

MAPS

NUMMER 35 • VOORJAAR 2007

REDACTIE

Taco Hoekwater, hoofdredacteur
Wybo Dekker
Frans Goddijn
Siep Kroonenberg



NEDERLANDSTALIGE T_EX GEBRUIKERSGROEP



Voorzitter
Hans Hagen
ntg-president@ntg.nl

Secretaris
Willi Egger
ntg-secretary@ntg.nl

Penningmeester
Wybo Dekker
ntg-treasurer@ntg.nl

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

Taco Hoekwater
taco@elvenkind.com

Postadres
Nederlandstalige T_EX
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

Copyright © 2007 NTG

De **Nederlandstalige T_EX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft de kennis en het gebruik van T_EX te bevorderen. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde document-opmaak in het algemeen en de ontwikkeling van ‘T_EX 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 T_EX en daarmee verwante programmatuur.

De NTG biedt haar leden ondermeer:

- ☐ Tweemaal per jaar een NTG-bijeenkomst.
- ☐ Het NTG-tijdschrift MAPS.
- ☐ De ‘T_EX Live’-distributie op DVD/CDROM inclusief de complete CTAN software-archieven.
- ☐ Verschillende discussielijsten (mailing lists) over T_EX-gerelateerde onderwerpen, zowel voor beginners als gevorderden, algemeen en specialistisch.
- ☐ De FTP server ftp.ntg.nl waarop vele honderden megabytes aan algemeen te gebruiken ‘T_EX-producten’ staan.
- ☐ De WWW server www.ntg.nl waarop algemene informatie staat over de NTG, bijeenkomsten, publicaties en links naar andere T_EX sites.
- ☐ Korting op (buitenlandse) T_EX-conferenties en -cursussen en op het lidmaatschap van andere T_EX-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 L^AT_EX- of ConT_EXt formaat. Bijdragen op alle niveaus van expertise zijn welkom.

Productie. De Maps wordt gezet met behulp van een L^AT_EX class file en een ConT_EXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.40.3-2.2 (Web2C 7.5.6) 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.

T_EX is een door professor Donald E. Knuth ontwikkelde ‘opmaaktaal’ voor het letterzetten van documenten, een documentopmaakstelsel. Met T_EX is het mogelijk om kwalitatief hoogstaand drukwerk te vervaardigen. Het is eveneens zeer geschikt voor formules in wiskundige teksten.

Er is een aantal op T_EX gebaseerde producten, waarmee ook de logische structuur van een document beschreven kan worden, met behoud van de letterzetmogelijkheden van T_EX. Voorbeelden zijn L^AT_EX van Leslie Lamport, $\mathcal{A}_M\mathcal{S}$ -T_EX van Michael Spivak, en ConT_EXt van Hans Hagen.

Inhoudsopgave

Redactioneel, <i>Taco Hoekwater</i>	1
Announcement: TUG conference 2007, <i>Taco Hoekwater</i>	2
René van der Heijden, <i>NTG Bestuur</i>	3
The MPlib Project, <i>Hans Hagen & Taco Hoekwater</i>	4
Tokens in Lua _{TeX} , <i>Hans Hagen</i>	5
Integrating the pool file, <i>Taco Hoekwater</i>	9
PDF / TikZ, <i>Willi Egger</i>	11
External graphics for LaTeX, <i>Siep Kroonenberg</i>	18
Review: Alphabetgeschichten, <i>Hans Hagen & Taco Hoekwater</i>	27
Folding Sheets for a Modular Origami Dodecalendar, <i>Richard Hirsch</i>	30
Con _{TeX} t User Meeting 2007, <i>Mojca Miklavc</i>	37
EuroBach _{TeX} 2007, <i>Michael Guravage</i>	43
MiK _{TeX} installeren valt erg mee, <i>Frans Goddijn</i>	51

Redactioneel

Hopelijk valt deze Maps nog net voor de voorjaars-bijeenkomst bij jullie in de bus. Of, ik zou eigenlijk moeten zeggen: dwarrelt in de bus. Want het is een dunnetje geworden, deze keer. Nu verwachten jullie natuurlijk dat ik ga klagen over een gebrek aan kopij en vrije tijd, want dat doe ik immers meestal.

Maar deze keer is de Maps juist doelbewust zo dun. We hadden meer dan genoeg artikelen over kunnen nemen uit de EuroT_EX 2007 proceedings om een hele, hele dikke Maps te kunnen maken. Maar dat zou heel bewerkelijk geweest zijn, want die artikelen zouden we allemaal moeten herpagineren voor de Maps layout, en de relatief kleine oplage van de Maps maakt dat bovendien al snel een prijzige onderneming.

Daar komt bij dat de EuroT_EX proceedings hoe-dan-ook in een grote oplage gedrukt gaat worden, omdat alle Dante en Gust leden er één krijgen. Het leek ons daarom alles bij elkaar veel praktischer om die extra service uit te breiden naar de NTG leden. We weten nog niet precies wanneer ze de deur uit zullen gaan, maar ergens in de nazomer zullen jullie dus allemaal een extra enveloppe van de NTG ontvangen!

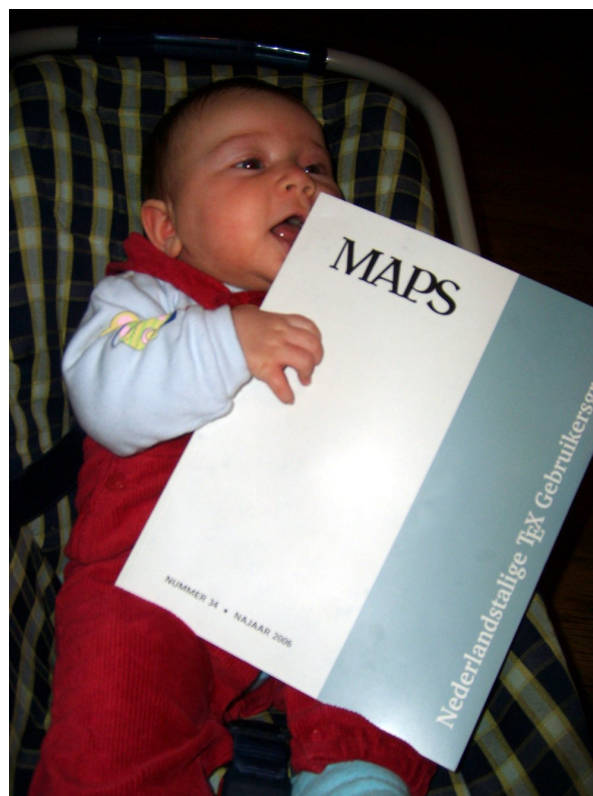
Aan het einde van het najaar volgt dan natuurlijk weer gewoon de volgende Maps. En dáárvoor hebben we wel weer dringend behoefte aan kopij en extra vrijwilligers. Heb je wat te melden, of wil je helpen met de productie van de Maps, stuur de redactie dan een mailtje. We zijn altijd blij met nieuwe artikelen, vooral die van eigen bodem.

Ondertussen is er natuurlijk ook nog deze Maps. We hebben vrij veel aandacht voor grafische onderwerpen deze keer, zoals het eerste deel van een tweeluik van *Richard Hirsch* over het maken van een driedimensionale kalender met behulp van MetaPost. Van *Willi Egger* hebben we een korte kennismaking met het relatief nieuwe tekenpakket PDF / TikZ, en *Siep Kroonenberg* gaat dieper in op het gebruik van externe figuren in L^AT_EX.

In het afgelopen voorjaar hebben er twee internationale conferenties plaatsgevonden waar we een verslag van hebben opgenomen: van de ConT_EXt user meeting in Epen een artikel van de hand van *Mojca Miklavc*, en van de EuroBachOT_EX in Polen een verslag door *Michael Guravage*. Ik wil Michael overigens nog van harte bedanken voor het proeflezen en corrigeren van al ons gebrekkige Engels.

Van *Hans Hagen* en mijzelf zijn er een paar korte artikeltjes met nieuws rondom MetaPost en LuaT_EX. En er is een nieuw boek van Hermann Zapf uit, daar hebben we een review van geschreven. *Frans Goddijn* heeft MikT_EX geïnstalleerd, wat weer niet zonder slag of stoot ging, maar als ik goed snap werkt nu alles naar behoren.

Tot slot nog even terug naar Maps 34. Weten jullie nog dat we toen een paar fotootjes hebben afgedrukt van het kersverse dochttertje van Paweł (pagina 77)? Natuurlijk ging er toen een exemplaar per post naar Polen, en een paar dagen later kregen we onderstaande foto in de email. Om op te eten, toch?



Veel leesplezier, en tot de volgende keer!

Taco Hoekwater, hoofdredacteur

TUG 2007: Practicing T_EX

Workshops and presentations on

**L^AT_EX, T_EX, MetaPost,
ConT_EXt, LuaT_EX,
and more**

July 17–20, 2007

**San Diego State University
San Diego, California, USA**

<http://tug.org/tug2007>
tug2007@tug.org

Keynote address: *Peter Wilson*,
The Herries Press

- April 23, 2007 — presentation proposal deadline
- May 18, 2007 — early bird registration deadline
- July 17–20, 2007 — workshop and conference



Further information

Conference attendees will enjoy an opening night reception and an (optional) banquet one evening. Coffee and lunch will be served each day of the meeting. Located on the campus of San Diego State University, an easy trolley ride from downtown San Diego. Inexpensive campus housing is available.

Conference fee, hotel, and other information is available on the web site.

Sponsorship

We thank the present sponsors: the German-speaking T_EX users group DANTE e.V., von Hoerner & Sulger GmbH, MacKichan Software, and Adobe Systems Inc. have provided generous support; San Diego State University is our host; and special thanks to the many individual contributors.

If you'd like to support the conference, promote T_EX products and services, or otherwise provide sponsorship, see the web site for donation and advertising options.

Hope to see you there!

Sponsored by the T_EX Users Group

René van der Heijden

2007[†]

“Wanneer ik deze wereld ga verlaten, weet ik nauwelijks wat mij het meest zal spijten: de mensen die ik nooit heb ontmoet, of de plaatsen die ik nooit heb gezien. Als het om de wereld van het boek gaat kan ik mijn hoofd tamelijk tevreden neerleggen: ik heb mijn portie gelezen.”

W.N.P. Barbellion, 10 maart 1917.



Een van de aardige dingen van een bijeenkomst van T_EX gebruikers is dat je “elkaar weer eens ziet”; er is een vaste kern die altijd komt. In de ruim tien jaar dat ik NTG bijeenkomsten bezoek was René van der Heijden voor mij zo’n ankerpunt geworden. Als hij er was, dan waren we compleet en konden we beginnen.

Op vrijdag 6 april 2007 is René van der Heijden overleden. Als bestuur wisten we al wat eerder dat was vastgesteld dat hij ongeneeslijk ziek was. Het kwam evengoed als een schok.

René: T_EX op de Atari, schaken en altijd geïnteresseerd in nieuwe ontwikkelingen rond T_EX waarvoor hij, zoals hij me vaak vertelde, helaas niet altijd tijd had. Nu een bekend gezicht is weggefallen, zal de volgende bijeenkomst toch wat anders zijn. Hij was er altijd, en wij zullen hem missen.

We wensen zijn vrouw Tia en haar omgeving veel sterkte.

Hans Hagen

The MPlib Project

MetaPost as a reusable component

As you probably know, MetaPost development has restarted approximately two years ago. After a period of investigating user demands, it has now become obvious that MetaPost is showing its age.

The problems lie not so much in the actual drawing language that is used, but in the 1980s Metafont legacy that is very noticable in the way the program interacts with the user and in how it deals with the computing environment in general.

Some of the big user-side problems that resurface on a regular basis are:

- The model used for the handling of external labels is outdated.
Running a per-file preprocessor to create the labels was already problematic before, but it is getting worse now that both T_EX and Troff are moving away from their traditional output formats.
- All number handling is based on fractions of a 32-bit integer.
User input often hits one of the many boundaries that are a result of that. For instance, no numbers can be any larger than 16384, and there is a noticeable lack of precision in the intersection-point calculations.
- MetaPost cannot be used as a system-level service.
In fact, MetaPost cannot even be used as a system-wide library, because the many global variables make it non-shareable.
- Lack of 3-D support.
Even technical drawings that are nominally considered to be two-dimensional, like the ones in highschool math and physics books, often need to handle projections of 3-D objects to a plane.

Much of the needed development to fix these issues can be done in the normal course of events, because the needed extensions or changes to the program are isolated to a small section of the source code (this is for instance true for 3-D projection support), or because the needed changes are so well understood that it is trivial to make many changes (this is true for upgrading the 32-bit internal calculus).

But the handling of labels and the lack of system integration require massive changes to the source code

as well as to the build system, and therefore it was very unlikely that this would ever get done without extra incentives: a significant amount of time and effort that has to be dedicated to those particular problems.

An estimate of the needed programming hours to turn MetaPost into a modern, re-entrant system library with a modern form of inter-process communication was created:

Converting from the use of hunderds of global variables into a data structure that is passed on from one function to another	200
Adding a unified redirection layer for the input and output, allowing files as well as buffers to be used	100
Designing and implementing a new subsystem for label typesetting	150
Adding an interface for configurable default error responses	50
Total	500

If writing documentation is included in that estimate, it makes for six months of full-time (40 hours a week) programming, time that simply could be allotted within anybody's free hours in any way. It was clear to us that, to get these tasks done within a reasonable time frame, at least some of the work would have to be done during office hours. And that requires money.

So, a funding proposal was written and at the Dante 2007 meeting Hans proposed this new project for funding. Dante immediately stepped in for 50% of the requested amount (6000 euro) and within a week other user groups joined in as well: TUGIndia (1000 euro), TUG (1500 euro), NTG (2000 euro) and CSTug (1000 euro). Currently 500 euro is still missing, but we are confident that this gap will be bridged or overcome.

