief.

De huidige distributie werd gemaakt met het METATYPE1 pakket door Bogusław Jackowski, Janusz M. Nowacki en Piotr Strzelczyk. Versie 2.01 bevat een zeer uitgebreide tekenset (bijvoorbeeld Cyrillisch, Grieks, de meestgebruikte wiskundige en monetaire symbolen, extra ligaturen), evenals extra varianten (licht, normaal, halfvet en vet in normale en versmalde vorm). Ik heb aparte sets van steeds 256 karakters in Type 1 gemaakt voor Windows en Linux (in diverse encodings), en volledige fontsets in TrueType en OpenType met meer dan 1060 tekens elk.

Heel wat mensen hebben mij grootmoedig geholpen tijdens het voorbereiden van deze versie van Antykwa Toruńska. Ik zou mijn dankbaarheid aan mevr. Janina Gradzielewska (de vrouw van Zygfryd Gardzielewski) willen uitdrukken, en aan Bogusław Jackowski, Andrzej Tomaszewski en Marcin Woliński. Jerzy Ludwichowski vertaalde dit document in het Engels. Zeer veel dank.

Janusz Marian Nowacki
april 2005

GUST
Grupa Użytkowników Systemu TEX

The Polish TEX Users Group, GUST, supports initiatives leading to digitization of Polish traditional fonts. GUST also supports polonization of popular fonts published under GNU or GNU-like licences. The resulting fonts are freely available.

Antykwa Toruńska

ver. 2.03

Type designer: Zygfryd Gardzielewski, Toruń, Poland

Font author: Janusz Marian Nowacki, Grudziądz, Poland

April 2005

J.Nowacki@gust.org.pl
www.janusz.nowacki.strefa.pl

Zygfryd Gardzielewski (1914–2001)

Antykwa Toruńska (meaning just "Antiqua of Torun") is a two-element typeface designed by Zygfryd Gardzielewski, a typographer from Toruń (Thorn), Poland. A few people cooperated on the font during the metal type design phase. The font is mainly used for typesetting of small prints. Its characteristic features are the widening of vertical stems at the top and the wave-like form of some of the horizontal and diagonal lines as well as of the serifs. Antykwa Toruńska was first cast in metal in 1960 in the Grafmasz typefoundry in Warsaw. It was produced in the ordinary, semibold and slanted faces in sizes from 6 to 48 dd.

I began digitizing Antykwa Toruńska in early 1995. The first, very poor version was based on a font catalogue. It was a very imperfect source. The next version was based on photocopies of the original design drawings made available to me by Zygfryd Gardzielewski, to whom I am very grateful for this. Andrzej Tomaszewski, a well known Polish typographer, made me aware that this version was also of poor quality because it too closely tried to follow the original which was created in the pre-digital era, thirty years ago, using the traditional approach. Version 1.0, which deserves to be called computerised in that it was precise and reused character building elements across the font, was released in 1998 and comprised (similarly as the original) of three faces: upright, bold and italic.

The current distribution was generated using the METATYPE1 tools package authored by Bogusław Jackowski, Janusz M. Nowacki and Piotr Strzelczyk. Version 2.01 contains a greatly extended character set (e.g., cyrillic, greek, most often used mathematical symbols and currency symbols, additional ligatures), as well as additional typefaces (light, regular, medium and bold in normal and condensed widths).

I prepared separate Type 1 sets of 256 characters for Windows and Linux (in various encodings) and complete TrueType and OpenType font sets with over 1060 glyphs each.

A lot of people helped me generously in preparing this version of Antykwa Toruńska. I would like to express my gratitude to Mrs Janina Gradzielewska (Zygfryd Gardzielewski's wife), Bogusław Jackowski, Andrzej Tomaszewski, Marcin Woliński. Jerzy Ludwichowski translated this document into English. Very many thanks.

A woodcut by Zygfryd Gardzielewski.

INSTALLING IN A TEX SYSTEM

CAUTION: the installation requires previous versions of Antykwa Toruńska to be uninstalled.

The Antykwa Toruńska PostScript font package (file `AntykwaTorunska-tex-2_03.zip`) for TEX conforms to the TEX Directory Structure (TDS) v. 1.1 which allows for an easy installation, e.g., in the popular TEX Live distribution. It suffices to copy the `doc`, `fonts` and `tex` subdirectories into their counterparts in the global or local tree of your TEX installation. In the next step one should add the font map name, e.g., `antt.map` to the `web2c/updmap.cfg` file and run the `updmap` program.

ABCabc
ABCabc
ABCabc
ABCabc
ABCabc
ABCabc
ABCabc
ABCabc

The original, cast in lead Antykwa Toruńska was designed by Gardzielewski in only three variants: normal, semibold and italic. Now eight variants are available.

ABCabc
ABCabc
ABCabc
ABCabc

The Condensed fonts are provided for special applications.

ABCabc
ABCabc
ABCabc
ABCabc

A smallcaps variant is also available.

THE PACKAGE for TEX

The complete distribution of Antykwa Toruńska for TEX contains the following files in a TDS-conformant tree:

doc/fonts/antt/

- documentation and examples of use

fonts/enc/dvips/antt

- files with reencodings into different TEX font layouts

fonts/map/dvips/antt

- the font mapping files (to be included – using the `updmap` program – in the global `psfonts.map` file used by the `dvips` program as well as the analogues for the `pdftex` and `dvipdfm` programs)
  The `.map` files for the following encodings are included: `ec`, `qx`, `texnansi`, `wncy`, `t2a`, `t2b`, `t2c`, `greek`, `cs`, `t5`.

fonts/afm/public/antt

- metric files `*.afm`

fonts/tfm/public/antt

- metric files `*.tfm` for TEX
  - ▷ **latin characters**
    (`*` *denotes the encoding used:* `ec`, `cs`, `t5`, `qx` *or* `texnansi`)

    normal text fonts

    `*-anttl` — Antykwa Toruńska Light-Regular
    `*-anttli` — Antykwa Toruńska Light-Italic
    `*-anttm` — Antykwa Toruńska Medium-Regular
    `*-anttmi` — Antykwa Toruńska Medium-Italic
    `*-anttr` — Antykwa Toruńska Regular
    `*-anttri` — Antykwa Toruńska Italic
    `*-anttb` — Antykwa Toruńska Bold
    `*-anttbi` — Antykwa Toruńska BoldItalic

    Caps text fonts

    `*-anttlcap` — Antykwa Toruńska Caps Light-Regular
    `*-anttlicap` — Antykwa Toruńska Caps Light-Italic
    `*-anttmcap` — Antykwa Toruńska Caps Medium-Regular
    `*-anttmicap` — Antykwa Toruńska Caps Medium-Italic
    `*-anttrcap` — Antykwa Toruńska Caps Regular
    `*-anttricap` — Antykwa Toruńska Caps Italic
    `*-anttbcap` — Antykwa Toruńska Caps Bold
    `*-anttbicap` — Antykwa Toruńska Caps BoldItalic

Nautical ("old style") digits are built into the caps variants.

normal Condensed text fonts

- *-anttcl — Antykwa Toruńska Condensed Light-Regular
- *-anttcli — Antykwa Toruńska Condensed Light-Italic
- *-anttcm — Antykwa Toruńska Condensed Medium-Regular
- *-anttcmi — Antykwa Toruńska Condensed Medium-Italic
- *-anttcr — Antykwa Toruńska Condensed Regular
- *-anttcri — Antykwa Toruńska Condensed Italic
- *-anttcb — Antykwa Toruńska Condensed Bold
- *-anttcbi — Antykwa Toruńska Condensed BoldItalic

caps Condensed text fonts

- *-anttclcap — Antykwa Toruńska Condensed Caps Light-Regular
- *-anttclicap — Antykwa Toruńska Condensed Caps Light-Italic
- *-anttcmcap — Antykwa Toruńska Condensed Caps Medium-Regular
- *-anttcmicap — Antykwa Toruńska Condensed Caps Medium-Italic
- *-anttcrcap — Antykwa Toruńska Condensed Caps Regular
- *-anttcricap — Antykwa Toruńska Condensed Caps Italic
- *-anttcbcap — Antykwa Toruńska Condensed Caps Bold
- *-anttcbicap — Antykwa Toruńska Condensed Caps BoldItalic

All Antykwa Toruńska fonts contain the characters of the cyrillic alphabets.
The "wncy" encoding allows the use of ASCII characters for text entry into the TEX source code. For LATEX the "t2a", "t2b" and "t2c" encodings are used.

▷ **Cyrillic characters**
(* *denotes the encoding used:* wncy, t2a, t2b *or* t2c)

text fonts

- *-anttl — Antykwa Toruńska Cyrillic Light-Regular
- *-anttli — Antykwa Toruńska Cyrillic Light-Italic
- *-anttm — Antykwa Toruńska Cyrillic Medium-Regular
- *-anttmi — Antykwa Toruńska Cyrillic Medium-Italic
- *-anttr — Antykwa Toruńska Cyrillic Regular
- *-anttri — Antykwa Toruńska Cyrillic Italic
- *-anttb — Antykwa Toruńska Cyrillic Bold
- *-anttbi — Antykwa Toruńska Cyrillic BoldItalic

Condensed text fonts

- *-anttcl — Antykwa Toruńska Condensed Cyrillic Light-Regular
- *-anttcli — Antykwa Toruńska Condensed Cyrillic Light-Italic
- *-anttcm — Antykwa Toruńska Condensed Cyrillic Medium-Regular
- *-anttcmi — Antykwa Toruńska Condensed Cyrillic Medium-Italic
- *-anttcr — Antykwa Toruńska Condensed Cyrillic Regular
- *-anttcri — Antykwa Toruńska Condensed Cyrillic Italic
- *-anttcb — Antykwa Toruńska Condensed Cyrillic Bold
- *-anttcbi — Antykwa Toruńska Condensed Cyrillic BoldItalic

ΣΩαβγδ
ΣΩαβγδ
ΣΩαβγδ
ΣΩαβγδ

ΣΩαβγδ
ΣΩαβγδ
ΣΩαβγδ
ΣΩαβγδ

Antykwa Toruńska contains all characters from the basic greek alphabet. I do not provide the alternative characters known as the "polytonic" characters.

ABC abc

123 ⇐ ⇒ ⇑ ⇓ ♣ ♠ ♡ ◇ ≫ ≻ ≽ ∈ ↓ ↙ ↘ ↖ ∩ ⊎ ⊗ ⊙ ⊣ ℮

Antykwa Toruńska fonts contain also mathematical characters and various other symbols. They are not covered in the standard encoding tables. The fonts with the "exp" (expert) prefix were created to enable the use of such symbols.

▹ **greek characters** *(coded as cp1253)*

normal fonts

- greek-anttl — Antykwa Toruńska Greek Light-Regular
- greek-anttli — Antykwa Toruńska Greek Light-Italic
- greek-anttm — Antykwa Toruńska Greek Medium-Regular
- greek-anttmi — Antykwa Toruńska Greek Medium-Italic
- greek-anttr — Antykwa Toruńska Greek Regular
- greek-anttri — Antykwa Toruńska Greek Italic
- greek-anttb — Antykwa Toruńska Greek Bold
- greek-anttbi — Antykwa Toruńska Greek BoldItalic

Condensed fonts

- greek-anttcl — Antykwa Toruńska Condensed Greek Light-Regular
- greek-anttcli — Antykwa Toruńska Condensed Greek Light-Italic
- greek-anttcm — Antykwa Toruńska Condensed Greek Medium-Regular
- greek-anttcmi — Antykwa Toruńska Condensed Greek Medium-Italic
- greek-anttcr — Antykwa Toruńska Condensed Greek Regular
- greek-anttcri — Antykwa Toruńska Condensed Greek Italic
- greek-anttcb — Antykwa Toruńska Condensed Greek Bold
- greek-anttcbi — Antykwa Toruńska Condensed Greek BoldItalic

▹ **The Expert fonts** *(nonstandard encoding)*

- exp-anttcl — Antykwa Toruńska Exp Light-Regular
- exp-anttcli — Antykwa Toruńska Exp Light-Italic
- exp-anttcm — Antykwa Toruńska Exp Medium-Regular
- exp-anttcmi — Antykwa Toruńska Exp Medium-Italic
- exp-anttcr — Antykwa Toruńska Exp Regular
- exp-anttcri — Antykwa Toruńska Exp Italic
- exp-anttcb — Antykwa Toruńska Exp Bold
- exp-anttcbi — Antykwa Toruńska Exp BoldItalic

fonts/type1/public/antt

- the complete (each with over 1060 glyphs) PostScript files *.pfb, used by the TEX system drivers.

tex/latex/antt

- *.sty or *.fd files for LATEX users prepared by Petr Olšák and Marcin Woliński

tex/plain/antt

- *.tex files prepared by Petr Olšák

ANTYKWA TORUŃSKA IN LaTeX

1. AS THE DEFAULT DOCUMENT FONT.

To make Antykwa Toruńska the default font for the current document is easily done with the anttor package. Antykwa Toruńska is available in a number of layouts (character sets). The typesetting of English language documents can be done with the default OT1 encoding. Encodings in T1, T2 variants and OT2 are also available. Typesetting of Polish language documents can be done by enabling the OT4 layout (e.g., by loading the polski package). A richer subset of the font repertoire is available through the use of the QX layout:

```
\usepackage{polski,anttor}
\usepackage[QX]{fontenc}
```

Moreover, when calling the anttor package the options light and/or condensed are available to request the light and/or condensed variants of the type, respectively. The result of the standard directives \textit, \textbf, \textsc and their combinations for different package loading options is shown on the left.