Work will start in the autumn of this year, and it is our current estimate that the project will be complete by the summer of 2008. The actual programming will be carried out by Taco. Hans Hagen will lead the project, and Bogusław Jackowski will be in charge of quality control.

Tokens in LuaTeX

Hans Hagen

tokenization

Most TeX users only deal with (keyed in) characters and (produced) output. Some will play with boxes, skips and kerns or maybe even leaders (repeated sequences of the former). Others will be grateful that macro package writers take care of such things.

Macro writers on the other hand deal with properties of characters, like catcodes and a truckload of other codes, with lists made out of boxes, skips, kerns and penalties. But even they cannot look much deeper into TeX's internals. Their deeper understanding comes from reading the TeXbook or even looking at the source code.

When someone enters the magic world of TeX and starts asking around a bit, he or she will at some point get confronted with the concept of tokens. A token is what ends up in TeX after characters have entered its machinery. Sometimes it even seems that one is only considered a qualified macro writer if one can talk the right token-speak. So, what are those magic tokens and how can LuaTeX shed light on this?

In a moment we will show examples of how LuaTeX turns characters into tokens, but when looking at those sequences, you need to keep a few things in mind:

- A sequence of characters that starts with an escape symbol (normally this is the backslash) is looked up in the hash table (which relates those names to meanings) and replaced with its reference. Such a reference is much faster than looking up the sequence each time.
- Characters can have special meanings, for instance a dollar is often used to enter and exit math mode, and a percent symbol starts a comment and hides everything following it on the same line. These meanings are determined by the character's catcode.
- All the characters that will end up actually typeset have catcode letter or other assigned. A sequence of items with catcode letter is considered a word and can potentially become hyphenated.

examples

We will now provide a few examples of how TeX sees your input.

Hi there!

Hi there!

cmd	chr	id	name
letter	72	H	
letter	105	i	
spacer	32		
letter	116	t	
letter	104	h	
letter	101	e	
letter	114	r	
letter	101	e	
other_char	33	!	

Here we see three kinds of tokens. At this stage a space is still recognizable as such, but later this will become a skip. In our current setup, the exclamation mark is not a letter.

Hans \& Taco use Lua\TeX \char 33\relax

Hans & Taco use LuaTeX!

cmd	chr	id	name
letter	72	H	
letter	97	a	
letter	110	n	
letter	115	s	
spacer	32		
char_given	38	1114152	&
spacer	32		
letter	84	T	
letter	97	a	
letter	99	c	
letter	111	o	
spacer	32		
letter	117	u	
letter	115	s	
letter	101	e	
spacer	32		
letter	76	L	
letter	117	u	
letter	97	a	
call	1554614	1114740	TeX
char_num	0	1115630	char

```
other_char      51  3
other_char      51  3
relax          1114112      1117492  relax
```

Here we see a few new tokens, a `char_given` and a call. The first represents a `\chardef` i.e. a reference to a character slot in a font, and the second one a macro that will expand to the \TeX logo. Watch how the space after a control sequence is eaten up. The exclamation mark is a direct reference to character slot 33.

```
\noindent {\bf Hans} \par \hbox{Taco} \endgraf
```

Hans

Taco

cmd	chr	id	name
start_par	0	1141958	noindent
left_brace	123		
call	1650250	1114412	bf
letter	72	H	
letter	97	a	
letter	110	n	
letter	115	s	
right_brace	125		
spacer	32		
par_end	1114112	1114870	par
make_box	122	1115680	hbox
left_brace	123		
letter	84	T	
letter	97	a	
letter	99	c	
letter	111	o	
right_brace	125		
spacer	32		
par_end	1114112	1127274	endgraf

As you can see, some primitives and macros that are bound to them (like `\endgraf`) have an internal representation on top of their name.

```
before \dimen2=10pt after \the\dimen2
```

```
before after 10.0pt
```

cmd	chr	id	name
letter	98	b	
letter	101	e	
letter	102	f	
letter	111	o	
letter	114	r	
letter	101	e	
spacer	32		
register	1	1117302	dimen
other_char	50	2	

```
other_char      61  =
other_char      49  1
other_char      48  0
letter          112  p
letter          116  t
spacer          32
letter          97   a
letter          102  f
letter          116  t
letter          101  e
letter          114  r
spacer          32
the              0      1114887  the
register         1      1117302  dimen
other_char      50  2
```

As you can see, registers are not explicitly named, one needs the associated register code to determine it's character (a dimension in our case).

```
before \inframed[width=3cm]{whatever} after
```

```
before whatever after
```

cmd	chr	id	name
letter	98	b	
letter	101	e	
letter	102	f	
letter	111	o	
letter	114	r	
letter	101	e	
spacer	32		
call	1824889	3226639	inframed
other_char	91	[
letter	119	w	
letter	105	i	
letter	100	d	
letter	116	t	
letter	104	h	
other_char	61	=	
other_char	51	3	
letter	99	c	
letter	109	m	
other_char	93]	
left_brace	123		
letter	119	w	
letter	104	h	
letter	97	a	
letter	116	t	
letter	101	e	
letter	118	v	
letter	101	e	
letter	114	r	
right_brace	125		
spacer	32		

```

letter      97  a
letter     102  f
letter     116  t
letter     101  e
letter     114  r

```

As you can see, even when control sequences are collapsed into a reference, we still end up with many tokens, and because each token has three properties (cmd, chr and id) in practice we end up with more memory used after tokenization.

```
compound|-|word
```

```
compound-word
```

cmd	chr	id	name
letter	99	c	
letter	111	o	
letter	109	m	
letter	112	p	
letter	111	o	
letter	117	u	
letter	110	n	
letter	100	d	
call	1869296	125	
other_char	45	-	
call	1869296	125	
letter	119	w	
letter	111	o	
letter	114	r	
letter	100	d	

This example uses an active character to handle compound words (a ConT_EXt feature).

```
hm, \directlua 0 { tex.sprint("Hello World") }
```

```
hm, Hello World!
```

cmd	chr	id	name
letter	104	h	
letter	109	m	
other_char	44	,	
spacer	32		
convert	23	1166957	directlua
other_char	48	0	
spacer	32		
left_brace	123		
spacer	32		
letter	116	t	
letter	101	e	
letter	120	x	
other_char	46	.	
letter	115	s	

```

letter     112  p
letter     114  r
letter     105  i
letter     110  n
letter     116  t
other_char  40  (
other_char  34  "
letter     72  H
letter     101  e
letter     108  l
letter     108  l
letter     111  o
spacer     32
letter     87  W
letter     111  o
letter     114  r
letter     108  l
letter     100  d
other_char  33  !
other_char  34  "
other_char  41  )
spacer     32
right_brace 125

```

The previous example shows what happens when we include a bit of lua code ... it is just seen as regular input, but when the string is passed to Lua, only the chr property is passed, so we no longer can distinguish between letters and other characters.

A macro definition converts to tokens as follows.

```
[B][A]
```

cmd	chr	id	name
def	0	1114818	def
undefined_cs		1115536	Test
mac_param	35		
other_char	49	1	
mac_param	35		
other_char	50	2	
left_brace	123		
other_char	91	[
mac_param	35		
other_char	50	2	
other_char	93]	
other_char	91	[
mac_param	35		
other_char	49	1	
other_char	93]	
right_brace	125		
spacer	32		
undefined_cs		1115536	Test
left_brace	123		
letter	65	A	
right_brace	125		

```

left_brace    123
letter        66  B
right_brace   125

```

As we already mentioned, a token has three properties. More details can be found in the reference manual so we will not go into much detail here. A stupid callback looks like:

```
callback.register('token_filter', token.get_next)
```

In principle you can call `token.get_next` anytime you want to intercept a token. In that case you can feed back tokens into \TeX by using a trick like:

```

function tex.printlist(data)
  callback.register('token_filter', function ()
    callback.register('token_filter', nil)
    return data
  end)
end

```

Another example of usage is:

```

callback.register('token_filter', function ()
  local t = token.get_next
  local cmd, chr, id = t[1], t[2], t[3]
  -- do something with cmd, chr, id
  return { cmd, chr, id }
end)

```

There is a whole repertoire of related functions, one is `token.create`, which can be used as:

```

tex.printlist{
  token.create("hbox"),
  token.create(utf.byte("{"), 1),
  token.create(utf.byte("?"), 12),
  token.create(utf.byte("}"), 2),
}

```

This results in: ?

While playing with this we made a few auxiliary functions which permit things like:

```

tex.printlist (
  table.unnest ( {
    tokens.hbox,
    tokens.bgroup,
    tokens.letters("12345"),
    tokens.egroup,
  } ) )

```

Unnesting is needed because the result of the `letters` call is a table, and the `printlist` function wants a flattened table.

The result looks like: 12345

cmd	chr	id	name
make_box	122	1115680	hbox
left_brace	123		
letter	49	1	
letter	50	2	
letter	51	3	
letter	52	4	
letter	53	5	
right_brace	125		

In practice, manipulating tokens or constructing lists of tokens this way is rather cumbersome, but at least we now have some kind of access, if only for illustrative purposes.

```
\hbox{12345\hbox{54321}}
```

can also be done by saying:

```
tex.sprint("\hbox{12345\hbox{54321}}")
```

or under Con \TeX 's basic catcode regime:

```

tex.sprint(tex.ctxcatcodes,
  "\hbox{12345\hbox{54321}}")

```

If you like it the hard way:

```

tex.printlist ( table.unnest ( {
  tokens.hbox,
  tokens.bgroup,
  tokens.letters("12345"),
  tokens.hbox,
  tokens.bgroup,
  tokens.letters(string.reverse("12345")),
  tokens.egroup,
  tokens.egroup
} ) )

```

This method may attract those who dislike the traditional \TeX syntax for doing the same thing. Okay, a carefull reader will notice that reversing the string in \TeX takes a bit more trickery, so ...

Hans Hagen

Integrating the pool file

Taco Hoekwater

Abstract

This short article discusses the method that is used in MetaPost and lua \TeX to integrate the string pool file into the program.

This method allows the redistribution of a single updated executable in place of both a program and a data file, and this makes updating those programs easier on both the user and the developer (me).

How a pool file is created

The readers who regularly update their (pdf) \TeX or MetaPost executables will probably be familiar with the concept of pool files already, but I will explain the mechanics in some detail.

Programs written in the WEB language normally do not contain the strings inside the executable proper, but in a separate file, called the 'pool file'.

The most important reason for the existence of this file is that back when Knuth was working on \TeX and Metafont, there was not yet a standardized way to handle strings inside the Pascal language, so he had to invent his own solution for printing messages and warnings.

In order to illustrate what is in a pool file, I will show you the required steps. First, here is a bit of WEB source from MetaPost:

```
...
if minx_val(h)>maxx_val(h) then
  print("0 0 0 0")
else begin
  ps_pair_out(minx_val(h),miny_val(h));
  ps_pair_out(maxx_val(h),maxy_val(h));
end;
print_nl("%%Creator: MetaPost ");
print(metapost_version);
print_nl("%%CreationDate: ");
```

this excerpt is from one of the PostScript output routines. Here, there are still recognizable strings that are used as function arguments (as well as the symbolic value `metapost_version`, that is actually a macro resolving to a string).

The processor tangle converts this input into a proper Pascal source file. While doing so, it resolves all of the many WEB macros that are present in the

code. `metapost_version` is one of those, but also the constructs like `minx_val(h)` and `maxx_val(h)`. It also removes the underscores from function names, because traditional Pascal compilers did not allow `_` to appear in identifiers.

What we are focusing on now, is that it also collects all of the double-quoted strings in the input. It puts all of the unique multi-character strings into an internal array, and replaces the actual string in its output with the index number it has given the string inside that array. Of course, functions like `print()` are written in such a way that they expect numbers as arguments instead of string values.

The Pascal output file looks like this:

```
...
if mem[h+2].int>mem[h+4].int then print(1279)
else begin pspairout(mem[h+2].int,mem[h+3].int);
pspairout(mem[h+4].int,mem[h+5].int);end;
println(1281);print(256);println(1282);
```

As you can see, this file is clearly intended for a compiler only. The complete lack of indentation makes it near impossible for a human to read the generated code, but of course a Pascal compiler has no problem with it.

Nowadays, creating an executable program from the WEB source file happens in a few extra steps, and one of these steps is a conversion from Pascal to C source code, by means of the `web2c` system. You may find the output of `web2c` easier to read, because it re-indents the code for human reading:

```
...
if ( mem [h + 2] .cint > mem [h + 4] .cint )
  print ( 1279 ) ;
else {
  pspairout(mem [h + 2] .cint,mem [h + 3] .cint);
  pspairout(mem [h + 4] .cint,mem [h + 5] .cint);
}
println ( 1281 ) ;
print ( 256 ) ;
println ( 1282 ) ;
```

So, where did the strings go? `tangle` put the multi-character strings into a separate file, in this case named `mp.pool`. Each line of that file contains two digits indicating the length of the string, followed by the string itself. Around line 1000, you will find this:

```
...
070 0 0 0
20%%HiResBoundingBox:
20%%Creator: MetaPost
16%%CreationDate:
...
```

07 is the length in bytes of '0 0 0 0', 20 is the length of '%%HiResBoundingBox: ', including the trailing space character, etcetera. Single character strings are not written to the pool file, because there is no need: all single-character strings simply have an assumed index value matching their contents, and the first string in the pool file receives index number 256.

The Pascal source code (or C source code) is now converted into an executable, and you end up with `mpost.exe` as well as `mp.pool`. The pool file is stored somewhere in the `TEXMF` tree, and one of the very first things that the `--ini` version of `MetaPost` does, is that it reads `mp.pool` to initialize its internal arrays. When the user executes the `dump` command, `MetaPost` writes all of the string items to the `.mem` file, from where it will be retrieved by production runs of `MetaPost`.

There is nothing wrong with this system as such. In fact, it has worked well for nearly 30 years. But it does make updating executables a bit harder than desired: users not only have to copy the actual program to a folder in the path, but they also have to figure out where to place the new and improved `mp.pool` file.

As the maintainer of `MetaPost` and `luaTEX`, both programs that are updated frequently, I was getting annoyed with having to explain to almost each updating user what a pool file was, why it was important, and where it should go in their `TEXMF` tree.

How a pool file disappears again

So I decided to do something about it, and that was how the `makecpool` program was born. The concept is very simple: it converts the `mp.pool` into a C source file named `loadpool.c`. In fact, it is so obvious that the idea has been proposed a number of times already, for instance by Fabrice Popineau. But somehow it has never made it to the core `TEX` distribution yet.

The structure of the created file is straightforward: there is one big static array, and a fairly simple C function that replaces the Pascal procedure for pool file reading. In abbreviated form, `loadpool.c` looks like this:

```
/* This file is auto-generated by makecpool */

#include <stdio.h>
#include "mpdir/mplib.h"

static char *poolfilearr[] = {
    "1.000",
    ...
    "0 0 0 0",
    "%%HiResBoundingBox: ",
    "%%Creator: MetaPost ",
    "%%CreationDate: ",
    ...
    NULL };

int loadpoolstrings (integer spare_size) {
    char *s;
    strnumber g=0;
    int i=0,j=0;
    while ((s = poolfilearr[j++])) {
        int l = strlen (s);
        i += l;
        if (i>=spare_size) return 0;
        while (l-- > 0) strpool[poolptr++] = *s++;
        g = makestring();
        strref[g]= 127;
    }
    return g;
}
```

In the stage where the various C files are compiled into `mpost.exe`, this file is included in the list, and in that way the strings will be embedded in the program. At run-time, the C function is called to put the strings for the C array into the internal storage area instead of the original file reader.

The result: there is only one single executable file that can be freely distributed to the users. The source code for `makecpool` is part of the `MetaPost` and `luaTEX` distribution package.

Taco Hoekwater
taco(a)elvenkind.com

PGF / TikZ

Willi Egger

Abstract

For those who are looking for an alternative for external graphic drawing tools, PGF / TikZ offers a wealth of possibilities. PGF is a macro-package that, together with its user interface TikZ, comprises a kind of "graphics language" to build graphics inside the text as inline graphics or as pictures of larger size. PGF is a macro-package originally written for LaTeX. In the meantime it is also available for use within ConTeXt. The package comes with a large set of libraries for different kinds of graphics. There is extensive documentation and a tutorial. For support a mailing list and web-site are available. Users of the package with ConTeXt have to install the xkeyval package version 1.8. PGF and TikZ are distributed under the GNU Public License version 2.

Introduction

The drawing environment is called PGF. PGF is the acronym for "Portable Graphics Format". The user interface is called TikZ. TikZ stands for "TikZ ist *kein* Zeichenprogramm" meaning it is not an interactive drawing/painting program. The package provides a kind of "graphics language" with which to program a graphic comparable to programming the text in TeX.

PGF / TikZ is a macropackage that was originally written for LaTeX. In the meantime ConTeXt users can also profit from this package and use it within ConTeXt.

The package is written and maintained by Till Tantau, professor at the Institut für Theoretische Informatik, Universität Lübeck.

The macropackage is distributed under the GNU Public License version 2.

Installation

LaTeX

Because the package is basically a LaTeX-package, it is probably already installed on your system. Otherwise install it as usual in the TeX tree e.g. from CTAN.

ConTeXt

The easiest way to install the package under ConTeXt is to download it and unpack it in a temporary folder. Move the contents of the *generic* folder to `... \tex\generic`. Move the contents of the folder *context* to `... \tex\context\third` (the third party module folder). In order to get the package working you need to install the xkeyval version 1.8 package written by Hendri Adriaens. Download it from CTAN. Install the files either in `... \tex\generic` or `... \tex\latex`. Move the folder *doc* to `... /doc`. Run `mktextlsr` in order to update the file-database.

Plain TeX

Provided that you use a full installation of TeX, like the TeX-live distribution, the package might be installed already. Otherwise, install the necessary files contained in the archive into the respective folders of the TeX-tree. For use of the package under Plain TeX you need also the xkeyval package version 1.8 or above by Hendri Adriaens and the xcolor package version 2.0 by Uwe Kern.

Syntax differences

The graphics language syntax for the different \TeX -environments is generally set up in such a way that the user can use the commands as he is accustomed to do in his \TeX -environment:

In order to use the package, it must first be loaded:

La \TeX	Con \TeX t	Plain \TeX
<code>\usepackage{tikz}</code>	<code>\usemodule[tikz]</code>	<code>\input tikz.tex</code>

Starting a sequence of commands for a graphic:

La \TeX	Con \TeX t	Plain \TeX
<code>\begin{tikzpicture}</code>	<code>\starttikzpicture</code>	<code>\tikzpicture</code>
<code>...</code>	<code>...</code>	<code>...</code>
<code>\end{tikzpicture}</code>	<code>\stoptikzpicture</code>	<code>\endtikzpicture</code>

Structure of the package

The package is built in three layers: system, basic and front end. The system-layer provides the highest abstraction level, i.e. it provides the translations of commands into `\special`-commands as required by the different driver environments (dvips, dvipdfm, pdftex). This layer is not intended for use by the user.

The second layer is called basic-layer. This layer provides a set of commands for building complex graphics without being obliged to use the syntax of the system layer.

Lastly, the front end layer provides a set of commands which make the use of the basic layer more convenient. The most natural front end is TikZ. For illustration, if you wanted to draw e.g. a triangle with the basic layer you would have to issue up to 5 commands, whereas with the TikZ-front end it will suffice to say `\draw (0,0)--(1,0)--(1,1)--cycle;`.

TikZ

Defining points in a graphic

There are different possibilities to define a point in a graphic. First, a point can be defined by giving the coordinates in dimensions known to \TeX inside a pair of round brackets e.g. `(2cm,10pt)`. This represents the xy-coordinate system. Giving three dimensions inside a pair of round brackets `(1,1,1)` will invoke the xyz-coordinate system. Omitting the unit of a dimension will cause TikZ to use the predefined dimension of 1cm. A third way of defining a point is to indicate a vector in the form of `(30:1cm)`. This will cause TikZ to move 1cm in the direction of 30 degrees. Finally, points can be specified relative to another point. e.g. `(1,0) ++(1,0) ++(0,1)` specifies the following movements: `(1,0) (2,0) (2,1)`. One can also define a relative point by adding a single `+`: `(1,0) +(1,0) +(0,1)` - which defines the following coordinates: `(1,0) (2,0) (1,1)`. The difference is, that the points defined with a single `+` will not change the current point which is in the example `(1,0)`.

Paths

Paths consisting of a series of straight and curved elements are defined similar to MetaPost. Path elements do not necessarily need to be connected to each other. Actions on (closed) paths are draw, fill, shade and clip. These actions can be applied to paths in any combination.

Grouping

Within a TikZ-picture grouping of elements can be achieved by using scopes

La \TeX : `\begin{scope}[options] ... \end{scope}` and

Con \TeX t: `\startscope[options] ... \stopscope`.

The application of styles enables the user to apply options defined outside the graphic to (part of) the graphic.

Transformations

Shifting an object can be done with `\shift{1,2}` or `\shift{+1,2}` i.e. with relative positioning. There is also `xshift=10pt` and `yshift=1cm`. Objects can be rotated with the command `rotate` or rotated around a given point with `rotated around`. `scale=2` will scale the object or picture by the given factor. There is also `xscale` and `yscale` for scaling in a single direction.

For-loops

LaTeX-users can issue the built in loop constructs or the `\multido` command from pstricks. ConTeXt-users can use the `\dorecurse{}{}` looping mechanism. TikZ offers yet another for-loop, based on series represented by dimensionless real numbers, which are given as `{1,2,3}` or `{1,...,10}` or even `{1,3,...,20}`. In case two figures are given before the ellipsis, the difference between the two figures is used as step. The basic command is

```
\foreach \variablename in {...series...}
  \draw(\variablename pt, -1pt) -- (\variablename pt,1pt)
```

Text and Labels

Text can be added to any given point of a path. When TikZ encounters the keyword `node` during the construction of a path, the elements of the node will be added as a TeX-box after the complete path is ready. A node consists of the keyword `node` followed by options between square-brackets and followed by the text enclosed in curly braces.

Node-options are numerous: `left/right/below/above` = *dimension*, anchors are `north`, `south`, `west`, `east`, `north east` ... Furthermore text may be positioned along the path with the option `sloped`. Texts may be moved towards the beginning or end of a path by `very near` or `near end`. These options can be combined with `above` etc. and `sloped`.

Libraries

TikZ comes with a large set of specialized libraries. Each library needs to be loaded separately: Libraries are loaded by `\usetikzlibrary{library-name}` in LaTeX and `\usetikzlibrary[library-name]` in ConTeXt.

- Arrow Tip Library, *arrows*
This library contains a large pallet of different tip-styles like triangular tips, barbed and bracket-like, circle and diamond like, partial tips and line caps.
- Automata Drawing Library, *automata*
For the drawing of finite automata and Turing machines.
- Background Library, *backgrounds*
Various background types are provided.
- Entity-Relationship Drawing Library, *er*
This library will be loaded for drawing E/R diagrams.
- Mind-map Library, *mindmap*
For those who need a nice presentation of their mind-maps, this library is useful.
- Pattern Library, *patterns*
For filling shapes with pattern this library is loaded.
- Petri-Net Library, *petri*
For the construction of Petri-nets this library is needed.
- Plot Handler Library, *plothandlers*
The plot handler library is loaded automatically by TikZ.

- Plot Marks Library, *plotmarks*
This library defines additional plot marks next to the standard marks, which are $*$, x and $+$.
- Shape Library, *shapes*
The shapes library contains a number of predefined shapes.
- Snake Library, *snakes*
This library defines a series of non-straight lines like coils, braces, bumps, expanding waves, saw or yes, snake lines.
- To Path Library, *topaths*
This library is loaded automatically by TikZ and provides predefined to paths which are used in the to path operation.
- Tree Library, *trees*
This library provides different styles to draw trees.



Examples

Below are some examples that demonstrate what TikZ can do. The examples are taken from the manual, and are formatted for ConTeXt. LaTeX-users can best refer to the manual, because examples are given there in LaTeX-code.

Using TikZ code inline in the text

For drawing a sloped line within the text row you would type

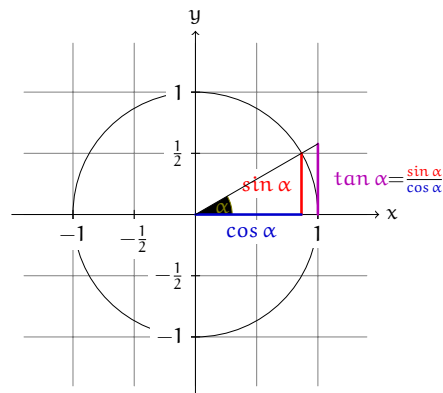
```
\tikz \draw(0pt,0pt) -- (20pt,6pt);
```

and get . Here a big grey dot should be placed . This is coded as

```
\tikz \fill[black!60] circle (1ex);
```

.

Trigonometry



The angle α is 30° in the example ($\pi/6$ in radians). The **sine** of α , which is the height of the red line, is $[\sin \alpha = 1/2.]$ By the Theorem of Pythagoras ...

Figure 1. Trigonometry

```
\usemodule[tikz]
\starttikzpicture[scale=2, cap=round]
  % Local definitions
  \def\costhirty{0.8660256}
  % Colors
  \definecolor[anglecolor] [r=.5,g=.5,b=0]
  \definecolor[sincolor] [r=1,g=0,b=0]
  \definecolor[tancolor] [r=.7,g=0,b=.7]
  \definecolor[coscolor] [r=0,g=0,b=.8]
  \definecolor[fillcolor] [.625black]
  % Styles
```

```

\tikzstyle{axes}=[]
\tikzstyle{important line}=[very thick]
\tikzstyle{information text}=[rounded corners,inner sep=1ex]
% The graphic
\draw[style=help lines,step=0.5cm] (-1.4,-1.4) grid (1.4,1.4);
\draw (0,0) circle (1cm);
\startscope[style=axes]
  \draw[>-] (-1.5,0) -- (1.5,0) node[right] {$x$} coordinate(x axis);
  \draw[>-] (0,-1.5) -- (0,1.5) node[above] {$y$} coordinate(y axis);
  \foreach \x/\xtext in {-1, -.5/-\frac{1}{2}, 1}
  \draw[xshift=\x cm] (0pt,1pt) -- (0pt,-1pt) node[below,fill=white]
    {$\xtext$};
  \foreach \y/\ytext in {-1, -.5/-\frac{1}{2}, .5/\frac{1}{2}, 1}
  \draw[yshift=\y cm] (1pt,0pt) -- (-1pt,0pt) node[left,fill=white]
    {$\ytext$};
\stopscope
\filldraw[fill=green!20,draw=anglecolor] (0,0) -- (3mm,0pt) arc(0:30:3mm);
\draw (15:2mm) node[anglecolor] {$\alpha$};
\draw[style=important line,sincolor] (30:1cm) -- node[left=1pt,fill=white]
  {$\sin \alpha$} (30:1cm |- x axis);
\draw[style=important line,coscolor]
  (30:1cm |- x axis) -- node[below=2pt,fill=white] {$\cos \alpha$} (0,0);
\draw[style=important line,tancolor] (1,0) -- node[right=1pt,fill=white]
  {$\tan \alpha$ {\color{black}=}
    \frac{{\color{sincolor}\sin \alpha}}{{\color{coscolor}\cos \alpha}}}
  (intersection of 0,0--30:1cm and 1,0--1,1) coordinate (t);
\draw (0,0) -- (t);
\draw[xshift=2.5cm]
node[right,text width=6cm,style=information text]
{
  The {\color{anglecolor} angle $\alpha$} is $30^\circ$ in the
  example ($\pi/6$ in radians). The {\color{sincolor}sine of
  $\alpha$}, which is the height of the red line, is
  [{\color{sincolor} $\sin \alpha$} = 1/2.]

  By the Theorem of Pythagoras ...
};
\stoptikzpicture

```

Simple organigramma / tree

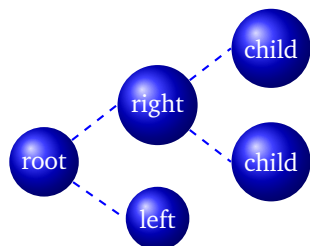


Figure 2. A tree

```

\usemodule[tikz]
\usetikzlibrary[arrows,snakes,backgrounds,trees]
\starttikzpicture[parent anchor=east,child anchor=west,grow=east]
  \tikzstyle{every node}=[ball color=blue,circle,text=white]
  \tikzstyle{edge from parent}=[draw,dashed,thick,blue]

```

```

\begin{tikzpicture}
\node {root}
  child {node {left}}
  child {node {right}}
  child {node {child}}
  child {node {child}}
};
\stoptikzpicture

```

TikZ in cooperation with GNUplot

One can use GNUplot to calculate the points needed in a path. TikZ will, after reading its instructions, prepare a command file containing GNUplot commands describing the function to be drawn. In a second run GNUplot is invoked with these commands from which a data file is produced. TikZ imports this data file and draws the graph.