2. IN DOCUMENT FRAGMENTS.

The LaTeX name for Antykwa Toruńska is antt. To typeset a text fragment with Antykwa the following commands are sufficient (no packages need to be loaded, Latin Modern is the default font):

```
If one examines subcapitalist deconstructivist theory, one
is faced with a choice: either reject Debordist situation or
{\fontfamily{antt}\selectfont conclude that reality must come
from the masses}, but only if the premise
```

> If one examines subcapitalist deconstructivist theory, one is faced with a choice: either reject Debordist situation or conclude that reality must come from the masses, but only if the premise

The light and/or condensed variants may be obtained by giving relevant arguments to \fontseries:

```
If one examines subcapitalist deconstructivist theory, one
is faced with a choice: either reject Debordist situation or
{\fontfamily{antt}\fontseries{l}\selectfont conclude that
reality must come {\fontseries{bc}\selectfont
from the masses}, but only if the premise
```

> If one examines subcapitalist deconstructivist theory, one is faced with a choice: either reject Debordist situation or conclude that reality must come **from the masses**, but only if the premise

It is easier to use the *family* name to select the light and/or condensed variants:

antt	normal
anttl	light
anttc	condensed
anttlc	light condensed

The description of the \fontfamily and \fontseries commands can be found in the fntguide.tex document which is distributed with LaTeX.

LaTeX source:

```
\textsc{If one examines
\textbf{subcapitalist}
deconstructivist} theory, one
is faced with a choice: either
reject \textbf{Debordist
situation or \textsc{conclude
that reality must come from the
\textit{masses}}} , but only if
the premise of postsemantic
theory is invalid; otherwise,
\textit{Bataille's model
\textbf{of cultural}narrative
is one of ``capitalist
discourse'', and hence
intrinsically elitist.}
```

```
\usepackage{anttor}
```

If one examines **subcapitalist** deconstructivist theory, one is faced with a choice: either reject **Debordist situation or conclude that reality must come from the *masses***, but only if the premise of postsemantic theory is invalid; otherwise, *Bataille's model **of cultural** narrative is one of "capitalist discourse", and hence intrinsically elitist.*

```
\usepackage[light]{anttor}
```

If one examines **subcapitalist** deconstructivist theory, one is faced with a choice: either reject **Debordist situation or conclude that reality must come from the *masses***, but only if the premise of postsemantic theory is invalid; otherwise, *Bataille's model **of cultural** narrative is one of "capitalist discourse", and hence intrinsically elitist.*

```
\usepackage[condensed]
                {anttor}
```

If one examines **subcapitalist** deconstructivist theory, one is faced with a choice: either reject **Debordist situation or conclude that reality must come from the *masses***, but only if the premise of postsemantic theory is invalid; otherwise, *Bataille's model **of cultural** narrative is one of "capitalist discourse", and hence intrinsically elitist.*

```
\usepackage[light,
      condensed]{anttor}
```

If one examines **subcapitalist** deconstructivist theory, one is faced with a choice: either reject **Debordist situation or conclude that reality must come from the *masses***, but only if the premise of postsemantic theory is invalid; otherwise, *Bataille's model **of cultural** narrative is one of "capitalist discourse", and hence intrinsically elitist.*

ANTYKWA TORUŃSKA IN PLAIN

The standard font declarations in the plain format look like this:

```
\font\rm qx-anttr at 9pt
\font\bf qx-anttb at 9pt
\font\it qx-anttri at 9pt
```

Now, the declared fonts might be used as follows:

```
\rm If one examines subcapitalist deconstructivist theory, one
is faced with a~choice: {\it either reject Debordist situation
or conclude that reality must come} from the masses, {\bf but
only if the premise of postsemantic theory is invalid};
otherwise, Bataille's model of cultural
```

If one examines subcapitalist deconstructivist theory, one is faced with a choice: *either reject Debordist situation or conclude that reality must come* from the masses, **but only if the premise of postsemantic theory is invalid**; otherwise, Bataille's model of cultural

For special purposes more elaborate macro definitions might be employed, e.g.:

```
\newdimen\sizeoffont
\def\declarefont#1#2{% #1: TeX name; #2:  TFM file name
 \expandafter\def\csname #1\endcsname
  {\expandafter\afterassignment\csname #1_\endcsname
   \sizeoffont=}%
 \expandafter\def\csname #1_\endcsname
  {\font\currfont #2 at \sizeoffont\relax \currfont}%
 \baselineskip=1.2\sizeoffont
}
\declarefont{anttn}{qx-anttr}
\declarefont{anttb}{qx-anttb}
\declarefont{antti}{qx-anttri}
```

which in the text look like this:

```
\anttn9pt If one examines subcapitalist deconstructivist
theory, one is faced with a~choice: {\antti9pt either reject
Debordist situation or conclude that reality must come} from
the masses, {\anttb9pt but only if the premise of postsemantic
theory is invalid}; otherwise, Bataille's model of cultural
```

It is worthwhile to get familiar with Petr Olšák's OFS package. It immensely easies font handling in plain and LaTeX documents. Here is an example of how to use OFS:

```
% input the package
\input ofs [pantyk]
% or in LaTeX
\usepackage [pantyk] {ofs}
% Cork encoding
\def\fotenc{8t}
% define the document
% default font
\setfonts
      [AntykwaToruńska/9pt]
```

```
Antykwa Toruńska with OFS
can be used as the default
font in the following eight
variants: {\lr Light},
Regular, {\mr Medium},
{\bf Bold}, {\li Light
Italic}, {\it Italic},
{\mi Medium Italic}, {\bi
Bold Italic}.
```

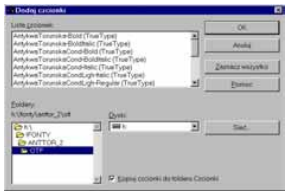
Antykwa Toruńska with OFS can be used as the default font in the following eight variants: Light, Regular, **Medium**, **Bold**, *Light Italic*, *Italic*, ***Medium Italic***, ***Bold Italic***.

Additional font variants can be defined, e.g.:

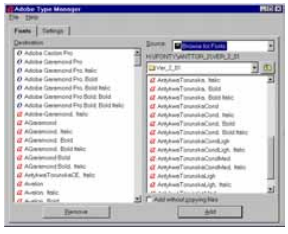
```
% Caps font Medium
\fontdef\cscmr
[AntykwaToruńskaCaps-mr/9pt]
% Title font Light
\fontdef\title [!-lr/16pt]
% Small font Regular
\fontdef\small [!/6pt]
```

The OFS package can do a lot more. I encourage you to read the documentation:

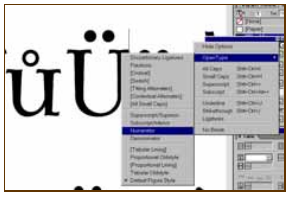
`ftp:/math.feld.cvut.cz/pub/olsak/ofs/`



The MS Windows dialog window "Adding fonts"



The ATM (Adobe Type Manager) dialog window for managing PostScript fonts.



Selecting font features in the InDesign program.

INSTALLING IN WINDOWS SYSTEMS

1. WINDOWS XP

True Type fonts (from AntykwaTorunska-ttf-2_03.zip) and OpenType (from AntykwaTorunska-otf-2_03.zip) with the complete font set can be installed in the fonts dialog window: *Start* → *Settings* → *Control Panel* → *Fonts*.

Thanks to Unicode, typesetting in languages based on the latin, cyrillic and greek alphabets is possible. Mathematical characters and symbols, nautical digits and small caps are also available. Following features are built into OpenType fonts:

- `sups` — *Superior*
- `sinf` — *Inferior*
- `numr` — *Numerators*
- `dnom` — *Denominators*
- `cpsp` — *Capital Spacing*
- `onum` — *Old Style Numerals*
- `smcp` — *Small Capitals*
- `liga` — *Standard Ligatures*
- `dlig` — *Discretionary Ligatures*
- `frac` — *Fractions*

The availability of these features depends upon application support, e.g., the Adobe InDesign program offers all of them.

2. WINDOWS 98 and older versions

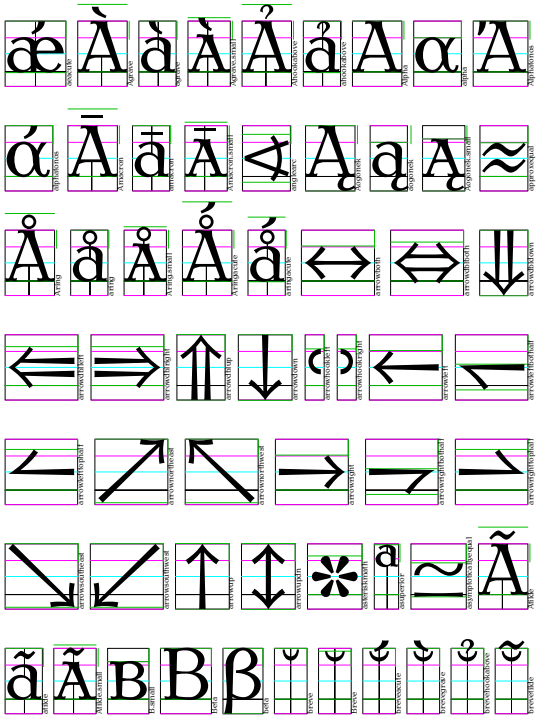
The True Type fonts (.ttf) may be installed in a similar manner. For the installation of Type 1 fonts (file AntykwaTorunska-type1-2_03.zip) the Adobe Type Manager program which manages PostScript fonts is required. These fonts contain 256 characters each in the following encodings: standard encoding, standard encoding small caps, cp1250, cp1250 small caps, qx, qx small caps, cp1253, cyrylic. I do not recommend OpenType fonts with this operating systems as there is no adequate multilanguage support.

3. Other operating systems

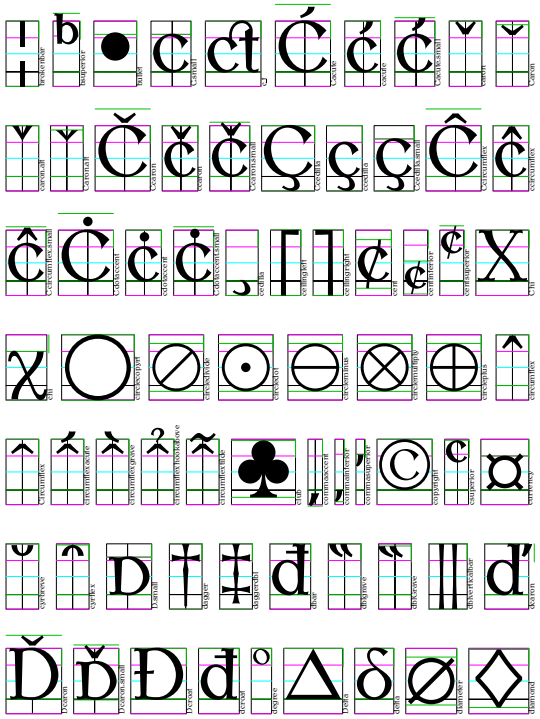
Antykwa Toruńska fonts can also be installed in and used with those operating systems which support the OpenType format, as e.g., Linux (X-Window version 4 and higher) or Apple Mac OS (version X and higher).