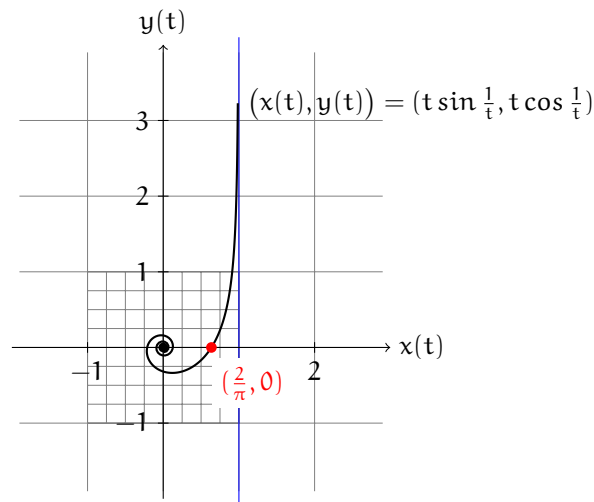


Figure 3. A function calculated by GNUplot

```

\usemodule[tikz]
\starttikzpicture
\draw[gray,very thin] (-1.9,-1.9) grid (2.9,3.9)
  [step=0.25cm] (-1,-1) grid (1,1);
\draw[blue] (1,-2.1) -- (1,4.1);
\draw[->] (-2,0) -- (3,0) node[right] {$x(t)$};
\draw[->] (0,-2) -- (0,4) node[above] {$y(t)$};
\foreach \pos in {-1,2}
  \draw[shift={(\pos,0)}] (0pt,2pt) -- (0pt,-2pt) node[below] {$\pos$};
\foreach \pos in {-1,1,2,3}
  \draw[shift={(0,\pos)}] (2pt,0pt) -- (-2pt,0pt) node[left] {$\pos$};
\fill (0,0) circle (0.064cm);
\draw[thick,parametric,domain=0.4:1.5,samples=200]
  function{(t*t*t)*sin(1/(t*t*t)), (t*t*t)*cos(1/(t*t*t))}
  node[right]
  {$\bigl(x(t),y(t)\bigr) = (t\sin \frac{1}{t}, t\cos \frac{1}{t})$};
\fill[red] (0.63662,0) circle (2pt)

```



```
node [below right,fill=white,yshift=-4pt] {$(\frac{2}{\pi},0)$};
\stoptikzpicture
```

The contents of the file created by GNUplot looks as follows:

```
#Curve 0, 200 points
#x y type
0.00530 -0.06378 i
0.04363 -0.05043 i
0.06711 -0.01790 i
0.06896 0.02170 i
0.05014 0.05606 i
0.01712 0.07631 i
-0.02110 0.07849 i
-0.05579 0.06337 i
-0.08032 0.03512 i
-0.09097 -0.00029 i
-0.08696 -0.03664 i
-0.06987 -0.06850 i
-0.04284 -0.09192 i
-0.00982 -0.10460 i
0.02515 -0.10585 i
0.05841 -0.09629 i
...
0.98511 3.18914 i
0.98543 3.22793 i
```

Internet

More information on PGF and TikZ can easily be found on the Internet. For a stable release of the package one can visit CTAN or from <http://sourceforge.net/projects/pgf/>.

If you want to get the most recent developments you can fetch the latest version from CVS. The command would be something like

```
cvs -z3 -d:pserver:anonymous@pgf.cvs.sourceforge.net:/cvsroot/pgf
                                -co checkout pgf
```

For extensive examples there is a web-site by Kjell Magne Fauske, Norway:

<http://www.fauskes.net/pgftikzexamples>

For support one can join the following mailing-list. Visit

<https://lists.sourceforge.net/lists/listinfo/pgf-users>

for subscription.

Summary

TikZ is a tool for making various kinds of drawings. For \TeX -users the style of setting up such drawings is familiar, because you program a drawing similarly to how you program a \TeX -text. TikZ is the natural front end to lower level PGF functionality. On top of this, it is possible to draw graphs using points generated by GNUplot.

Acknowledgement

I would like to thank Michael Guravage for looking through the text and turn it into correct English.

Willi Egger

External graphics for LaTeX

Siep Kroonenberg
N.S.Kroonenberg at rug dot nl

Abstract

In this article, we discuss graphics file formats, software to create graphics and procedures to convert them to LaTeX- and pdf_latex-compatible formats.

Keywords

Graphics converting bitmap vector compression eps pdf jpeg lossy lossless resolution

This article is about preparing external graphics for use with LaTeX.

We start out with a quick overview of types of graphics. If you understand what kind of data you are dealing with, you will have a much better chance of getting good results.

Next, we list programs for creating graphics, both free and commercial.

The final part is about programs and procedures for converting graphics into LaTeX-compatible formats.

1 Types of graphics

Graphics can be defined in different ways, depending on the type of information they contain and on the software with which they have been created. Figures 1–6 contain some examples, each together with an enlarged detail.

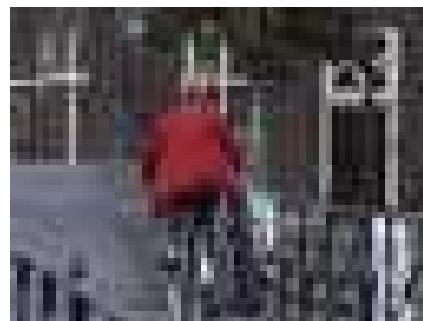


Figure 1. Bitmapped art: a photograph

A bitmap is built up as a grid of pixels. Figures 1 and 2 show a photograph and a screenshot respectively. The grid structure is obvious in the enlarged detail.

Vector graphics are defined in terms of lines, circles, curves and other geometric shapes. They keep their sharpness at any scale; see figures 3–5.

Some file formats can contain both bitmapped and vector data. In figure 6, the bitmapped background becomes fuzzy when enlarged, but the text on top remains sharp.

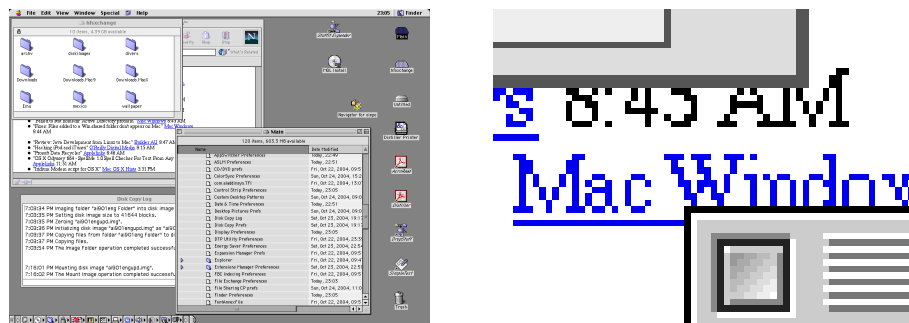


Figure 2. Bitmapped art: screenshot

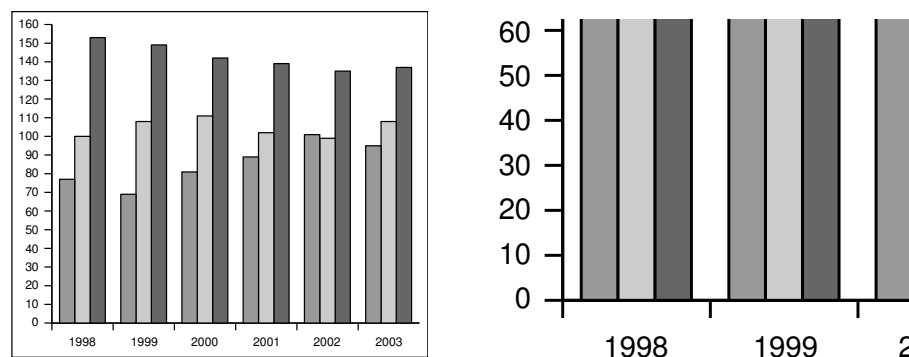


Figure 3. Vector art: OpenOffice.org graph



Figure 4. Vector art: adapted from a Ghostscript example file

1.1 Jpeg compression

High-resolution bitmapped files can get very big. There are various ways to reduce those file sizes.

Lossless compression works by storing information in a more compact way. A very simplified example: instead of enumerating a thousand identical white pixels one by one, you can say at once that the next one thousand pixels are white. Lossless compression can be quite effective when there are large areas of solid colors or regular patterns. Png is a lossless bitmapped format that can be processed directly by pdf_latex.

Lossless compression doesn't work so well with photographic images. When we no longer insist on exactly preserving every bit of information and accept lossy

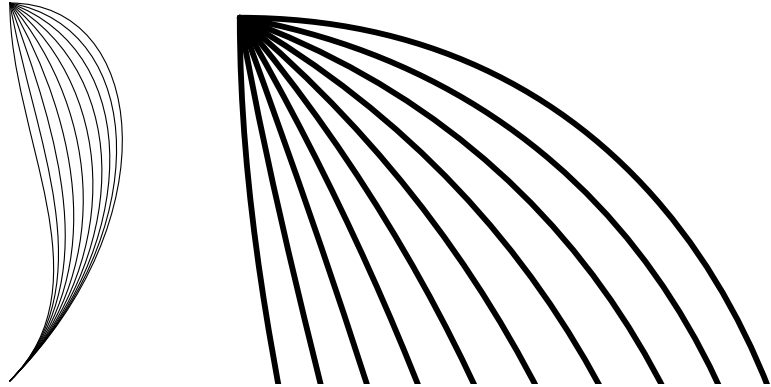


Figure 5. Vector art: generated with MetaPost



Figure 6. Bitmapped and vector combined



Figure 7. Don't use jpeg compression for screenshots.

compression, then very good results can be reached with jpeg. At medium to high quality settings, the loss of information is essentially invisible, but the compression rate is easily 10:1. When you save in jpeg format you can usually make your own tradeoff between file size and image quality.

For non-photographic bitmapped images such as screenshots or logos, jpeg compression produces visible artifacts; see figure 7. To be fair, for the right image quality was set very low in order to make the artifacts more obvious. Here, compression isn't all that good either compared to png. Nevertheless, many people use jpeg compression indiscriminately, even when png would have been much better.¹

1.2 Bitmap resolutions

The resolution of a bitmap should be high enough to look sharp, but, in order to keep file size and loading times within bounds, the resolution should not be higher

than necessary.

For screen viewing, the ideal resolution would be exactly one bitmap pixel per screen pixel, but of course you may not know at what screen resolution and zoom level your document will be viewed.

For printing, good resolutions are:

Photographs. 150–300 DPI (dots per inch) depending on the output device. Most printers and imagesetters simulate grays and other tints with dot patterns or halftone screens. As a consequence, the effective printed resolution of a photograph is much lower than the resolution of the output device, no matter how high the resolution of the original photograph.

Charts and diagrams. 600–1200 DPI. 600 is enough to avoid visible blockiness. Higher resolutions can mean finer detail, if the printer resolution is also high. But vector formats are better for such graphics.

Screenshots. Keep the original resolution.

Of course I am talking about resolution after scaling; if you print a 2" wide, 300 DPI image at a width of 4" then the effective resolution is 150 DPI.

It will do no good to increase the resolution of an existing low-resolution image; it might even make the output fuzzier. So either try to find or create a better original, or use your bad picture as-is.

1.3 Problems with vector graphics

Missing fonts. If some standard fonts (Times, Helvetica, Courier, Symbol, Zapf Dingbat) are not embedded then `epspdfk` (see section 3.2) can help. If the pdf output target is set to prepress then fonts will be embedded during conversion to pdf. `Epspdfk` can convert back and forth between eps and PostScript on the one hand and pdf on the other. You can either apply a pdf-to-eps-to-pdf conversion to individual graphics or a pdf-to-ps-to-pdf conversion to the document as a whole.

If other fonts are missing then you have a real problem.

If you don't mind using Ghostscript from the command-line: the `epswrite` output device replaces characters with little drawings of their shapes: a fool-proof way to get rid of font problems. Don't do this with large amounts of text. I guess it only works if the font is embedded or is known to Ghostscript in some other way.

Zero-width lines. If your graphic has some very thin lines, then check by zooming in whether the lines have some positive width. A line width of zero will be interpreted by the output device as a width of one pixel, which is fine for the screen or for an old 300 DPI laserprinter, but not for a 2400 DPI imagesetter. A line width of e.g. 0.3pt should be safe. You may be able to fix this from within the program with which the graphic was created.

Transparency and fill patterns. Many programs and graphic formats do not support transparency or fill patterns; upon conversion these features might either get lost altogether or simulated with e.g. bitmaps, which might make the file much larger and virtually ineditible. Hang on to the original – which you should do in any case!

General fixes. You may be able to import and fix problems in a draw program: substituting fonts, changing line widths, replacing pattern fills with something else; see section 2 on draw programs.

As a last resort, you can convert your graphic to a bitmapped png file of sufficiently high resolution.

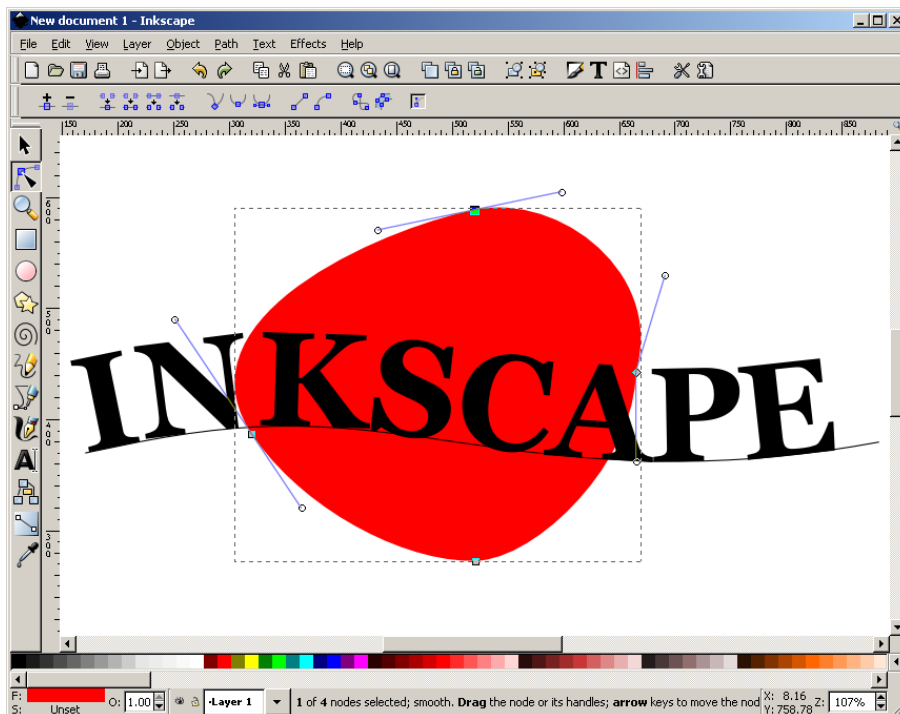


Figure 8. Inkscape, a free vector draw program for Linux, Windows and Mac OS X

2 Creating graphics

In computer graphics, the term drawing means vector art and painting means bitmapped art.

2.1 Drawings and diagrams

You can create drawings and diagrams in Word or PowerPoint. However, it may be difficult to convert graphics from these programs intact to eps or pdf, the formats needed by LaTeX and pdflatex² respectively. Instead, check out some free alternatives such as Inkscape, which is a specialized draw program resembling CorelDRAW, or the Draw module of OpenOffice. OpenOffice.org is fairly good at reading and writing MS Office files, and has a pdf export option.

If you spend a lot of time creating and editing vector graphics, consider buying a professional draw program such as CorelDRAW or Adobe Illustrator.

Under Mac OS X there are several less expensive commercial programs worth checking out, such as OmniGraffle from the Omni Group (www.omnigroup.com), Create from Stone Design (www.stone.com) and Intaglio from Purgatory Design (www.purgatorydesign.com).

In any case, investigate at an early stage how to get your drawings into LaTeX, see section 3.

2.2 Draw programs with LaTeX support

Draw programs which can use TeX for typesetting text include Ipe (cross-platform) and TpX (Windows-only). TpX is also available from CTAN. Both are free.

Section 3.3 has some information on Ipe file formats.

TpX generates LaTeX code, which you have to load into your document with an `\input` command. The various output options of TpX are a bit too complicated to explain here, but you can read all about it in the TpX online help.

2.3 Charts

Charts are normally generated as a byproduct of spreadsheets or mathematical, statistical or econometric software.

2.4 Bitmaps: paint programs and image editors

There exists a large selection of free and inexpensive paint programs and image editors. Paint is a very basic paint program which is included with Windows. IrfanView is a small, simple and free image viewer and converter for Windows. If you need something more substantial, have a look at the GIMP, which was originally developed for Linux. Yet another option is Adobe PhotoShop Elements, which may be easier to work with. It is commercial, but costs a fraction of its professional big brother, PhotoShop, which is the favorite of professional designers. A favorite on the Macintosh is GraphicConverter.

2.5 Screenshots

You can take screenshots without specialized software.

Windows. The PrtScrn key will copy the entire screen to the clipboard, and Alt-PrtScrn the active window. Most paint- and image-editing programs can retrieve the screenshot from the clipboard, usually with Edit / Paste. They also have tools to crop images.³

Linux. Here, PrtScrn will probably take a screenshot as well, either to the clipboard or to a file. I usually take screenshots with the Gimp (File > Acquire > Screenshot).

You can also take a screenshot from the commandline, with `import` from the ImageMagick suite: type `import file.png` and click either a window or the desktop background.

Mac OS X. Cmd-Shift-3 copies the entire screen to a file on your desktop, and Cmd-Shift-4 lets you make a selection.

3 Converting to LaTeX- and pdflatex-compatible formats

I'll only discuss graphics usage for the two most popular output options: generating PostScript with LaTeX and dvips, and generating pdf with pdflatex. The `graphicx` package will automatically detect these two cases. In the LaTeX-plus-dvips case, it will look for graphics in eps format, and in the pdflatex case it will look for graphics in pdf, jpg and png format.⁴

These are stable formats, with little room for ambiguity. Eps and pdf can contain just about any kind of graphics information. This small selection of file formats is therefore not a real limitation, but it can mean extra work.

3.1 Converting bitmaps to png and jpg

Many image editors and paint programs, including IrfanView, can convert to png and jpg. Convert from the ImageMagick suite is a command-line option. ImageMagick also has a GUI viewer and converter: `display` under Unix/X11 and `IMDisplay` under Windows; see Figure 9.

3.2 Converting between PostScript, eps and pdf

Epspdf and epspdf. These utilities can convert between PostScript, eps and pdf, often with no loss of information. They can also remove unwanted borders (compute tight boundingbox option). `Epspdf` is a GUI program, and `epspdf` is its command-line counterpart.

epstopdf. An alternative for converting eps to pdf is the command-line program `epstopdf`, which is part of most TeX distributions and is probably already on our system.

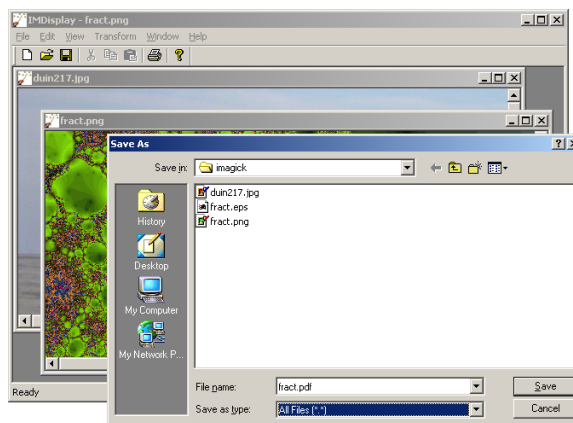


Figure 9. IMDisplay: ImageMagick's GUI component for Windows

convert. On many systems, convert from the ImageMagick suite will use Ghostscript when converting between eps and pdf, and will in that case convert vector graphics to vector graphics. Prefix the target pdf filename with EPDF: if you want to preserve the eps boundingbox. View the result at a high zoom level to make sure that it is still a vector graphic.

Ghostscript and pdftops. You can also use Ghostscript directly for converting eps and PostScript to pdf, and either Ghostscript or pdftops from the xpdf suite for converting in the other direction. But this is not the place for explaining the use of these programs.

3.3 Working with Ipe files

Ipe, see section 2.2, can store drawings in its own private format, with extension ipe or xml, but it can also save them in eps- and pdf format. Ipe drawings in eps- and pdf format have private information hidden inside, which makes it possible for Ipe to read them back in. But if you convert those files to something else then this private information will get lost.

Ipe cannot read arbitrary eps or pdf, but it comes with a command-line import utility pdftoipe. You can put a shortcut to it on your desktop and then simply drop a pdf file on it. The result of the conversion will be in the same directory as the original. You will very likely have to do some cleanup afterwards, especially if there is a lot of text.

3.4 Converting bitmaps to eps

The simpler paint programs and image editors don't convert to eps or pdf, but more advanced ones, such as the GIMP, do. Convert and [IM]Display from the ImageMagick suite also convert to eps and pdf.

You can also convert to eps or pdf with Ipe: create a new, empty document. This happens automatically when you start the program. Select File, Insert image... and load the image that you want to convert. Now save the Ipe document in eps- or pdf format.

3.5 Wmf, emf and the Windows clipboard

Wmf and emf are native vector formats for Windows, and can be read by most Windows graphics programs, including TpX, which I already mentioned in section 2. Another option is Wmf2eps. This shareware program does exactly what its name implies, and its conversions are quite accurate. It uses Windows' native PostScript printerdriver in the background.⁵

If you cannot even save as wmf or emf then again both TpX and Wmf2eps can copy the clipboard content to a file. Note that for TpX you do this with Tools / Capture EMF, not with Paste from the Edit menu.

3.6 Exporting eps and PostScript from Windows programs

If a Windows program doesn't have a usable export option, then you can try to 'print' to a PostScript file. This is approximately what the above wmf2eps program does.⁶

For this, you need to have a PostScript printer driver. If you don't have one installed, go to 'Printers' and start up the Add Printer wizard. Choose Local Printer and uncheck automatic detection. As printer port, you can pick FILE, otherwise you would have to manually check 'Print to File' anytime you print. A good choice for manufacturer and model would be 'Generic' and 'MS Publisher Imagesetter' respectively.

Pay attention to printer settings: in the Print dialog, click 'Properties', then 'Advanced' (on either tab). In the 'Advanced Document Settings' tree, under 'Graphic', 'TrueType Font' should be set to 'Download as Softfont'⁷

Now navigate to first 'Document Options', then 'PostScript Options'. For 'PostScript Output Option' the default setting is 'Optimize for speed'. Change that to 'Optimize for Portability' or 'Archive Format', or, for single pages only, 'Encapsulated PostScript'. These non-default options presumably produce cleaner PostScript code, without printer-specific hacks. Experiment with this and other options if you run into problems (e.g. bad-looking screen output, or part of a graphic getting cut off, or conversion to bitmap).

What works best may depend on your Windows version: under Windows 2000, Archive worked best for me, but I have been warned that this option was unusable in older Windows versions.

Next, the setting 'TrueType Font Downloading Option' should be set to 'Outline', not 'Automatic' or 'Bitmap'.

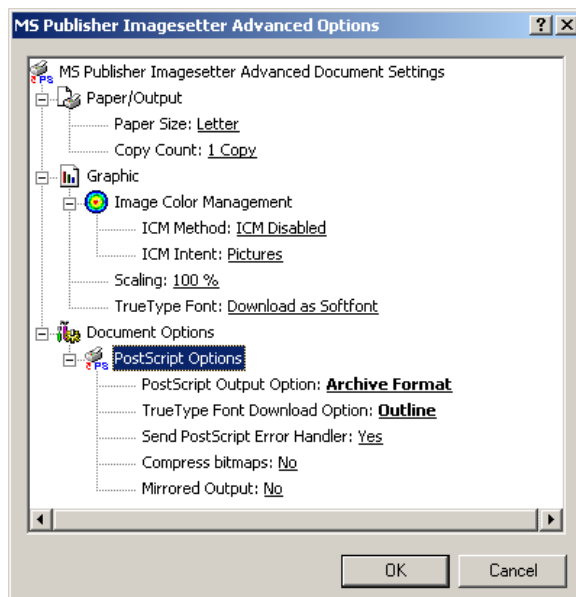


Figure 10. configuring a PostScript printer driver

Notes

1. Jpeg2000 (extension .jp2 or .j2c) is a successor of the jpeg format. This type of compression supports both lossy and lossless compression. Pdflatex can use pdf graphics which internally use jpeg2000 compression, but at the moment you cannot use jpeg2000 images directly.
2. Although pdflatex can also use the bitmapped jpeg and png file formats, we would rather not convert vector graphics to bitmaps.
3. With IrfanView this is not quite obvious, but if you just start to drag the cursor then it changes into a selection tool, and then you can apply Edit / Crop selection.
4. If you specify a non-default output driver as optional parameter to the graphicx package, then different graphics formats may be supported.
5. Most Windows PostScript drivers use the same core. Individual PostScript drivers add to this core a ppd- or PostScript Printer Definition file which is basically an enumeration of the printer's features and properties. Wmf2eps comes with its own ppd.
6. This section is lifted almost literally from my article 'Epspdf, easy conversion between PostScript en Pdf' in Maps 34.
7. The alternative setting is 'Substitute with Device Font'. For prepress use, you should always include all fonts. It is possible to include fonts after the fact, when converting to pdf, but then you run the risk of noticeable discrepancies between the original font and the actually included font.