10

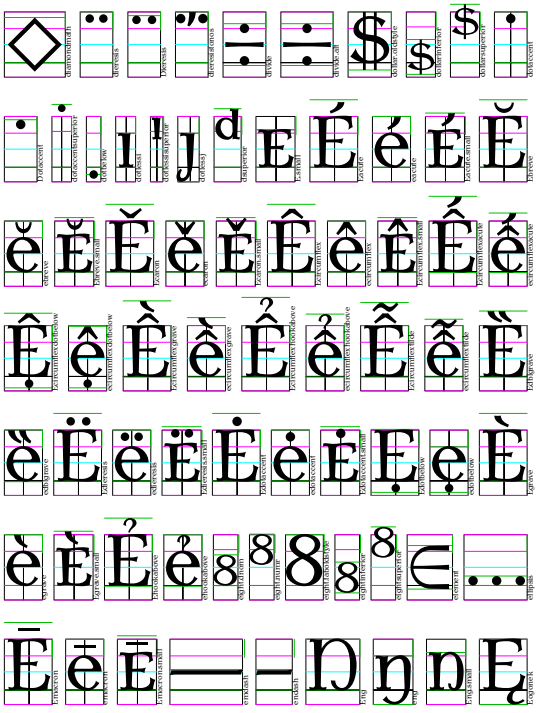
11



12

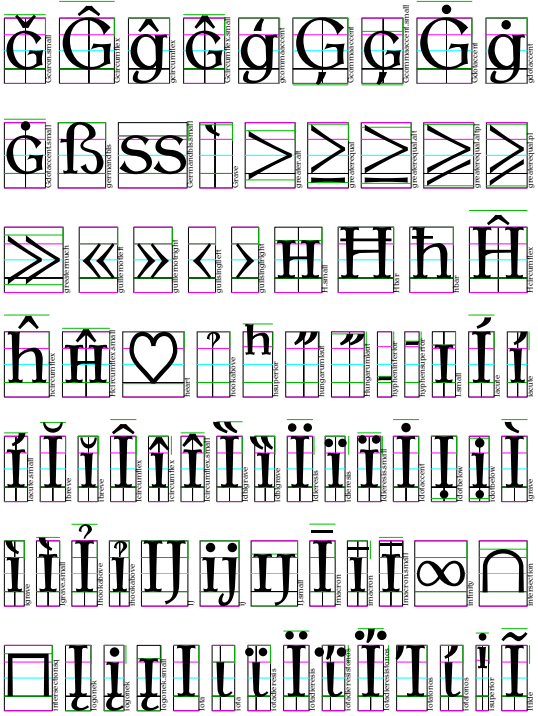


13



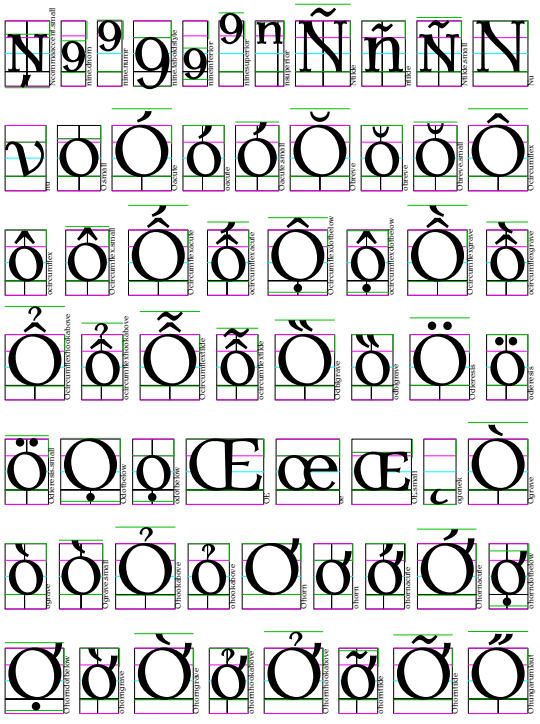
14

15



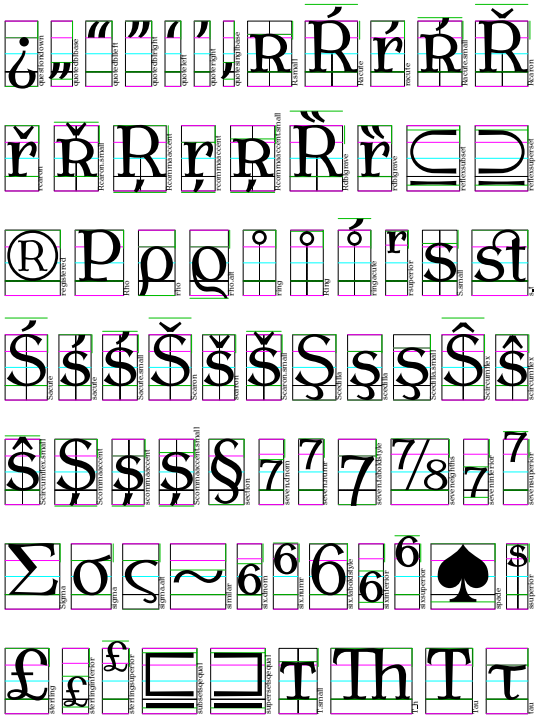
16

17

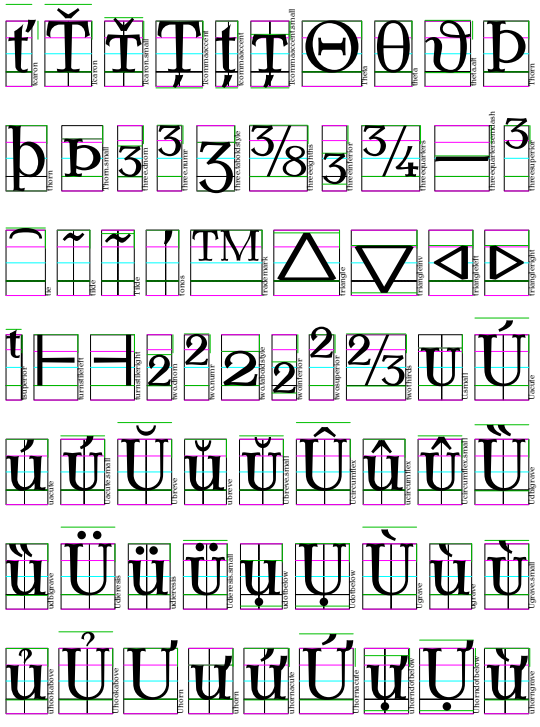


18

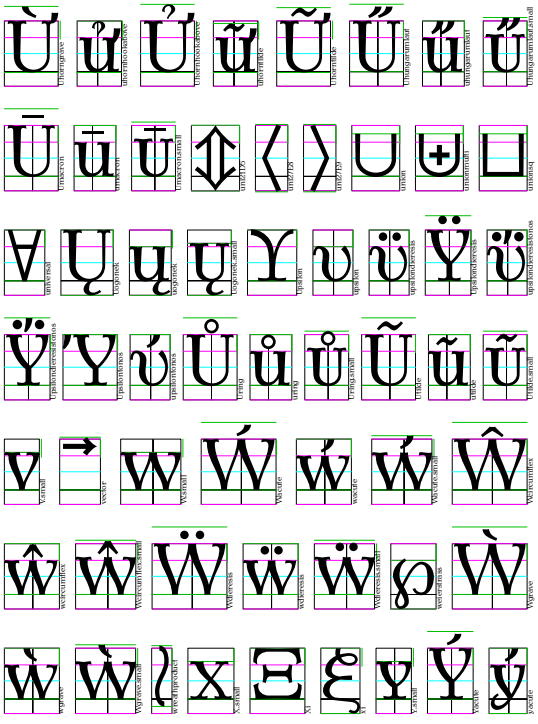
19



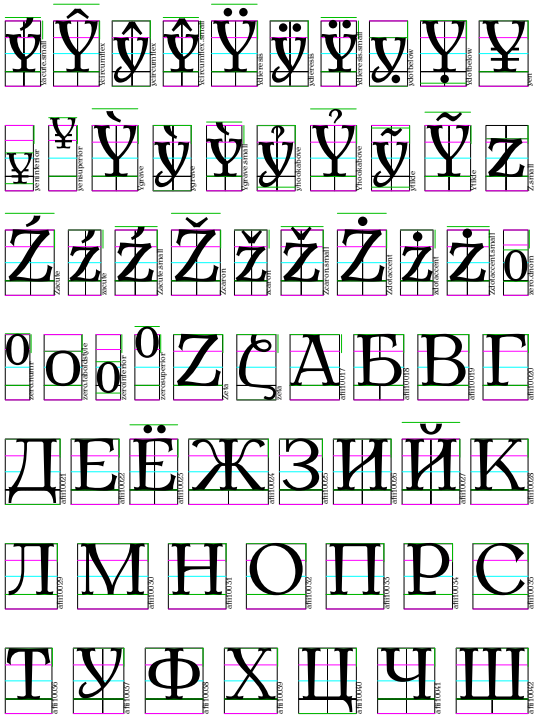
20



21

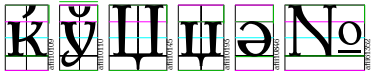


22



23

24



25

Characters in the qx-anttr font, with decimal codes:

0: α	1: Δ	2: β	3: δ	4: π	5: Π	6: Σ	7: µ	8: …
9: fk	10: Ω	11: ff	12: fi	13: fl	14: ffi	15: ffl	16: ı	17: ȷ
18: `	19: ´	20: ˇ	21: ˘	22: ¯	23: °	24: ¸	25: ß	26: æ
27: œ	28: ø	29: Æ	30: Œ	31: Ø	32:	33: !	34: ”	35: #
36: $	37: %	38: &	39: ’	40: (	41:)	42: *	43: +	44: ,
45: -	46: .	47: /	48: 0	49: 1	50: 2	51: 3	52: 4	53: 5
54: 6	55: 7	56: 8	57: 9	58: :	59: ;	60: ¡	61: =	62: ¿
63: ?	64: @	65: A	66: B	67: C	68: D	69: E	70: F	71: G
72: H	73: I	74: J	75: K	76: L	77: M	78: N	79: O	80: P
81: Q	82: R	83: S	84: T	85: U	86: V	87: W	88: X	89: Y
90: Z	91: [	92: “	93:]	94: ^	95: ˙	96: ‘	97: a	98: b
99: c	100: d	101: e	102: f	103: g	104: h	105: i	106: j	107: k
108: l	109: m	110: n	111: o	112: p	113: q	114: r	115: s	116: t
117: u	118: v	119: w	120: x	121: y	122: z	123: –	124: —	125: ˝
126: ˜	127: ¨	128: €	129: Ą	130: Ć	131: >	132: ≥	133: ≈	134: Ę
135: Į	136: <	137: ≤	138: Ł	139: Ń	140: ~	141: ^	142:	143: †
144: ‡	145: Ś	146: Š	147: Ş	148: °	149: Ţ	150: ¸	151: Ų	152: Ÿ
153: Ź	154: Ž	155: Ż	156: IJ	157: {	158: }	159: §	160:	161: ą
162: ć	163: ®	164: ©	165: ÷	166: ę	167: į	168: –	169: ×	170: ł
171: ń	172: ±	173: ∞	174: «	175: »	176: ¶	177: ś	178: š	179: ş
180: •	181: ţ	182: —	183: ų	184: ÿ	185: ź	186: ž	187: ż	188: ij
189: ·	190: "	191: '	192: À	193: Á	194: Â	195: Ã	196: Ä	197: Å
198: \	199: Ç	200: È	201: É	202: Ê	203: Ë	204: Ì	205: Í	206: Î
207: Ï	208: Đ	209: Ñ	210: Ò	211: Ó	212: Ô	213: Õ	214: Ö	215: ¤
216: ‰	217: Ù	218: Ú	219: Û	220: Ü	221: Ý	222: Þ	223: |	224: à
225: á	226: â	227: ã	228: ä	229: å	230: _	231: ç	232: è	233: é
234: ê	235: ë	236: ì	237: í	238: î	239: ï	240: ð	241: ñ	242: ò
243: ó	244: ô	245: õ	246: ö	247: ◁	248: ∅	249: ù	250: ú	251: û
252: ü	253: ý	254: þ	255: „					