References

- CTAN, The Comprehensive T_EX Archive Network. <http://www.ctan.org/>.
- Epspdf and epspdfk. <http://tex.aanhet.net/epspdf/>.
- Ghostscript Homepage. <http://www.cs.wisc.edu/~ghost/>.
- ImageMagick Homepage. <http://imagemagick.org/>.
- Inkscape draw program. <http://inkscape.org/>.
- Ipe draw program. <http://tclab.kaist.ac.kr/ipe/>.
- IrfanView image viewer and converter for Windows. <http://www.irfanview.com/>.
- OpenOffice.org homepage. <http://www.openoffice.org/>.
- The Gimp image editor for Windows. <http://gimp-win.sourceforge.net/>.
- TpX draw program. <http://sourceforge.net/projects/tpx/>.
- Wmf2eps Homepage. <http://www.wmf2eps.de.vu/>, wmf2eps can be downloaded from CTAN.
- Xpdf Homepage. <http://www.foolabs.com/xpdf/>.

Siep Kroonenberg
N.S.Kroonenberg at rug dot nl

Alphabetgeschichten

Eine Chronik technischer Entwicklungen

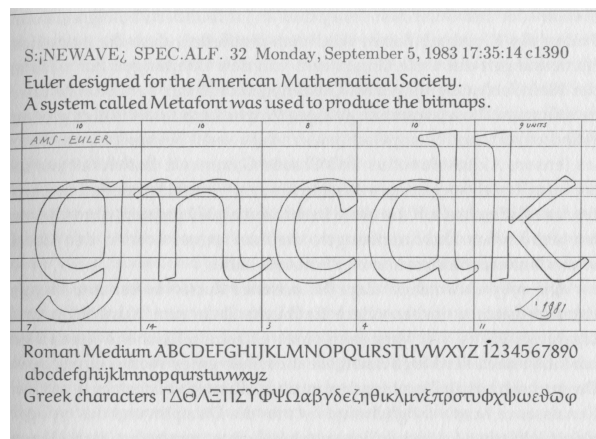


Hans Hagen & Taco Hoekwater

Introduction

It pays off to be a Dante member! Some time ago each member received a copy of Hermann Zapf's monograph 'Alphabetgeschichten', a gift from Hermann himself. For many users of computers the name 'Zapf' may ring a bell because of the omnipresent Zapf dingbats fonts. But with Hermann Zapf being one of the greatest designers of our time, there is much more to learn about him.

Being an honorary member of Dante, Hermann is quite familiar with $\text{T}_{\text{E}}\text{X}$ and friends, and he is in contact with several $\text{T}_{\text{E}}\text{X}$ ies. He worked with Donald Knuth on the book '3:16', a calligraphic masterpiece. He is also responsible for the design of the Euler font family (we will tell you more about this in an upcoming Maps issue). In the recent font projects (Latin Modern and $\text{T}_{\text{E}}\text{X}$ Gyre) we consult Hermann on matters that we are unsure about.



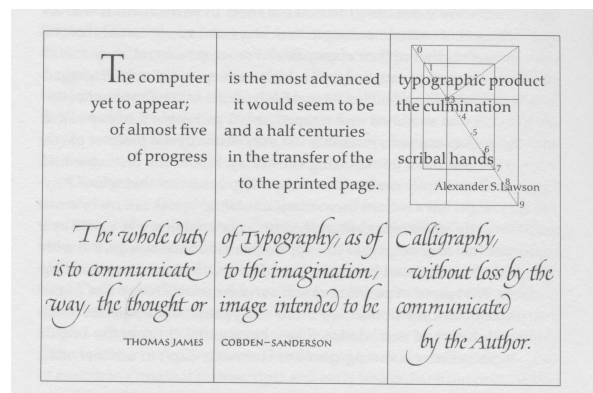
Two versions

There are two versions of this book, the German version and an English translation and it is a pleasure

to have both, especially because they are not entirely the same. The German version has a few more pages than the English translation. And not only because of the language, there are also true differences in the contents.

Born at November 8, 1918 Hermann has grown up in and been a witness of turbulent times. The German version sheds more light on how difficult it was to survive in these times and how much art got lost in that period. He wrote down nice anecdotes about this era, for instance how the ability to write in 1 mm script impressed his army superiors so much that it kept him out of trouble. Both books have some differences in the graphics that go with that period and in the English version some quotes are shortened.

The English book catches up on its last pages. Since 1977 Hermann Zapf is an associate professor at the Rochester Institute of Technology. In the postscript to this version the curator describes the influence Hermann has had on them in the past 30 years. At the time we write this review, Hermann is visiting this institute, where he is involved in a calligraphic and typographic display on 27 glass panels surrounding the new facilities.



If you manage to lay hands on a copy, you will notice that it's printed on thick cream-colored paper and very well bound in a dark blue hard cover with gold initials on the front. At the traditional Dante Christmas Party in 2006 in Darmstadt, Herman told the audience that nowadays it's not trivial to get such paper in the quantities needed: most paper plants only produce paper of moderate quality in any bulk. But here, large quantities of special paper were needed; keep in mind that he gave away a free copy to each of the more than 2000 Dante members.

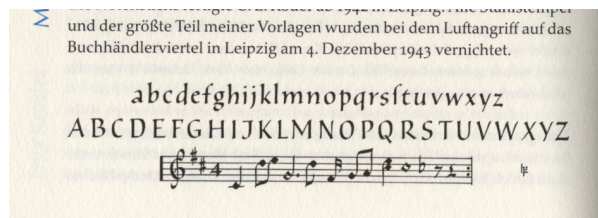
Other interesting differences between the versions are in paragraph breaks and whitespace. As with Dutch, German needs a few more words than English to express ideas, but the general impression is that the German version is the most informative. Other subtle differences are in the used technical terms. The English version qualifies Palatino Sans as 'sans serif', but the German text talks about 'Grotesk'.

A lifetime

One possible reason why Hermann has always been able to catch up with technology and could adapt quite well to the transition from lead to computer, was that originally he wanted to be an 'Elektroingenieur', but calligraphy attracted him more.

Hermann was never stuck on characters only. The book starts with a colorful full-page illustration of flowers and small beetles.

Also, in his early period he created a few 'Notenschriften'. The book shows many examples of handwriting and the grand finale is Zapfino, which is available as a OpenType font with many (complex) features.



Greek, Arab, you name it ... he draws it. Among his most well known fonts are Optima and Palatino. Both fonts date back half a century when lead was still leading, but they were recently redesigned to take advantage of new technologies. Last year a sans serif family named Palatino Sans was added, and an Arab variant is in the making.

The first Optima was drafted on thousand lire notes in 1950. In 1975, this font was used for the Vietnam Veterans Memorial in Washington.

Hermann spent quite some time in the USA, running his own company there, teaching at several designer



schools and working with Donald Knuth. He is still associated with the Rochester Institute of Technology in New York.

In pdf_{TeX} there is a feature that informally is called 'hz-optimization'. This feature is inspired by the work of Hermann on the 'hz-Programm' and in the book Hàn Thế Thành's work and Hermann's communication with Hàn Thế Thành are explicitly mentioned.

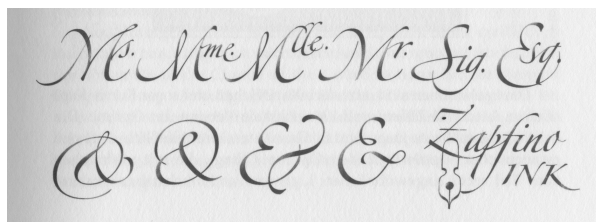
Although an old printing press has a prominent position in his house in Darmstadt, Hermann has always been involved in new technologies. He went from typesetting in lead to using phototypesetters to computer based typesetting. The Zapfino font, that adapts its choice of glyphs to the circumstances is a prime example of this. Steve Jobs of Apple Computers made sure that on this platform the Zapfino behaves how it should behave.

For those who use the dingbats there is good news as well. The Zapf Essentials are the improved and extended version of these symbols. We now finally have everything available that Hermann originally had



in mind when he started drafting this symbol set.

The book also shows samples of Zapfino Ink, yet another innovation. Here color and shades make their way into the font but we have to wait till the font technologies are ready for that. The book tells us that this is being worked on.



The book is typeset in Palatino Nova with displayed quotations in the brand new Palatino Sans. In the not too wide margin keywords are typeset. These are rotated 90 degrees and printed in blue, which

adds a very nice touch to the book's typographic feel. Especially so where the keyword in question is actually a font name, because each of those is typeset in the font that is indicated.

Afterword

At the Dante Christmas Party 2006 we showed Hermann some of his work on a digital ink device and he seemed quite impressed with what new technologies can provide. However we fully agree with the following quote from his monograph:

“Ein gedruckter Buchstabe und ein schön gestaltetes Buch sind etwas Beständiges, Bleibendes im Vergleich zu dem schnellen Zugriff zu einer Information im Internet und dessen Flüchtigkeit der Wiedergabe am Bildschirm. Es ist das etwas schwer zu beschreibende eigenartige Erlebnis des Lesers, wenn er ein Buch in seiner Händen hält. Ein Buch spricht die Sinne an, der Druck auf dem Papier, das Umblättern der Seiten, ganz im Gegensatz zu der abstrakten elektronischen Darstellung eines Textes.”



Both language versions of the monograph can be ordered directly from the Merchandise section of the Linotype website, <http://www.linotype.com/26/merchandise.html>.

If you prefer to order elsewhere, The ISBN number is 3-9810319-5-4 for 'Alphabetgeschichten', or 3-98103129-6-2 for 'Alphabet Stories'.

Hans Hagen & Taco Hoekwater

Folding Sheets for a Modular Origami Dodecalendar

Richard Hirsch
richard.hirsch at gmx dot net

Abstract

Twelve square sheets of paper can be folded in such a way that they can be assembled to a pentagon dodecahedron (*origami*). The single units are called modules, hence the name *modular*. If the sheets bear calendrical information at the right places, the dodecahedron shows the calendar for each month on its faces: the *dodecalendar*.

In this article we let MetaPost calculate piece by piece the information that needs to be printed on the module paper to enable us to fold the modules and assemble the dodecahedron.

Keywords

MetaPost, tutorial

Introduction

MetaPost

MetaPost is a programming language and an interpreter that produces PostScript output. It is well suited to produce technical drawings and – being derived from Metafont with basically the same capabilities and just a few extensions – suggests itself to illustrate \TeX documents. As we will see, MetaPost provides a very natural way of describing relations of points in the plane (3D extensions do exist also).

However, the richness of features and concepts makes it hard for the beginner to start. The user's manual from John Hobby [1] and the Metafontbook from Donald Knuth [2] want to be read and for the fine touch even a glimpse into the source code (`plain.mp`) may be necessary.

Examples, however, are a good way to start with. There are excellent sources in the Internet (c.f. <http://www.tug.org/metapost.html> for an extensive list), and I hope, the origami dodecalendar will be another one that might attract you to MetaPost.

Dodecalendar

At <http://www.origami-cdo.it/modelli/> instructions for folding a 12-piece modular origami dodecalendar can be found. Twelve square paper sheets are folded into twelve modules that can be assembled to a pentagon dodecahedron. If the paper is properly printed, each face shows a calendar for one month, see figure 1.

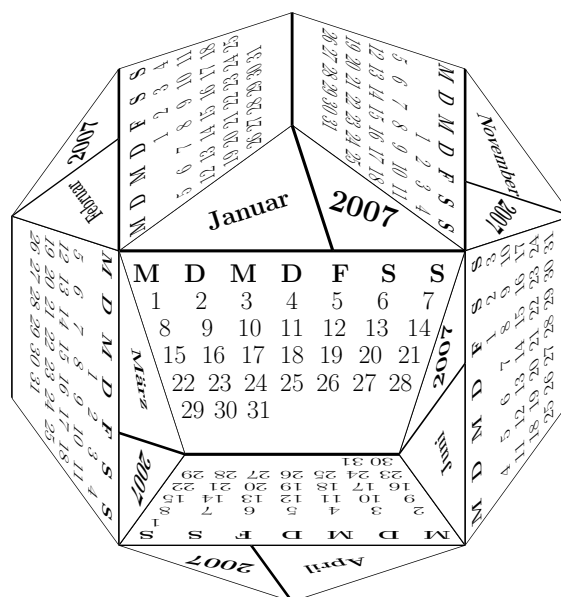


Figure 1: the modular origami dodecalendar

Aim of this Article

We want to print those sheets on our own and use MetaPost for this purpose. Traditional origami doesn't allow preprinted guides that show where the creases must go, but since we have to print the calendars anyway, we can as well print those marks – not for cheating, of course, but to avoid ugly creases on the faces of our dodecalendar.

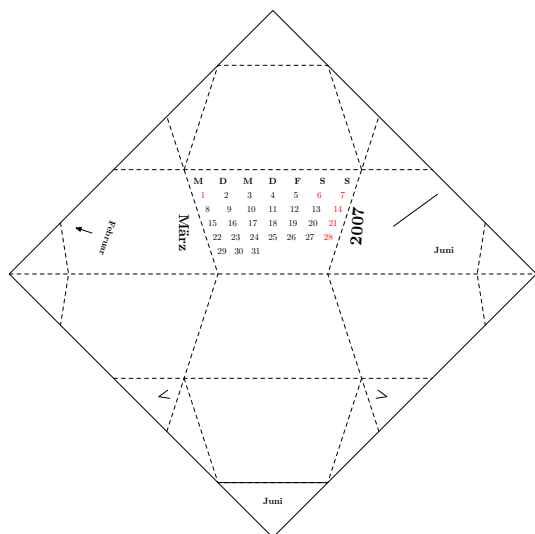


Figure 2: folding sheet with creases shown for January 2007 (sic!, see figure 1)

Folding Sheets

Figure 2 shows how the preprinted sheets of paper should look like. We number the relevant points as shown in figure 3.

A good start for calculating all those points seems to be the base of the regular pentagon highlighted in figure 3. But first of all we have to perform some setup.