Characters in the cork-anttr font, with decimal codes:

0: `	1: ´	2: ˆ	3: ˜	4: ¨	5: ˝	6: °	7: ˇ	8: ˘
9: ¯	10: ˙	11: ¸	12: ˛	13: ,	14: ‹	15: ›	16: “	17: ”
18: „	19: «	20: »	21: –	22: —	23:	24:	25: ı	26: ȷ
27: ff	28: fi	29: fl	30: ffi	31: ffl	32:	33: !	34: "	35: #
36: $	37: %	38: &	39: ’	40: (	41:)	42: *	43: +	44: ,
45: -	46: .	47: /	48: 0	49: 1	50: 2	51: 3	52: 4	53: 5
54: 6	55: 7	56: 8	57: 9	58: :	59: ;	60: <	61: =	62: >
63: ?	64: @	65: A	66: B	67: C	68: D	69: E	70: F	71: G
72: H	73: I	74: J	75: K	76: L	77: M	78: N	79: O	80: P
81: Q	82: R	83: S	84: T	85: U	86: V	87: W	88: X	89: Y
90: Z	91: [	92: \	93:]	94: ^	95: _	96: ‘	97: a	98: b
99: c	100: d	101: e	102: f	103: g	104: h	105: i	106: j	107: k
108: l	109: m	110: n	111: o	112: p	113: q	114: r	115: s	116: t
117: u	118: v	119: w	120: x	121: y	122: z	123: {	124: |	125: }
126: ~	127:	128: Ă	129: Ą	130: Ć	131: Č	132: Ď	133: Ě	134: Ę
135: Ğ	136: Ĺ	137: Ľ	138: Ł	139: Ń	140: Ň	141: Ŋ	142: Ő	143: Ŕ
144: Ř	145: Ś	146: Š	147: Ş	148: Ť	149: Ţ	150: Ű	151: Ů	152: Ÿ
153: Ź	154: Ž	155: Ż	156: IJ	157: İ	158: đ	159: §	160: ă	161: ą
162: ć	163: č	164: ď	165: ě	166: ę	167: ğ	168: Í	169: ľ	170: ł
171: ń	172: ň	173: ŋ	174: ő	175: ŕ	176: ř	177: ś	178: š	179: ş
180: ť	181: ţ	182: ű	183: ů	184: ÿ	185: ź	186: ž	187: ż	188: ij
189: ¡	190: ¿	191: £	192: À	193: Á	194: Â	195: Ã	196: Ä	197: Å
198: Æ	199: Ç	200: È	201: É	202: Ê	203: Ë	204: Ì	205: Í	206: Î
207: Ï	208: Đ	209: Ñ	210: Ò	211: Ó	212: Ô	213: Õ	214: Ö	215: Œ
216: Ø	217: Ù	218: Ú	219: Û	220: Ü	221: Ý	222: Þ	223:	224: à
225: á	226: â	227: ã	228: ä	229: å	230: æ	231: ç	232: è	233: é
234: ê	235: ë	236: ì	237: í	238: î	239: ï	240: ð	241: ñ	242: ò
243: ó	244: ô	245: õ	246: ö	247: œ	248: ø	249: ù	250: ú	251: û
252: ü	253: ý	254: þ	255: ß					

Variabele faxdocumenten aanmaken in LaTeX

Abstract
Beschreven wordt hoe LaTeX gebruikt wordt als schakel tussen een bestaande bedrijfsapplicatie en een faxserver; data worden automatisch geëxporteerd naar een LaTeX-document.

Keywords
Latex, Latex2RTF, macro's, faxen.

storm en storingsbevestigingen

De applicatie storm (STOringsRegistratie MSO) zorgt voor centrale verwerking van een groot aantal gegevens van ons bedrijf MSO[1]. MSO is een technische service-organisatie op ICT-gebied. Als klassieke *Third Party Maintenance*-organisatie werkt het bedrijf in opdracht van resellers, dealerorganisaties, systeem- en softwarehuizen, importeurs en fabrikanten. MSO houdt zich bezig met dienstverlening op het gebied van automatisering in de ruimste zin van het woord.

Iedere activiteit van MSO in opdracht van een klant – doorgaans gaat het om reparaties op computer- of randapparatuur van *klanten van een dealer* – resulteert in een storingsaanmelding. Van een reparatieopdracht wil de klant graag de feitelijke gegevens weten en daarvoor worden zgn. storingsbevestigingen verstuurd, per e-mail of per fax. Naast het verwerken van allerlei administratieve handelingen die het verrichten van reparaties met zich meebrengt wordt ook het versturen van de storingsbevestigingen door storm uitgevoerd. Bij het sturen van de storingsbevestiging werd tot nu toe een eenvoudig tekstje samengesteld met daarin de elementaire gegevens over de pas aangenomen reparatieopdracht (het serienummer, de debiteur- of dealergegevens, het storingsnummer en nog enkele aanvullende gegevens). Deze meldingen worden vaak per e-mail verstuurd, maar er zijn genoeg klanten die ze liever als fax krijgen: ze hebben dan meteen een 'hard copy'.

Mooiere teksten gaan faxen

Het bedrijf wil graag het uiterlijk van deze faxoutput verbeteren. Het project dat hiervoor is begonnen resulteerde ondermeer in dit artikel. Het onderwerp ligt enigszins in het verlengde van het artikel in maps nummer 30 van Roland Smith, maar de aanpak is anders.

De faxen worden verfraaid door de tekst uit te breiden en uit te voeren met een mooier lettertype. Daarnaast moeten er meer gegevens over de reparatie in vermeld worden – type van het apparaat, adressen, serienummers en tarieven – maar vooral: het geheel moet er meer *als een zakelijke brief* uitzien, inclusief het bedrijfslogo.

Voorlopige eerste keus: Latex2rtf

Binnen MSO gebruikt men MS-Office en de faxserver maakt voor de opmaak gebruik van MS-Word. Daarmee ligt rtf (*Rich Text Format*) als 'algemeen' documentformaat enigszins voor de hand. Rtf laat zich qua tekstindeling niet gemakkelijk programmeren, en de code ervan is evenmin bedoeld om door 'gewone mensen' gelezen of aangepast te worden.

Ik had eerder al het bestaan ontdekt van Latex2rtf en het leek mij geschikt voor deze toepassing: je kunt er een LaTeX-tekst mee omzetten naar rtf. In tweede instantie bleek Latex2rtf – hoewel het voor een eenvoudige omzetting verrassend goed werkt – uiteindelijk minder bruikbaar. De eerste resultaten met een bestaand tekstje waren hoopgevend, en het resultaat is wat je er van zo'n lichte applicatie mag verwachten. Maar vrij vlot bleken de beperkingen: `\hfill` en soortgelijke commando's werken niet; je kunt plaatjes invoegen, maar de eps-bitmaps en jpg's bleven onzichtbaar, alleen een plaatje van het type png bleek te werken; in de `tabular`-omgeving worden de kolommen allemaal even breed, ongeacht de tekst in de kolom. Opvallend is dat je met het gebruik van Latex2rtf uiteindelijk LaTeX zelf helemaal niet nodig hebt: Latex2rtf kan met een uitgetest LaTeX-document op eigen houtje de rtf-file aanmaken.

Toen later bleek dat een rtf-bestand voor het faxen helemaal niet nodig was en de faxserver zeker zo goed met pdf overweg zou kunnen koos ik ervoor pdf-LaTeX te installeren met behulp van de TeX-live-cd.

De methode van datatransport naar de LaTeX-tekst

De taal waarin storm is geschreven is Today, en de huidige uitbreidingen worden ontwikkeld in BuildProfessional[2], een ontwikkelomgeving die deels op Linux, deels op een Windows-pc 'draait'. In de applicatie storm is voor de opmaak van het faxdocument een functie (= subroutine) gemaakt (`F-bevestig_nw`) die er voor zorgt dat de data van de storing worden omgezet naar LaTeX. De gegevens van de klant en van de door de MSO-technici uit te voeren acties zijn bij de aanmelding van de storing in de storm-database opgenomen (meestal in ascii-vorm) en een aantal van deze gegevens moet in de fax weergegeven worden: om welk 'serienummer' het gaat dat gerepareerd wordt, naam van de klant, enzovoort. Om deze gegevens automatisch in een LaTeX-document te kunnen opnemen heb ik een paar macro's gedefinieerd die het overbrengen van de data uit storm op een tamelijk eenvoudige manier mogelijk maken.

Hoe je een macro moet opzetten weet waarschijnlijk iedere lezer, daarom ga ik hier alleen in op enkele details:

- je kunt geen cijfers (en ook geen bijzondere tekens `_ # % - = .`) in de naam van een macro opnemen; dat is jammer, want dat betekent dat je de namen van variabelen uit de programmatuur niet zo maar kunt overnemen; voor de meeste ervan moet voor de resulterende macro een aangepaste naam bedacht worden
- in de functie wordt per overgedragen data-item één tekstregel in een hulpbestand gezet
- iedere regel uit dit bestandje resulteert bij inlezen in het LaTeX-bestand in een macro.

Uit de functie `F-bevestig_nw`, die de benodigde macro's voorbereidt, heb ik één regel weergegeven om een indruk te geven hoe de belangrijkste code van dit functieonderdeel er uit ziet:

```
F-omschr_99.bevestig = "\N{\kostprijsvrk}{" F-kostprijs_vrk.1000par "}"; »
  FILE *INSERT bevestig
```

en dit levert bij inlezen in LaTeX de volgende code:

```
\N{\kostprijsvrk}  {" F-kostprijs_vrk.1000par "}
```

waarbij `F-kostprijs_vrk.1000par` een getal is, dus eigenlijk:

```
\N{\kostprijsvrk}  {  75.00}
```

Resultaat is de macro `\kostprijsvrk` waarmee het bedrag van de voorrijkosten in de tekst van de fax gezet kan worden. De verzameling van de data voor de fax in het hulpbestand ziet er bijvoorbeeld zo uit:

```
\N{\Wcompanyfax}     {Unirein}
\N{\Womschra}        {mee ter rep intern}
\N{\Womschrb}        {dit is regel twee v.d. melding van de klant}
\N{\kostprijsarbeid} {   80.00}
\N{\diagnosekosten}  {   80.00}
\N{\kostprijsvrk}    {   75.00}
\N{\verzendkosten}   {   22.00}
\N{\bezorgkosten}    {   22.00}
\N{\Wfromfax}        {naam van de faxverzender}
\N{\Wtofax}          {Hr. Janssen}
\N{\Wsubjectfax}     {storingsbevestiging 205040093}
\N{\Wfaxfax}         {0522-665150}
\N{\Womschrhelp}     {storingsbevestiging}
\N{\storingsnr}      {205040093}
\N{\ordernr}         {}
\N{\melddatum}       {2005-04-01}
\N{\debnr}           {48444}
\N{\facnaamdv}       {Unirein}
\N{\facadres}        {Postbus}
\N{\fachuisnr}       {786 }
\N{\facpostcode}     {1222AK}
\N{\facplaats}       {Heerlen}
\N{\werkadres}       {  1}
\N{\serienr}         {7902LH411074-V}
\N{\typeom}          {3C17700-ME SS3 4900 12p}
\N{\omschrijving}    {LAN Hub Ethernet}
\N{\omschrijvingt}   {3Com}
```