Setup

We are going to calculate the positions of the points shown in figure 3. Usually this is done in Metafont or MetaPost by collecting the x - and y -parts of the coordinate pairs in the arrays x and y and address them as points z . We will use coordinate pairs P_1 through P_{20} instead and won't bother with their components at all. This may be mainly a matter of taste, but there is at least one rationale too: For the dodecalendar we need twelve paper sheets for the modules, and for each of them the positions of the points in figure 3 must be known. Now MetaPost clears the contents of the coordinate pairs z for every new figure; thus all the calculations would have to be done over and over again. The positions in container P however, are valid through the whole MetaPost run. Therefore, first thing for us to do is to declare the container P for the coordinate pairs:

```
pair P[];      % positions of the points
```

(As you already may have guessed or known, just like in \TeX everything after a $\%$ -sign in the MetaPost source is ignored by the interpreter.)

Furthermore we have to give MetaPost some information about the size of our figure. Therefore we de-

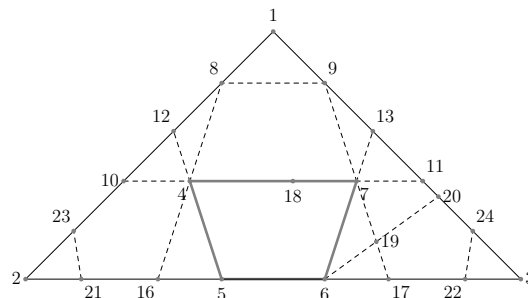
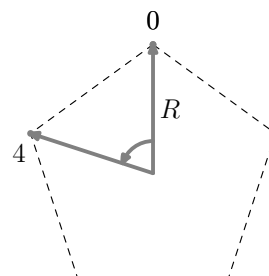
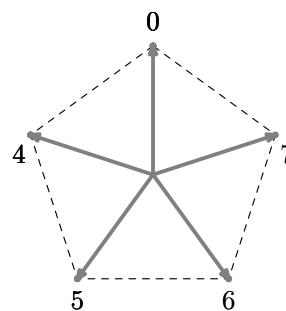


Figure 3: numbering of important points



(a) turning P_0 by 72°



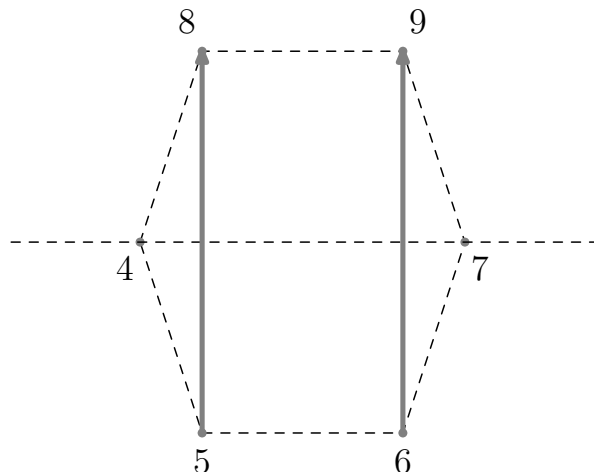
(b) all vertices of the pentagon

Figure 4: central pentagon

fine a basic unit of measurement for our drawing: the radius R of the escribed circle of the central pentagon (see figure 4) is set to be 2 cm:

```
R:= 2cm;      % scaling factor
```

The real MetaPost way to deal with the absolute size of the figure would be to tell MetaPost only something like $P_2 + 11\text{cm}*\text{right} = P_3$ and let it figure out the actual value of R itself. Unfortunately MetaPost can not perform transformations unless all components of the transformation matrix are known. Since we want to make heavy use of transformations, it is better for us to fix the scale of the figure at this time.

Figure 5: points P_8 and P_9

Regular Pentagon

First of all, we introduce point P_0 at the top of the pentagon to complete it. We set the origin of our coordinate system to the center of the pentagon and then we can define the position of P_0 simply as

```
P0 = R*up;
```

The vector `up` is predefined in MetaPost as “`up=-down=(0,1)`”.

Now, by means of MetaPost’s rotation command, the position of the other vertices can be calculated as easily as

```
for i:= 1 upto 4:
  P[i+3]:= P0 rotated (i*360/5);
endfor
```

Mirroring of the Pentagon’s Base

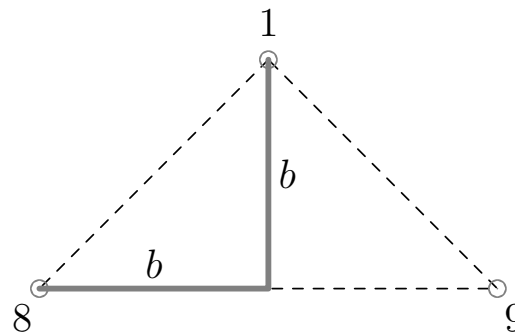
Now that MetaPost knows the positions of P_4 to P_7 , we can define points P_8 and P_9 : We get P_8 if we reflect P_5 about line $\overline{P_4P_7}$ and P_9 by doing the same with P_6 . MetaPost understands this immediately:

```
for i:= 8,9:
  P[i]:= P[i-3] reflectedabout (P4, P7);
endfor
```

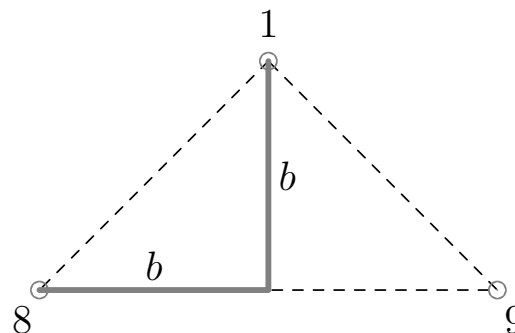
The Top Corner

The points P_8 and P_9 are of special interest since they lie on the edges of our folding sheet; until now all our drawing was completely independent of the actual size of the paper. But now we are going to deal with the boundaries of our sheet of paper; we start with the top (P_1).

We observe that the triangle $\triangle P_8P_9P_1$ is the top half of a square (see figure 2). We know the length of its



(a) Calculation by vector addition



(b) Calculation by means of the law of Pythagoras

Figure 6: P_1 , the top corner of the paper sheet

base since we have already P_8 and P_9 ; we call it a and let $b = a/2$ (see figure 6a).

```
a:= length(P9-P8);
b:= .5a;
```

Now MetaPost can find P_1 simply by vector addition

```
P1:= P8 + b*(right + up);
```

(Alternatively we could have made use of the Pythagorean theorem (and introduce MetaPost’s `dir` command and its `++`-operator): We know that the angle $\angle P_1P_8P_9$ is half of a right angle (i.e. 45°). The length of the hypotenuse c is $\sqrt{b^2 + b^2}$ (see figure 6b) and this root is just what the `++`-operator calculates. So shifting P_8 by $b++b$ in northeast direction yields P_1 – or in MetaPost’s terms: $P1:= P8 \text{ shifted } ((b++b)*dir\ 45).$)

Left Edge

Next we want to deal with the points on the left edge, P_2 , P_{10} and P_{12} . Point P_2 lies somewhere on the line through points P_5 and P_6 (see figure 7) and also somewhere on the line going through P_1 and P_8 . If we

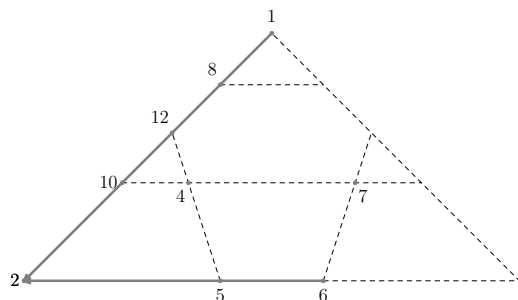
(a) Point P_2 as intersection point

Figure 7: points on the left and right edge

want to tell MetaPost about these facts we can use the volatile numeric variable *whatever* for “somewhere” and the expression $[p,q]$ for “on the line \overline{pq} ”. So we write:

```
P2 = whatever[P1,P8] = whatever[P5,P6];
```

This is all information MetaPost needs in order to know where point P_2 lies.

Please note, that we didn’t use the $:=$ assignment, but instead add two equations to MetaPost’s internal system of linear equations with $=$. This system is solved if we use P_2 in a drawing (or an immediate assignment with $:=$).

Points P_{10} and P_{12} are computed likewise:

```
P10 = whatever[P1,P8] = whatever[P7,P4];
P12 = whatever[P1,P8] = whatever[P5,P4];
```

Right edge

The positions of points P_3 , P_{11} and P_{13} could be found the same way, but we don’t want the MAPS to become dull.

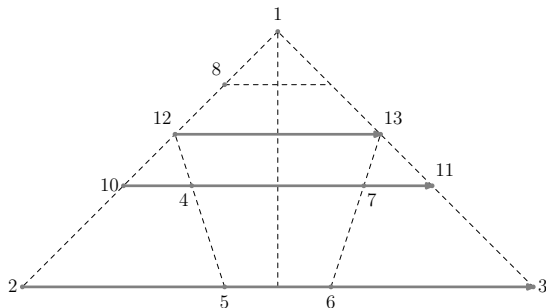


Figure 8: The points on the right edge

Instead we use another important feature of MetaPost: transformations. In figure 8 we note that point P_3 is just P_2 reflected about line $\overline{P_0P_1}$. In MetaPost

we can express that as $P_3 = P_2$ reflected about (P_0,P_1) but since we need to transform more than one variable the same way, we’d rather have an own transformation for this purpose. We can have that in MetaPost by defining

```
transform flipright;
flipright:= identity reflectedabout(P0,P1);
```

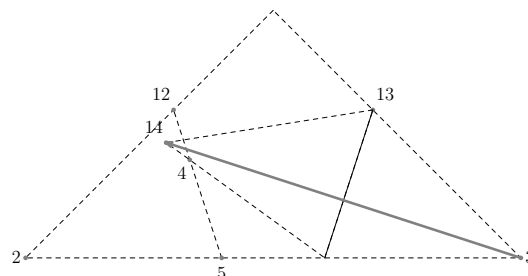
Now we could say $P_3 = P_2$ transformed flipright , but we go one step further and define the new macro flippedright :

```
def flippedright=transformed flipright enddef;
```

That enables us to write simply:

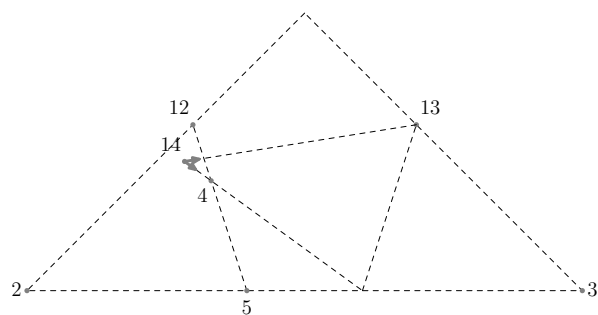
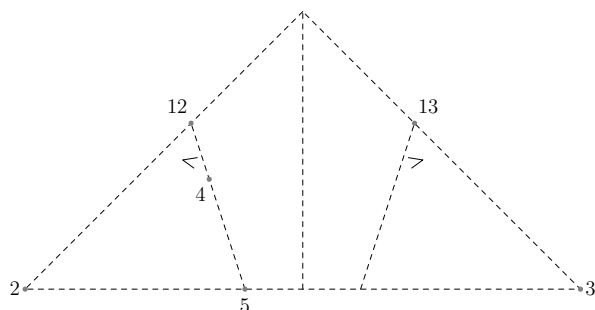
```
P3 = P2 flippedright;
P11 = P10 flippedright;
P13 = P12 flippedright;
```

Angle Marks

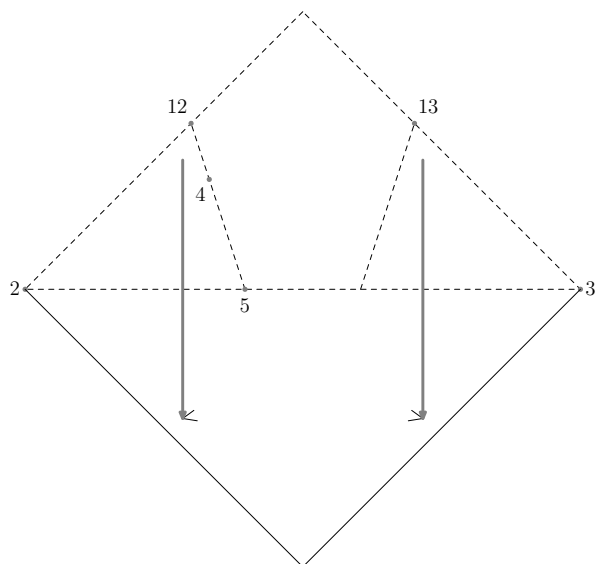
Figure 9: Points P_{14} and P_{15}

In the process of building a module, the corners P_2 and P_3 must be folded along $\overline{P_5P_{12}}$ and $\overline{P_6P_{13}}$ respectively. We must get the creases by having the angle marks at P_{14} and P_{15} preprinted on the sheets (see figure 9); this way P_2 can be folded to P_{15} and P_3 to P_{14} and the creases come up. Since we know already about transformations, defining P_{14} and P_{15} is too easy:

```
P14 = P3 reflectedabout (P6, P13);
P15 = P14 flippedright;
```

(a) the command `unitvector`

(b) the guides for the corners



(c) the guides on the lower half of the paper

Figure 10: Angles as guides for the corners

We don't want just points, however, but nice little angles of a certain length (`ticklength` say), into which the corners fit snugly (see figure 10). For this purpose we use MetaPost's `unitvector` command which reduces a given vector to length 1, but maintains its direction. With this tool we can build vectors with arbitrary length in a given direction, just by multiplying it with the proper dimension (see figure 10a). For the calculation of the endpoints of the angle's rays we define a *vardef* macro, which returns its last expression (like a function in certain other programming languages):

```
ticklength:= 0.1a;
vardef tkend(expr p, q) =
  p + ticklength*unitvector(q-p)
enddef;
```

The two rays of each angle are obtained by connecting P_{14} and the endpoints with the `--`-operator. It produces a path where the points are connected by straight lines. We can store these paths in suitable variables (of type `path`) and collect them in a container (`Line[]`). For the angle at point P_{15} we make use of the fact that transformations work on paths as well as on points:

```
path Line[];
Line14:= tkend(P14,P6)--P14--tkend(P14,P13);
Line15:= Line14 flippedright;
```

Finally we want the guidelines in the lower half of the paper. So we apply a final transformation. (Here the `:=` operator is mandatory, because the equation with `=` would be inconsistent.)

```
transform flipdown;
flipdown:= identity reflectedabout (P2,P3);
def flippeddown = transformed flipdown enddef;

for p:= 14,15:
  Line[p]:= Line[p] flippeddown;
endfor
```

Boundaries for Calendrical Information

In order to print the calendar, the month and year information at the correct places, we need to calculate the positions of points P_{16} to P_{19} (see figure 3).