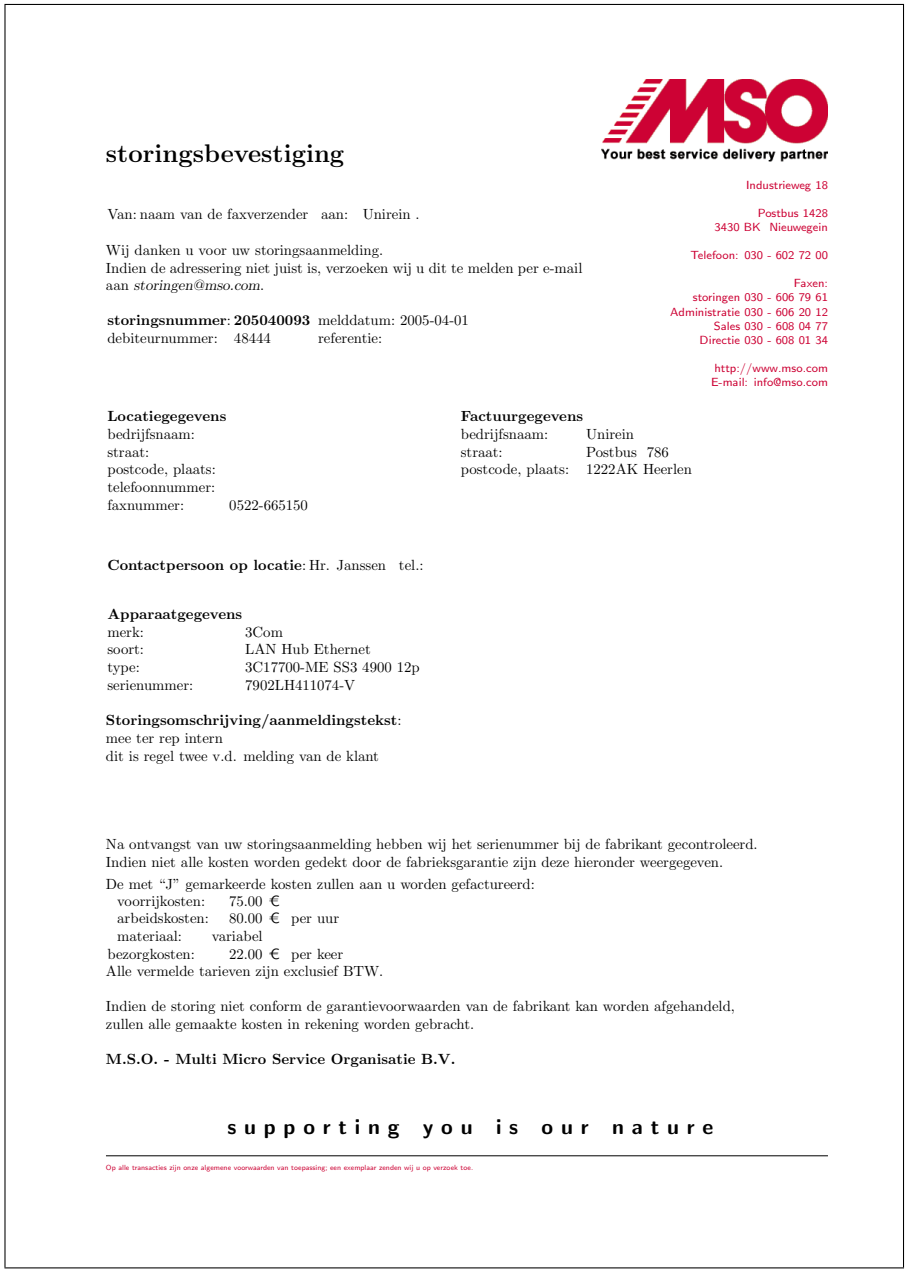
Met een eenvoudige systeemaanroep in `F-bevestig_nw` worden de gegevens uit `bevdat205040093.tex` samengevoegd met `bevorg.tex` door aanpassing van het `\input{bevdat.tex}`-commando daarin, en weggeschreven naar de latex-source voor de fax, `fax205040093.tex`:

```
sed 's/{bevdat.tex}/{bevdat205040093.tex}/' bevorg.tex > fax205040093.tex
```

`fax205040093.tex` is hierbij het bestand dat naar de te faxen pdf-file wordt omgezet. Bij `\input{bevdat205040093.tex}` wordt dus per getal (of tekststring) een macro gedefinieerd, en bij toepassing ervan verderop in het document wordt op die plaats het getal – de 'waarde' van de macro – in de tekst opgenomen. Het commando

```
\newcommand{\N}[2]{\newcommand{#1}{#2}}
```

in het hierna besproken LaTeX-bestand is hier het 'werkpaard' bij het overbrengen van de gegevens uit de applicatie naar de faxbrief.



storingsbevestiging

Your best service delivery partner

Industrieweg 18
Postbus 1428
3430 BK Nieuwegein

Telefoon: 030 - 602 72 00

Faxen:
storingen 030 - 606 79 61
Administratie 030 - 606 20 12
Sales 030 - 608 04 77
Directie 030 - 608 01 34

http://www.mso.com
E-mail: info@mso.com

Van: naam van de faxverzender aan: Unirein .

Wij danken u voor uw storingsaanmelding.
Indien de adressering niet juist is, verzoeken wij u dit te melden per e-mail aan *storingen@mso.com*.

storingsnummer: **205040093** melddatum: 2005-04-01
debiteurnummer: 48444 referentie:

Locatiegegevens
bedrijfsnaam:
straat:
postcode, plaats:
telefoonnummer:
faxnummer: 0522-665150

Factuurgegevens
bedrijfsnaam: Unirein
straat: Postbus 786
postcode, plaats: 1222AK Heerlen

Contactpersoon op locatie: Hr. Janssen tel.:

Apparaatgegevens
merk: 3Com
soort: LAN Hub Ethernet
type: 3C17700-ME SS3 4900 12p
serienummer: 7902LH411074-V

Storingsomschrijving/aanmeldingstekst:
mee ter rep intern
dit is regel twee v.d. melding van de klant

Na ontvangst van uw storingsaanmelding hebben wij het serienummer bij de fabrikant gecontroleerd.
Indien niet alle kosten worden gedekt door de fabrieksgarantie zijn deze hieronder weergegeven.
De met "J" gemarkeerde kosten zullen aan u worden gefactureerd:
voorrijkosten: 75.00 €
arbeidskosten: 80.00 € per uur
materiaal: variabel
bezorgkosten: 22.00 € per keer
Alle vermelde tarieven zijn exclusief BTW.

Indien de storing niet conform de garantievoorwaarden van de fabrikant kan worden afgehandeld, zullen alle gemaakte kosten in rekening worden gebracht.

M.S.O. - Multi Micro Service Organisatie B.V.

supporting you is our nature

Op alle transacties zijn onze algemene voorwaarden van toepassing; een exemplaar zenden wij u op verzoek toe.

Figuur 1. Het eindresultaat zoals het gefaxt (of als attachment geë-maild) kan worden.

Het basisbestand voor de LaTeX-storingsbevestiging

Het bestand `bevorg.tex` is feitelijk de volledige faxbrief *zonder de gegevens van de storingsbevestiging* en bevat alle gangbare delen van een LaTeX-bestand. Ik heb hiervan de eerste vijftien regels weergegeven:

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

```
\newcommand{\N}[2]{\newcommand{#1}{#2}}

\input{bevdat.tex} % bevdat.tex vervangen door bijv. bevdat205040093.tex

\begin{document}

\begin{minipage}[t]{\textwidth}
{\LARGE{\textbf{\Womschrhelp}}}%
\hfill \includegraphics[scale=.5]{MSOlogo3m.png}\\

\end{minipage}

\begin{minipage}[t]{.66\textwidth}
\begin{tabular}{llll}
Van:            & \Wfromfax  \ \ & aan:          \ \   & \Wfaxfax.\\
\end{tabular}
```

Iedere nieuwe storingsbevestiging wordt met het input-commando voorzien van de 'eigen' gegevens en tot een faxbrief verwerkt. Voorbeelden in dit fragment zijn `\Womschrhelp` en `\Wfaxfax` die vanuit het tussenbestand gevuld zijn met resp. de tekst "storingsbevestiging" en "0522-665150". Aan het slot van de storm-functie `F-bevestig_nw` wordt de verwerking van pdfLaTeX gestart met

```
pdflatex fax205040093.tex
```

Van het resultaat `\fax205040093.pdf` geeft figuur 1 een verkleinde weergave. Er blijven nog wel wat zaken op te lossen:

- onderscheppen van bepaalde tekens in namen van klanten "V&D" of misschien "_" of "@"
- wat gebeurt er als de LaTeX-compilatie – of een ander onderdeel van de documentopbouw – mislukt?

In de huidige testfase lopen de hier beschreven stappen vlekkeloos. De gebruiker van storm start de opdracht, en binnen een paar seconden komt het voorbeeld van het te versturen document in Acrobat Reader in beeld. Als de inhoud akkoord is kan de fax of de e-mail meteen verstuurd worden.

Ik hoop dat andere gebruikers met de hier beschreven oplossing hun voordeel kunnen doen.

Voetnoten

1. M.S.O. - Multi Micro Service Organisatie BV, http://www.mso.com
2. zie http://www.todaysystems.com.au

Ernst van der Storm
evdstorm@mso.com

Stroomdiagrammen maken met flow

Abstract
Flow is een handig programma om stroomdiagrammen te maken; via `\write18` kan het vanuit een LaTeX-document worden aangeroepen zodat eventuele wijzigingen vanzelf tevoorschijn komen.

Keywords
Latex, flow, write18, stroomdiagram, softwaredocumentatie.

Hoe maak je stroomdiagrammen met flow?

Als onderdeel van het vorige artikel wilde ik graag een schemaatje opnemen. Ik stuitte bij het zoeken naar een eenvoudig programma voor het maken van stroomschema's op flow[1], een 'taaltje' ontwikkeld door *Terry Brown* dat is voorzien van alle basiselementen van een stroomdiagram, maar vooral van een aantrekkelijk eenvoudige syntax (zie pagina 18 van *The LaTeX Graphics Companion* Goossens, Rahtz, Mittelbach, 1997).
De beschikbare elementen zijn: `Box`, `Oval`, `Choice`, `Tilt` en `Text`. Het laatste element is een box zonder rand.
Het is voor de opzet van een diagram niet nodig coördinaten op te geven voor de grootte van de elementen, maar voor de afmetingen en verplaatsingen die wél *afwijken* van de standaard moet je getallen invoeren.

De vorm van de elementen (grootte en hoogte/breedte-verhouding) kun je ook met eenvoudige getallen bepalen. Het is belangrijk vooraf de eenheid voor die getallen vast te leggen, bijvoorbeeld met:

```
\setlength{\unitlength}{2em}.
```

In de elementen kun je – al dan niet gecentreerd – tekst opnemen, en alle attributen die je in LaTeX kunt gebruiken zijn ook in flow bruikbaar. Vanzelfsprekend is het nodig er op te letten dat de tekst die binnen de randen van een box moet worden weergegeven niet te groot is, met andere woorden: de grootte van de box en de grootte van de tekst moeten op elkaar afgestemd zijn.

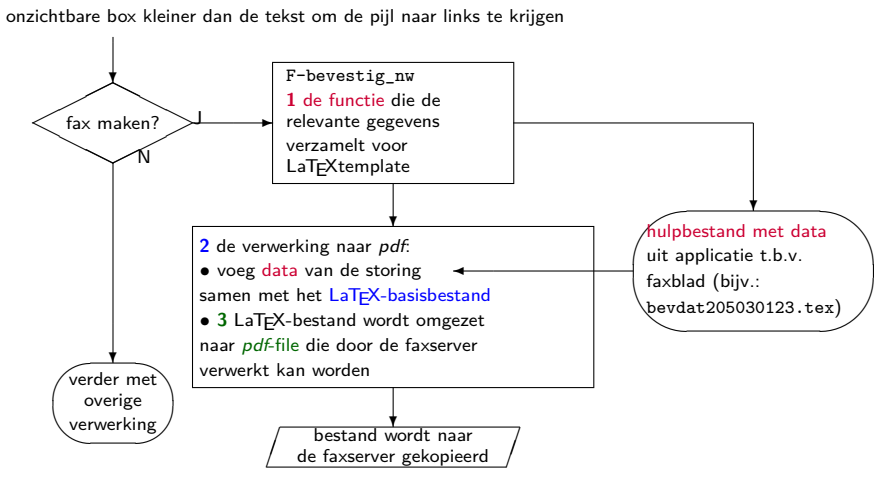
Er zijn drie types voor de verbindingslijnen tussen de elementen, namelijk `line`, `arrow` en `none`, die kun je instellen met commando `Settrack`. De richting in het schema geef je aan met respectievelijk `Right`, `Left`, `Down` en `Up`. Door te manipuleren met de `\unitlength` is een afbeelding eventueel te schalen:

```
\newlength{\um}\setlength{\um}{1bp}
\setlength{\unitlength}{15\um} % gebruik 19\um voor storm.tex.
```

Doorgaans moet het font in het stroomdiagram dan meegeschaald worden, bijvoorbeeld door `\scriptsize` te kiezen in plaats van `\normalsize`. Het resultaat is een `picture` dat zich naadloos laat opnemen in het LaTeX-document.

Vooraf ter illustratie een stukje flow:

```
TxtPos [l] [l] ~~
Text 6 1
  onzichtbare box kleiner dan de tekst om de pijl naar links te krijgen
Down .15
SetTrack Arrow
TxtPos [c] [c]
Choice . . J N 4 2
  fax maken?
Tag
Down 4
Oval 3 2
```



Met een `Tag` en `ToTag` kun je herhalingen construeren, of 'teruggaan' naar een aftakking. Een en ander is in flow.pdf voldoende gedocumenteerd.

Om dit goed werkend te krijgen moesten er wel een paar hindernissen genomen worden...

Hoe krijg je flow aan de praat?

Ik beschrijf de stappen die ik achtereenvolgens moest zetten. Let wel: ik werk doorgaans op een pc met Windows-XP dus de voorbeelden zijn daaraan ontleend:

- ☐ bereid aan de hand van een voorbeeld uit de documentatie een proef-documentje bijvoorbeeld `proef1.flo`
- ☐ uitproberen, vanaf de commandline: `flow proef1.flo proef1.tex`
- ☐ het resultaat is een stukje picture-code, dat in principe meteen ingevoegd kan worden in je document
- ☐ als blijkt dat in de eerste regel van het op deze manier gemaakte plaatje het volgende staat:

  `\begin{picture}(2989276266496.000000,2.000000)(0.000000,-2.000000)`

  dan levert dat bij verwerken van het picture een foutmelding op; oplossing is:
- ☐ flow.exe upgraden naar 0.99e; de laatste versie die voor gebruik in MS-dos/-

Windows kant en klaar beschikbaar was is versie 0.99b

- er zit in dit geval niets anders op dan met een geschikte compiler flow.c zelf te compileren (ik gebruikte MYSYS en Mingw); hoe dit gaat valt buiten het bestek van dit artikel.

Voor een ander doel had ik al vaak de behoefte gehad om tegelijk met een LaTeX-run een extern commando te laten uitvoeren – als bijvoorbeeld op een inhoudsopgave een externe bewerking gedaan moet worden – en daarvoor zocht ik naar een bruikbare en 'nette' oplossing.

Voor dit artikel moet na iedere *wijziging* van `schema2.flo` het flow-diagrammetje opnieuw omgezet worden naar het picture.
Dit is mogelijk met een `\write18{<externe opdracht>}`; de externe opdracht kan ieder commando zijn dat normaal op de command line van Windows of Linux ingetypt wordt.

Het is wel nodig eerst `write18` te activeren[2] door in het bestand `texmf.cnf` de parameter `shell_escape` op `t` te zetten:

- MS-Windows-gebruikers moeten eerst het `texmf.cnf`-bestand zien te vinden – waarbij een probleem is dat in windows de extensie .cnf hardnekkig als een snelkoppeling wordt weergegeven
- bij het gebruik van de software van de TeX-live-cd zijn er twee exemplaren van `texmf.cnf`; ik heb ze allebei gewijzigd (één van beide heeft 'voorrang') en dat leverde het gewenste resultaat op: de log-file meldt dat `write18` enabled is
- MS-Windows-gebruikers die Miktex van de TeX-live-cd hebben geïnstalleerd (TeXnicCenter) moeten de Miktex-equivalent van `texmf.cnf` zien te vinden: `miktex.ini`.

Nu gaat het erom het juiste commando te maken dat via `write18` uitgevoerd kan worden: `flow.exe schema2.flo schema2pr.tex`.
De aanroep van flow gaat dan als volgt:
`\write18{flow.exe schema2.flo schema2pr.tex}`.
Met dezelfde LaTeX-run wordt het hiermee gemaakte 'plaatje' in de tekst opgenomen: `\input schema2pr.tex`. Op deze manier kan het stroomschemaatje gelijktijdig met de rest van het document gewijzigd worden en worden de laatst opgeslagen wijzigingen meteen in het eindresultaat verwerkt.

Op `www.ctan.org/tex-archive/support/flow` zijn source en documentatie van `flow` te vinden.

Voetnoten

1. een standaard-C programma voor het maken van stroomdiagrammen in de LaTeX picture-omgeving.
2. de shell-command optie staat standaard uit uit veiligheidsoverwegingen (in miktex.ini: write18=disable).

Ernst van der Storm
evdstorm@mso.com

Verslag EuroTEX 2005

Keywords
eurotex 2005, conferentie, verslag, tug

Abstract
Een verslag van de 16de jaarlijkse EuroTEX conferentie, gehouden in Pont-à-Mousson van 7 tot 11 maart 2005.

De aanloop

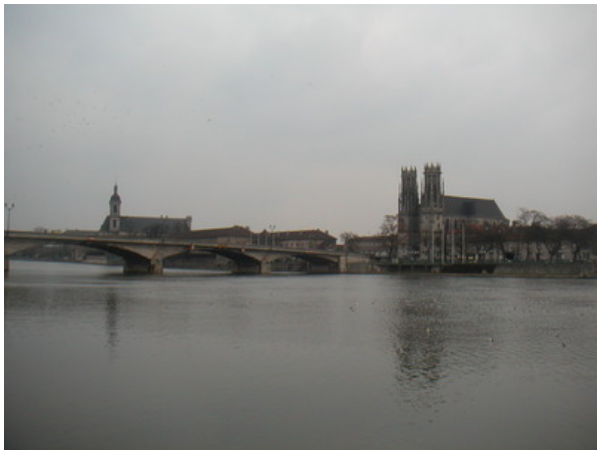
De jaarlijkse conferentie van de Europese TEX–gebruikersgroepen werd dit jaar al heel vroeg in het jaar gehouden, namelijk van 7 tot 11 maart. De locatie dit jaar was het voormalig klooster van de orde der Prémontrés in Pont-à-Mousson, een klein dorpje aan de Moezel in Noord-Frankrijk, vlakbij de grens met Duitsland.

Dat de conferentie werd gehouden op een punt vlakbij de grens tussen Frankrijk en Duitsland was toepasselijk, omdat tijdens deze euroTEX tevens de zestiende verjaardag gevierd werd van de verenigingen Dante en Gutenberg. 16 is een speciaal getal voor programmeurs, omdat het een macht van twee is: 2^{2^2}. En voor conferenties over TEX betekent dat dan weer, dat het mogelijk is om Donald Knuth over te halen om de conferentie te bezoeken.

Ik wil de NTG graag hartelijk bedanken voor de verstrekte beurs, en Hans Hagen voor het mij laten meerijden. Mijn deelname aan deze conferentie werd daardoor mogelijk gemaakt, ik had dit artikel anders niet kunnen schrijven.

Zondagochtend om negen uur stonden Michael Guravage en ikzelf klaar aan de achterzijde van het station Utrecht-CS om door Hans opgepikt te worden. Ondanks het hele koude weer verliep de reis voorspoedig, en we kwamen 's middags rond theetijd al in Pont-à-Mousson aan. Het programma zou pas de volgende ochtend beginnen, dus we hadden de rest van de zondag de tijd om (hernieuwd) kennis te maken met de andere deelnemers.

Rond koffietijd arriveerde de gezamenlijk georganiseerde en door de NTG gesponsorde bus uit Oost-Europa. Er kan rustig gesteld worden dat deze bus een gigantisch succes was.



Maandag

De eerste lezing na de officiële opening van de conferentie ging over Mem (voorheen Lambda, voorheen polyglot), een experimenteel pakket voor LaTEX onder Aleph, de opvolger van Omega (een artikel over Aleph is verschenen in de vorige MAPS[1]). De auteur, Javier Bezos, legde uit wat zijn belangrijkste doelen zijn (het zetten van meertalige documenten), wat hij verstaat onder een 'multi–lingual document', en hoe Mem kan helpen met het zetten daarvan.

Aan de hand van een Frans-Grieks document (met het Griekse gedeelte in drie verschillende encodings) legde hij uit wat de problemen zoal zijn en hoe die opgelost kunnen worden. Eén van de handigheidjes die onderdeel zijn van Mem, is een ietwat uitgebreide vorm van OTP-files. Dit aangepaste formaat staat het gebruik van Unicode karakter-namen toe in plaats van alleen maar numerieke waardes. Hierdoor zijn zogenaamde 'MTP'-bestanden een stuk begrijpelijker dan de gewone OTP-bestanden.

Mem is een experimentele omgeving, en er zijn nog wat bekende bugs, maar wie geïnteresseerd is in het onderwerp kan ermee aan de slag. Mem is beschikbaar op het internet:

http://mem-latex.sourceforge.net

1. Giuseppe Bilotta, *The Aleph-project*, (English), MAPS 31, 2005, pages 9–11.

De tweede lezing van de ochtend werd gegeven door Yannis Haralambous en Gábor Bella. De lezing ging over Omega natuurlijk, maar Yannis maakte eerst van de gelegenheid gebruik om wat reclame te maken voor zijn zojuist gepubliceerde boek[2].
De titel van deze lezing was “Omega becomes a sign processor”, en zal niemand verbazen dat het grootste gedeelte van de tijd nodig was om uit te leggen wat een ‘sign’ is, en waarom signs nodig zijn bij het verwerken van documenten. Het draait allemaal om het verschil tussen een ‘letter’ en een ‘representatie’. De huidige TEX-en zijn niet in staat dat onderscheid op een eenduidige manier te maken, en daardoor ontstaan er allerlei problemen bij het zetten van niet-westerse talen.

In de toekomstige versie van Omega (versie 2) zal dit probleem worden opgelost door het introduceren van ‘signs’ (‘tekens’, dus) die zowel de gegevens voor een ‘letter’ als die voor de ‘representatie’ bevatten. Een ‘teken’ bevat dan bijvoorbeeld het gegeven: `letter = U+0061 LATIN LETTER A`, maar tegelijkertijd ook: `representatie = glyph 97`. Dit is een heel simpel geval, maar in een Arabisch voorbeeld werd het ‘teken’ al snel uitgebreid zodat het dit bevatte: `letter = U+062C, representatie = glyph 18, form = 1, color = red`, en voor ligaturen en kerning-informatie werd het nog wat complexer.

Zoals altijd was Yannis’ lezing zeer professioneel, onderhoudend en goed te volgen. Een hele prestatie gezien de complexiteit van het onderwerp. Het wachten is wel op het verschijnen van Omega 2, want voorlopig is het allemaal nog theorie.

Het eerste praatje (na de koffiepauze) was de presentatie van een groepje mensen die werken aan een taxonomie voor automatische zetsystemen. Dit groepje (Chris Rowley, Joachim Schrod en Christine Detig) probeert een vocabulaire voor, en een formele classificatie van, automatische zetsystemen te maken. Het doel hierbij is om het mogelijk te maken vergelijkingen te trekken tussen verschillende aanpakken, en hopelijk op die manier het wetenschappelijk onderzoek in dit veld te stimuleren.

De lezing presenteerde geen conclusies, maar wel een paar dia’s om aan te geven in welke richting het project zich ontwikkelt. Typisch is dat de poging om tot een classificatie te komen zich veeleer lijkt te ontwikkelen tot een ‘landkaart’ dan tot de ‘boom’-structuur die je verwacht bij het horen van het woord taxonomie. Het onderzoek is nog lang niet af, dus op een latere conferentie zal vast nog meer volgen.

De laatste persoon die aan het woord was vóór de lunch was David Kastrup, met een uiteenzetting over de eisen die gesteld kunnen worden aan de programmeertaal waarin je een zetsysteem zoals TEX zou willen implementeren.

Natuurlijk wordt zo’n eisenpakket gekleurd door de bril van degene die de wensen opsomt, en wellicht daarom ging deze lezing voor een groot gedeelte over de problemen die er in de huidige TEX-en zijn bij het verwerken van kritische edities (zoals meervoudige voetnootsystemen en regelnummers). De gegeven voorbeelden maakten duidelijk wat er in de huidige TEX niet of slechts onvolledig gedaan kan worden. Daarbij werden mogelijke verbeteringen voor het algoritme opgenoemd, maar op een heel theoretisch niveau.

Pas aan het eind van zijn verhaal kwam hij eraan toe om aan te geven wat je zou willen van een programmeertaal om die algoritmen feitelijk in te programmeren, en helaas bleef dat bij nogal generieke eisen zoals ‘het moet mogelijk zijn de benodigde datastructuren eenvoudig uit te drukken’ en ‘overbodige zaken moeten vermeden worden’. Het maakte op mij nog geen coherente indruk, maar ook hierover zullen we later wellicht nog meer horen.

Jim Hefferon hield een praatje over de plannen die bestaan met betrekking tot CTAN. Dit sloot aan bij het ‘CTAN plans’ artikel in de TUGboat[3].
Een aantal van de plannen zijn al uitgevoerd: het integreren van Graham Williams’ TEX Catalogue; het verwijderen van de ‘supported’ directory in de LaTEX-boom; en het verplicht stellen van documentatie in PDF-vorm.

2. Yannis Haralambous, *Fontes et Codages*, (French), O’Reilly, first edition, 2004, 1012 pages. Hij verzekerde ons ervan dat er gewerkt wordt aan een Engelse vertaling.
3. Een herdruk hiervan vindt u elders in deze MAPS

Aan andere zaken wordt gewerkt, bijvoorbeeld het catalogiseren van pakketten via een trefwoordenregister (in samenwerking met een uitgebreider beheerssysteem, Jim liet hier een voorlopige demonstratie van zien); het beter integreren van de diverse mirrorservers met de drie hoofd–sites; en er wordt gewerkt aan het aanbieden van gecomprimeerde directories op alle mirrors.

Iets heel anders was het volgende onderwerp: Denis Roegel sprak over zijn MP2GL, een systeem dat MetaPost als invoer gebruikt voor het genereren van OpenGL C-code-fragmenten. Die code-fragmenten kunnen dan eenvoudig worden gecompileerd tot een OpenGL-programma.

Denis liet een aantal voorbeelden zien van MetaPost invoer, de door MetaPost geproduceerde PostScript uitvoer, en de uitvoer van OpenGL applicatie die werd gecompileerd vanuit de tekstuele (C programmacode) uitvoer van MetaPost.

Als ik dit allemaal goed heb begrepen is er sprake van een aantal specifieke MetaPost macros met verder niet veel systeemeisen. Maar ik moet een slag om de arm houden wegens het ontbreken van een artikel in de pre-prints van de conferentie.

Lezing zeven gaf ik zelf, met een korte presentatie van het MetaPost development team. De inhoud was een bewerking van mijn artikeltje in de vorige MAPS[4], met toevoeging van een aantal dia's waarop specifieke bugs en feature requests werden vermeld.
Aan de hand van deze lezing is er woensdagavond een BoF sessie[5] geweest over MetaPost, waarvan een verslag elders in deze MAPS te lezen is.

Na de koffiepauze zouden nog drie lezingen volgen. De eerste daarvan was van de hand van Péter Szabó, over zijn `examplep` pakket voor LaTEX. De titel was "Verbatim Phrases and Listings in LaTEX".

De lezing was niet heel erg bijzonder, maar het pakket zelf is met recht wonderbaarlijk te noemen. De lijst met unieke features die is weergegeven hieronder komt uit Péter's artikel in de syllabus. Die lijst beslaat echter alleen maar de 'extra bijzondere' features. Het pakket kan veel meer.

- □ layout of side–by–side display may depend on maximum source width.
- □ automatic hyphenation of inline verbatim.
- □ customizable isolation of page, section (and more) numbers between the Sample and its Host document.
- □ inline verbatim works safely inside macro arguments.
- □ abilitity to generate (example) files for inclusion on CD-ROM.

Stephan Lehmke en Andre Dierker van Quinscape namen de volgende voor hun rekening. Hun onderwerp was: "From RTF to XML to LaTEX".

Zij lieten zien hoe je van RTF via XML naar TEX kunt gaan, en dat er dan toch uitvoer mogelijk blijft die redelijk goed aansluit op het originele Word-bestand (en dat is iets wat in commerciële omgevingen vaak een eis is). Daarbij gebruiken ze de Open source tool Majix om de RTF te converteren naar XML, om dat vervolgens te verwerken met XMLTEX.

De huidige versie is nog niet geschikt voor algemeen gebruik omdat font- en kleurveranderingen nog niet geïmplementeerd zijn. Het plan bestaat om de XMLTEX-macros te uploaden naar CTAN, zodat er weer een extra manier beschikbaar komt om van Word naar TEX te komen.

De laatste spreker van de maandag was Jonathan Fine, met als titel "TEX forever!". Jonathan sprak over een aantal verschillende projecten die een interface aanleggen tussen TEX en de Python programmeertaal, maar in het bijzonder over zijn nieuwe project: QATEX.

QATEX is een Python script dat TEX uitvoert met omgeleide invoer en uitvoer kanalen. Het concept draait erom dat TEX-specifieke 'prompts' kan doen, die dan onderschept worden door de Python wrapper. Dat script voert vervolgens de gewenste Python code uit en stuurt het resultaat terug naar TEX als het antwoord op de prompt. De gebruiker ziet hier niets van, behalve dan dat de TEX programmeertaal plotseling is uitgebreid met een 'python interpreter'.

4. Taco Hoekwater, *MetaPost Developments*, (English), MAPS 31, 2005, page 8.
5. BoF staat voor 'Birds of a Feather'. Een BoF sessie is een informele bijeenkomst van diegenen die geïnteresseerd zijn in een bepaald onderwerp

Dinsdag

De eerste lezing op dinsdagochtend werd gegeven door Sebastian Rahtz. “The TEI/TEX Interface” ging over het gebruik van TEI XML bestanden in samenwerking met TEX.

Na een korte introductie van TEI XML documenten liet hij ons zien wat de verschillen zijn tussen publiceren met XML en met TEX. Aan de hand van de verschillen blijkt dat er vier verschillende trajecten open zijn als je TEX wilt gebruiken om de XML te zetten: De XML rechtstreeks verwerken, de XML converteren naar TEX, de XML vertalen naar een vorm van XML die functioneel op TEX lijkt en dat daarna alsnog converteren, of de XML vertalen naar XSL Formatting Objects.

Geen van deze opties is 100% geschikt in alle gevallen, maar het is altijd wel mogelijk een oplossing te vinden.

Natuurlijk waren de mensen van het LaTEX3-team ook op de conferentie, en Frank Mittelbach gaf ons de stand van zaken. Zijn nieuwtjes zijn ook allemaal na te gaan op de herziene website van het project:

http://wwww.latex-project.org

Het belangrijkste nieuws van het LaTEX-project is dat de huidige revisie van de LPPL (LaTEX-Project Public License) nu officieel de status ‘free’ heeft gekregen van het Debian-project. Dat betekent dat de LPPL nu een geldig alternatief is voor de GPL.

Frank legde uit dat er nog best veel werk komt kijken bij het ondersteunen van LaTEX2e, maar gelukkig is er toch nog tijd over om te werken aan LaTEX3. Tijdens het gehele vervolg van de conferentie kon je geen deur opendoen of er was wel een LaTEX-vergadering gaande, dus het is mij absoluut duidelijk dat ze er druk mee bezig zijn.

Na de koffie was het tijd voor Hans Hagen. Hij praatte over de layout van het Nieuw Archief voor de Wiskunde, waarvan het zetwerk al enige jaren gedaan wordt met een set ConTEXt macros die hij momenteel aan het opschonen is.

De vormgeving van het NAW is niet wat je zou verwachten van een wetenschappelijk tijdschrift met zetwerk in TEX, het ziet er meer uit als een glossy magazine. Onder andere komt dit doordat er een aantal verschillende, vrij complexe, bladspiegels worden gebruikt, en omdat het gebruikte font absoluut niet lijkt op het Computer Modern dat we gewend zijn. Als redactie proberen we Hans te porren om een artikel te maken voor MAPS 33.

Adam Twartoch sprak over ‘Typographic Perfection with OpenType?’. Adam is helemaal geen TEX-gebruiker, maar is werkzaam als fontdesigner.

Hij nam ons eerst mee op een excursie door het verleden van digitale fonts, van Ikarus in 1975, via Type1 en TrueType, tot het ontstaan van OpenType in 2000. Daarna vertelde hij ons over een aantal van de meer geavanceerde mogelijkheden van het OpenType fontformat, onder meer door te laten zien hoe je in *notepad* woorden kunt intypen met het OpenType font Zapfino, waarbij het font ‘zelf’ bij elke toetsaanslag de ligaturen vervangt in de geproduceerde tekst.

Na de lunch ging het verder met Gerd Neugebauer, die sprak over het implementeren van *namespaces* voor TEX. Het eerste gedeelte van zijn presentatie ging over de eisen waaraan namespaces moeten voldoen, zoals de mogelijkheid tot het importeren van namen uit andere namespaces en de conflicten die daarbij ontstaan.

Het tweede deel van zijn verhaal ging over de feitelijke implementatie van namespaces binnen exTEX, en een korte introductie van exTEX zelf. Dit project is een Java her-implementatie van TEX die losjes gebaseerd is op NTS, maar dan met wat meer focus op snelheid. In de huidige toestand is exTEX nog niet echt bruikbaar, maar er wordt aan gewerkt. De website is:

http://www.extex.org

Patrick Gundlach introduceerde de ConTEXt services die beschikbaar zijn op `contextgarden.net`: Wiki, texshow-web, de archieven van de email-lijsten, de ‘live’ ConTEXt-distributie en de source code browser.

Dit gebeurt aan de hand van hetzelfde artikel dat wij al in de vorige MAPS hebben afgedrukt[6], dus u kunt het daarin rustig nalezen.

6. Patrick Gundlach, *contextgarden.net*, (English), MAPS 31, 2005, pages 87-90.

De maker van pdfTEX, Hàn Thế Thành, introduceerde de micro-typografische extensies in pdfTEX, en de verbeteringen die daarin zijn aangebracht in de afgelopen tijd. Hijzelf is geruime tijd 'uit de running' geweest, maar sinds versie 1.20a is Hàn Thế Thành zelf weer betrokken bij de releases (de huidige versie is 1.21a).

Voor gebruikers zijn de belangrijkste nieuwtjes van de afgelopen tijd ongetwijfeld het verdwijnen van het `pdftex.cfg`-bestand, alle configuratie gebeurt vanaf nu in macros; en het beschikbaar komen van automatische font-expansie, zodat het niet meer nodig is om allerlei TFM-bestanden beschikbaar te hebben voor verschillende expansie–factors.

Johannes Küster is van plan weer een aanvang te maken met een oud TUG-project: de Newmath encoding. Dit project is tijdelijk stopgezet tijdens de euroTEX bijeenkomst in St. Malo (1998), omdat toen eerst aandacht moest worden besteed aan het toevoegen van de benodigde tekens aan Unicode. Nu dit geslaagd is, is het tijd om de draad weer op te pakken en wederom aan het werk te gaan met de encodings.

De lezing was vooral een introductie van het werk dat een aantal jaren geleden al gedaan is, en een planning van wat er nog moet gaan gebeuren voor er een werkbaar resultaat is. Johannes wil gaan werken aan een set 'Latin Modern' compatibele wiskunde fonts, maar natuurlijk moeten de feitelijke encodings ook nog afgemaakt worden. Later in de week is er een BoF sessie geweest over dit onderwerp. TUG's 'Math font group' heeft nog steeds een website waarop alle informatie beschikbaar is:

http://www.tug.org/twg/mfg/

De 'Latin Modern' fonts die ik al snel even noemde in de vorige paragraaf, waren het onderwerp van de volgende lezing, door Bogusław Jackowski.

Het Latin Modern project is begonnen op de EuroBachoTEX conferentie in 2002, en het doel is om te komen tot een set van outline fonts die compatibel zijn met Computer Modern, maar dan met zoveel mogelijk van de gelokaliseerde CM varianten erin geïntegreerd. De gehele implementatie wordt gedaan met behulp van het MetaType1 systeem, dat vooral bestaat uit een set macros voor MetaPost.

De font–set nadert zijn definitieve vorm (de huidige release is 0.982), en deze lezing liet een paar nog overbleven details zien. Het gaat dan bijvoorbeeld om het al dan niet dubbel opnemen van bepaalde geaccentueerde karakters, en de spatiëring rondom en in aanhalingstekens. De meest actuele Latin Modern fonts kunnen worden gedownload van CTAN, en wat oudere versie zijn beschikbaar in de meeste distributies.

Het hoogtepunt van deze dinsdag en wellicht van de hele conferentie was de panel–discussie met Hermann Zapf en Donald Knuth. De discussie werd geleid door Hans Hagen en Volker Schaa, maar verreweg het meeste aan het woord waren Zapf en Knuth zelf.

Het eerste onderwerp was Hermann Zapf's Optima-Nova. Vijftig jaar na de eerste versie van het Optima font, is Zapf door linotype in staat gesteld om een nieuwe versie te maken zonder de beperkingen die werden opgelegd door de productieprocessen van 50 jaar geleden. De nieuwe versie is verschenen als een OpenType font, en bevat naast de correcties ook nog een groot aantal extra tekens.

Om de verschillen tussen de oude en de nieuwe goed te kunnen laten zien, had Hans een presentatie gemaakt met op elke pagina een teken uit zowel de oude als de nieuwe versie van het font. Beide tekens overlapten elkaar en waren gedeeltelijk transparant weergeven, zodat de verschillen tussen beide versies in één oogopslag duidelijk waren. Aan de hand van deze slides heeft Zapf een aardige tijd besteed aan het in detail uitleggen waarom 'de oude versie slechter is dan de nieuwe'.

Hermann Zapf is momenteel bezig met een soortgelijke opschoning van zijn Palatino fonts.

Het andere grote onderwerp was Latin Modern. Aan de hand van een soortgelijke serie elkaar overlappende tekens op dia's werd Donald Knuth (en Zapf) uitgenodigd om commentaar te leveren op de nieuwe tekens die gemaakt zijn door Bogusław en zijn team. Knuth legde uit dat een aantal van de vreemde eigenschappen van Computer Modern een gevolg zijn van de zo nauwkeurig mogelijke navolging van het originele Monotype font.

Knuth vertelde een aardig verhaal over hoe de eerste versie van Computer Modern gemaakt werd door geprojecteerde 35mm dia's over te trekken en te digitaliseren. Pas veel later had hij de beschikking over de echte loodletters van het font. Van alle tekens in Computer Modern is hij momenteel nog het meest ontevreden over het dollar-teken.

In de twee uur durende sessie is er nog veel meer besproken. Ik hoop dat iemand kans heeft gezien de hele sessie uitgebreid op papier te zetten, want het was allemaal razend interessant. Er is nog veel meer (vooral over fonts) besproken, maar ik heb hier niet genoeg ruimte om alles uitgebreid uit de doeken te doen.

Na deze sessie volgde het gala–diner. Wat me daaraan vooral is bijgebleven is dat iedereen (maar vooral Knuth en Zapf) kadootjes kreeg van iedereen. Hermann heeft eigenhandig de namen van Dante en Gutenberg gecalligrafeerd op de twee verjaardagstaarten. Met een suikerspuit, uiteraard!

Woensdag

Woensdag begon al vroeg: Thomas Feuerstack en Klaus Höppner begonnen al om half negen met een introductie van proTEXt. ProTEXt is een nieuwe TEX-distributie voor Windows, waarvan alle leden van de NTG de CD reeds enige tijd in huis hebben. De achterliggende filosofie van deze distributie is dat minder keuze ook zorgt voor minder verwarring, en dat op die manier een laagdrempeliger distributie mogelijk wordt.

De componenten van de distributie zijn MikTEX, WinEDT / TeXnicCenter, Ghostscript en andere post-processors, en heel wat documentatie. In een Live demonstratie lieten ze zien hoe de installatie verloopt door een PDF-document te lezen en vandaaruit op hyperlinks te klikken die dan de installatieprogramma's opstarten. De gebruikersvriendelijkheid van dat installatieproces is voorbeeldig.

Jean-Michel Hufflen sprak over MlBibTEX, en dan vooral over het schrijven van bibliografische stijlen daarvoor. MlBibTEX maakt gebruik van een nieuwe taal (`nbst`) voor het implementeren van zulke stijlen, een taal die heel erg veel lijkt op XSLT, maar dan uitgebreid met zaken die nodig zijn voor het meertalige aspect van MlBibTEX.

Persoonlijk vind ik deze nieuwe taal een hele vooruitgang boven de traditionele `bst` stack–gebaseerde taal die is bedacht door Oren Patashnik voor de originele BibTEX. De nieuwe aanpak leunt op XML, dus het is noodzakelijk om wat achtergrondkennis in dat gebied te hebben of op te doen, maar daar staat tegenover dat het systeem veel minder 'eigenwijs' is. Er zijn voor de toekomst plannen om ook de bibliografische databases te gaan opslaan in een XML-formaat in plaats van in BIB-formaat.

De volgende lezing ging over "La machine à formulaires", een set TEX macros die bedoeld zijn om sollicitanten te helpen om te gaan met de merkwaardige aanneem–procedures van de universiteiten in Frankrijk. De alternatieve titel luidde "TEX for a Kafkaian world".

Als ik dit goed heb begrepen, dan is het noodzakelijk dat universitaire sollicitanten binnen enkele weken op een hele serie vacatures reageren, en dat al die sollicitatie–formulieren weliswaar sprekend op elkaar lijken, maar dat de bureaucratie wel eist dat ze allemaal apart gemaakt en verzonden worden.

Het doel van het macro–pakket dat Antoine Lejay presenteerde is dat al die sollicitaties zoveel mogelijk geautomatiseerd gemaakt kunnen worden. Onder andere door het samenvoegen van alle dubbele informatie in enkele `\include` bestanden.

Frank-Rene Schaefer praatte over SaferTEX. SaferTEX gebruikt een invoertaal die wat gelijkenissen vertoont met Wiki invoer: zo weinig mogelijk opmaak, en de opmaak die er is is meer zoals je zou werken als je achter een ouderwetse typemachine zou zitten.

Een uitgebreide compiler genereert vervolgens TEX invoer op basis daarvan. Er is zou een website moeten zijn tegen de tijd dat u dit leest:

http://safertex.sourceforge.net

Het eerste praatje na de koffiepauze ging over iWrapper. Jérôme Laurens sprak over het uitwisselen van TEX-bestanden tussen auteurs, en over de meta–informatie nodig is om een "self contained TEX document" te kunnen verspreiden. Het blijkt bijvoorbeeld noodzakelijk te zijn om ergens een lijst bij te houden van bestanden die meegezonden moeten worden.

De aan de hand van de eisen gedefinieerde 'TEX-Wrapper'-structuur wordt gebruikt door iTEXMac. De metadata van een project wordt daarbij opgeslagen in XML property lists in een aparte directory met als naam `<document>.texp`. Vooralsnog is dit concept nauw verbonden aan Mac OS X, maar het is eventueel mogelijk om hiervan gebruik te maken op andere platforms. De iTEXMac homepage:

http://itexmac.sourceforge.net

Stephan Lehmke kwam nogmaals op de spreekstoel voor een tweede verhaal, ditmaal over de productie van de catalogus van een verlichtingsbedrijf. Hij had een voorbeeld van die catalogus meegenomen, een enorm dikke pil van 650 pagina's, die gepubliceerd wordt in 14 verschillende talen.

Quinscape GmbH (waar Stephan werkt) heeft hiervoor een product ontwikkeld met de naam DocScape. Dit is een op pdflatex gebaseerde oplossing voor het publiceren van gegevens uit (product)databases. Hij liet een aantal voorbeelden zien uit de feitelijke productie van deze catalogus, en het resultaat was bepaald indrukwekkend. U kunt er zelf ook naar kijken, vanuit de links op deze pagina:

http://www.erco.com/download/

De laatste lezing van de conferentie was van de hand van David Kastrup, over zijn pakket voor het zetten van kritische edities. Dit het pakket heet 'bigfoot'.

Zoals te verwachten valt van een pakket voor kritische edities ging deze lezing vooral over het uitgebreide subsysteem voor het maken van voetnoten. Zo worden verschillende voetnoot-stromen tegelijkertijd ondersteund, kunnen voetnoten genest worden binnen elkaar, en kunnen ze afgebroken worden over paginagrenzen heen. Bigfoot is beschikbaar op CTAN.

Na de lunch vond de excursie plaats naar Nancy, maar nog voor de lunch werd de conferentie officieel afgesloten. Tevens werden er vast vooraankondigingen gedaan voor de komende conferenties.

Tegen de tijd dat u dit leest is het nog net niet te laat om af te reizen naar China voor de TUG 2005 conferentie die in Wuhan gehouden wordt van 23 to 25 augustus. Het is dan al wel te laat voor de BachoTEX van dit jaar die natuurlijk weer voor begin mei op het programma staat.

De EuroTEX van volgend jaar zal plaatsvinden in Boedapest. Deze aankondiging ging gepaard met een serie dia's die de indruk wekken dat ook die EuroTEX weer een bijzonder evenement zal worden. Ik hoop dat er volgend jaar juni in Hongarije minstens net zo veel Nederlanders aanwezig zullen zijn als er dit jaar in Frankrijk waren.

Een van de twee bussen voor de excursie naar Nancy was de bus uit Brodnica die de Polen en Duitsers had afgeleverd. Wellicht was het te verwachten, maar halverwege de reis naar Nancy moest er nog even een tussenstop gemaakt worden om de bouten van een wiel opnieuw aan te draaien. Gelukkig was er verder niets aan de hand, en de rest van de excursie verliep voorspoedig.

Bij het bezoek aan het 'fine arts' museum in Nancy viel op dat er een relatief groot aantal Vlaamse meesters in de collectie zaten. Verder was er natuurlijk een heleboel glaswerk te zien uit de Baum-collectie (Nancy is een centrum van de Noord-Franse glasindustrie), maar verder was het museum wat karig.

Vandaar dat we nog een flinke rondwandeling door het stadje hebben gedaan, gevolgd door warme chocolade in een café aan het grote centrale plein. Rond zeven uur waren we weer terug in Pont-à-Mousson.

De uitloop

Voor donderdag en vrijdagochtend stonden er tutorials op het programma. Zoals gewoonlijk voor tutorials liepen er twee tegelijkertijd, zodat ik niet overal aanwezig kon zijn.

Mijn ochtend werd gevuld met een tutorial over TEX en XML, waarbij ik de eer had om Sebastian Rahtz te mogen vervangen als hulpje van Hans Hagen. Wegens gebrek aan voorbereiding (net voor de conferentie bleek dat Sebastian plotseling een dag eerder weg moest waardoor hij niet zelf de tutorial kon doen) was mijn eigen inbreng gering, maar door de ConTEXt-voorbeelden van Hans was het wel een succesvolle tutorial.

Tegelijkertijd met de XML-in-TEX-tutorial liepen er twee andere tutorials na elkaar: één over TEXPower (door Stephan Lehmke) en een over de internals van exTEX (door Gerd Neugebauer).

De middag heb ik besteed aan het bijwonen van een vergadering van het pdfTEX-team, zodat ik van de andere drie tutorials van de dag ook niet veel heb meegekregen: Een lange tutorial van Denis Roegel over MetaPost, en twee korte: één over een installer van TEX Live 2004 voor Windows (door Staszek Wawrykiewicz), en één over Emacs&AucTEX (door David Kastrup).

Vrijdagochtend was er slechts één tutorial: een beginnerscursus ConTEXt (door Hans Hagen natuurlijk). Er was ook een 'Advanced LaTEX' tutorial voorzien, maar die vond geen doorgang bij gebrek aan een beschikbare leraar (het LaTEX-team was er nog wel, maar die waren druk aan het vergaderen en het leek dat er toch geen heel grote opkomst geweest zou zijn).

De ConTEXt-tutorial werd wel drukbezocht, en op verzoek van de aanwezigen ging die vooral over het doen van specifieke layouts zoals voor posters of wenskaarten. Daarbij is vooral het definiëren van overlays en het selecteren van specifieke fonts van belang, zodat een aantal voorbeelden daarvan de revue passeerden.

In de middag vertrok de bus met de Polen weer, en hoewel wij pas de volgende ochtend in de auto stapten om terug te gaan naar Nederland, was daarmee de conferentie feitelijk beëindigd. 's Avonds hebben we met Fabrice Popineau in een lokaal restaurantje nog gevieren aan de pizza gezeten.

Op het moment dat ik dit nawoord opschrijf is de conferentie al meer dan een maand geleden, en ik ben nog steeds bezig ben met het afwerken van diverse punten die zijn aangekaart in de wandelgangen van het klooster in Pont-à-mousson. Alleen al door dat feit is het mij duidelijk dat we kunnen terugzien op een zeer geslaagde bijeenkomst, en ik kijk vol vertrouwen uit naar EuroTEX in Boedapest volgend jaar!

Taco Hoekwater
taco@elvenkind.com