Since we obtained P_8 by reflecting P_5 about $\overline{P_4P_7}$ (see 'Mirroring of the Pentagon's Base'), we know from the interception theorems that the vector $P_8 - P_4$ is the same as $P_4 - P_{16}$ (see figure 12a). The position of P_{17} is again found by reflection (figure 11b).

```
P16:= 2[P8,P4];
P17:= P16 flippedright;
```

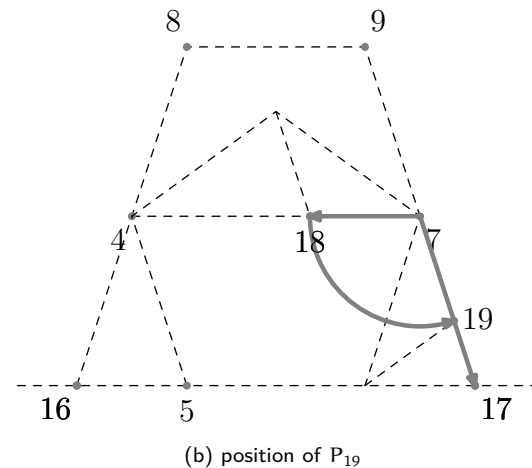
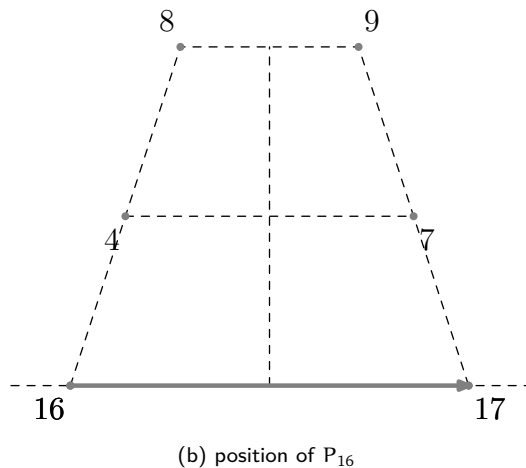
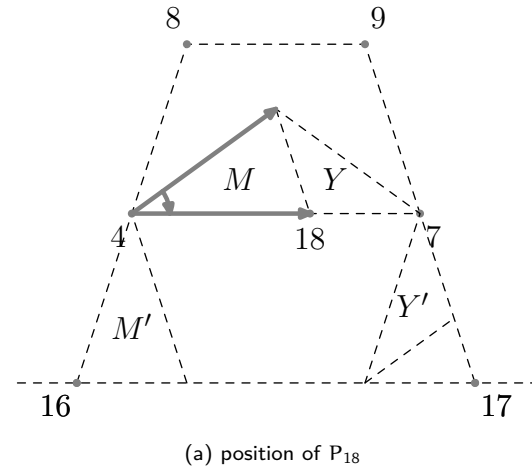
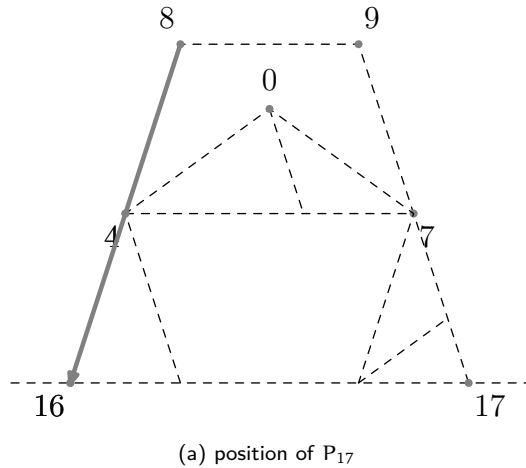


Figure 11: boundaries for calendrical information

Figure 12: boundary of the year-field

In order to determine the position of P_{19} we look at P_{18} first: The triangle M is isosceles, so $|P_0 - P_4| = |P_{18} - P_4|$ and we can obtain the position of P_{18} by rotating P_0 around P_4 :

```
w:= angle(P0-P4) - angle(P7-P4);
P18 = P0 rotatedaround (P4, -w);
```

(Yes, the expression $\text{angle}(P_7-P_4)$ is indeed 0 and could have been left out, and yes, the use of `unitvector` may have been appropriate here too; indeed, we are going to return to using `unitvector` soon when calculating P_{19} .)

Now, that we know the position of P_{18} we can focus again on P_{19} . The line P_0P_{18} divides the triangle $\triangle P_4P_7P_0$ into two smaller triangles M and Y that contain the information about the month and the year respectively (see figure 12a). We observe that the triangles Y and Y' are identical, only that the latter

falls onto another face of the dodecahedron. Thus $|P_7 - P_{18}| = |P_7 - P_{19}|$ and we get the position of P_{19} by applying the universal example of constructing a vector by multiplying its length with the unitvector of its direction:

```
P19 = P7 + unitvector(P17-P7)*length(P7-P18);
```

Guideline for aligning two modules

When all the modules have been folded, every two of them at a time are to be combined to a double module. To get the correct angle between the faces of the dodecahedron, the two modules must be aligned along line $\overline{P_6P_{20}}$. (As it happens, this is just the bisector of angle $\angle P_6P_3P_7$, but we pretend not to know about that.) Since we know already about the position of P_{19} , we can apply the same technique as in 'Left Edge' and have

```
P20 = whatever[P1,P3] = whatever[P6,P19];
```

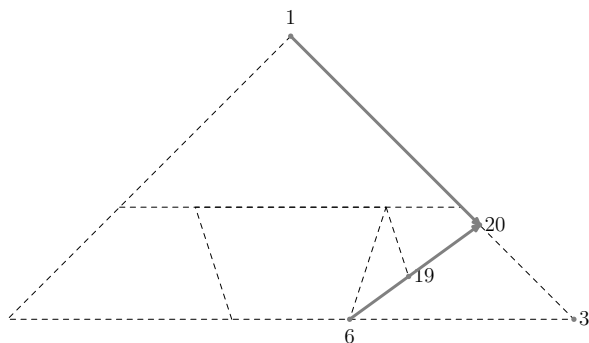


Figure 13: The guideline for aligning two modules

On the final sheets we don't want the whole line to show up; some part (20 % say) at the beginning and the end should be left out (see figure 2). We find the new starting point 20 % from P_6 on the line $\overline{P_6P_{20}}$. Sounds familiar? Yes, see section 'Mirroring of the Pentagon's Base' – we can apply the $[p,q]$ -expression again and store the resulting path in our `Line[]` container.

```
Line[20] := .2[P6,P20] -- .8[P5,P20];
```

Drawing the folding sheets

Now MetaPost knows about all the relevant points and we can let it draw the folding sheets. The result, together with some labels, is presented in figure 14. Alas, the calendar is still missing (see below).

```
beginfig(1);
  draw P1--P2--P1 flippeddown--P3--cycle;
  draw P8--P9;
  draw Line[20];
  for l:= 14, 15: draw Line[l]; endfor
endfig;
end.
```

Summary and Outlook

For now we have learned how to make use of some important features of MetaPost, among them

- ☐ addition of vectors,
- ☐ affine transformations like *shifting*, *rotation*, and *reflection*,
- ☐ solving linear equations (the *whatever*-statement),
- ☐ and last but not least how to draw *straight lines*.

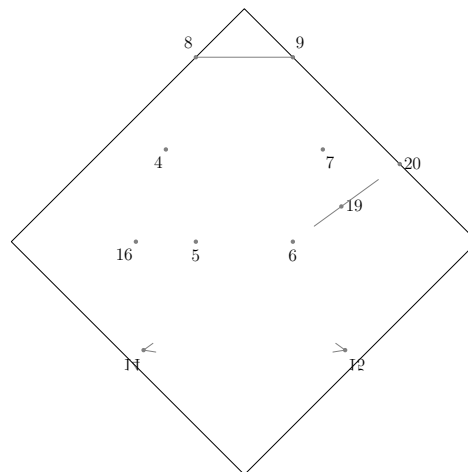


Figure 14: the printed information (with labels)

In the next issue of the MAPS we will have a look on MetaPost's algebraic capabilities and the possibilities to add labels and \TeX content to the picture when we are going to make MetaPost calculate and draw the calendrical information for the dodecahedron too.

In the meantime you can prepare a set of twelve folding sheets and assemble a dodecahedron as shown at the above mentioned web page. (If you don't know how to run MetaPost on your system, you can try the MetaPost Previewer at <http://www.tlhiv.org/MetaPostPreviewer>, just put the code without "beginfig(1);", "endfig;" and "end." into the textfield and press the preview button.)

Have fun!

References

- [1] John D. Hobby. A User's Manual for MetaPost. Technical Report 162, AT&T Bell Laboratories, Murray Hill, New Jersey, April 1992. Also available at <http://www.tug.org/docs/metapost/mpman.pdf>.
- [2] D. E. Knuth. *Computers and Typesetting*, volume C. Addison Wesley, Reading, Massachusetts, 1986.

Richard Hirsch
richard.hirsch at gmx dot net

ConT_EXt User Meeting 2007

Epen, March 23–25

Mojca Miklavec

The 23rd of March was a drippy Friday afternoon, but the rain did not stop T_EX-ies from using almost all modes of transport to gather in the small village of Epen on the Dutch–Belgian–German border. Most of the 26 attendees from 11 countries, arrived by plane, train, bus, or car, but a few preferred to roller blade, to buy a new bike (because the old one broke down on the way to the meeting), or to jump over the fences all the way from Aachen.

The extremely interesting and packed schedule meant evening discussions that often continued into the wee hours of the morning. And even with the delicious food, Taco usually had problems interrupting the lively discussions and wild coding to bring us down to the dining room before the food got cold.

Indeed, we hardly had time to breathe during the meeting. Those who were not in Epen should feel sorry for missing out on the fun!



Tutorials

Friday evening started with an excellent tutorial by Taco on how to write a ConT_EXt module. He led us from the basics of writing a module to the most important aspects, conventions, tips, and tricks. The tutorial included dozens of pages of documentation to be published on the wiki. Our assignment was to write a new FIXME module, which has been on the ConT_EXt wishlist for more than a year. Although no module had been written by the end of the meeting (perhaps because there were too many interesting talks), the tutorial and documentation should inspire more authors to provide new high-quality ConT_EXt modules.

Two more tutorials followed on Saturday.

Taco Hoekwater:
Writing a ConT_EXt module



Willi's tutorial was not just about setting up a page layout in ConT_EXt, but also dealt with typographical traditions concerning printing, and the related technical aspects like the properties of the paper sheets your book will be printed on.

In the picture you can see Sanjoy Mahajan solving some of the related questions: how many times do I need to fold this sheet to get 16 pages? And what is the grain direction of the paper?

Willi Egger:
Page layout, Arrangements and Posters



Willi in action

Hans Hagen: In contrast to math where many users can start from their LaTeX experience when switching to ConTeXt, XML processing is ConTeXt's speciality without beginner's manual. That meant that every single hand was raised when Hans asked about the interest to listen to his XML tutorial, although it was almost bed-time when it started.

XML

History of Typesetting & Typesetting of History

Taco Hoekwater: The first talk on Saturday morning was given by Taco Hoekwater, one of the first ConTeXt users outside PRAGMA ADE. It was enjoyable to listen to the summary of the early revolutionary ideas Hans implemented into his system at a time when some of the attendants in Epen were still school kids who had only dreamt about owning a computer.

A short history of ConTeXt

Thomas A. Schmitz: Coming from a completely different field as most attendees (humanities), Thomas shared his experience of ConTeXt related to typesetting Ancient Greek, struggling with font-system oddities and limitations of PDFTeX. Is LuaTeX going to offer the definitive answer to the problems he had to fight with?

Classical greek with ConTeXt

Idris Samawi Hamid: In contrast to the usual development cycles of ConTeXt and its modules, where documentation is lagging way behind the functionality, Idris brought with him a complete specification of a yet-to-be-written module for typesetting Critical Editions with multiple levels of footnotes, proposing a complete hierarchy of commentaries.

Critical Editions

Talks From User Experience . . .

Mari Voipio: Mari presented all the problems a newbie faces when switching from MICROSOFT products to ConTeXt: no WYSIWYG editor, no drop-down menu with fonts, no copy-paste to include images, no easy way to create tables . . . Would a WORD2CONTEXT tutorial help?

ConTeXting in MS WINDOWS— a user's view

Sanjoy Mahajan: Sanjoy described how he uses ConTeXt to typeset his mathematics textbook (*Street-fighting mathematics*), covering project structure, page layout, and figure-text integration (figure placement).

Typesetting a physics textbook with ConTeXt



The conference room

Duncan's company uses ConT_EXt for typesetting multilingual documents (including Arabic) from XML sources, but he showed a great deal of courage when he dared to make a presentation in PowerPoint during ConT_EXt meeting, for which he has been "punished" appropriately during the live demo.

Duncan Hothersall:
Using ConT_EXt as part of a larger system

Useful tools

Patrick's presentation of his TEXTMATE extensions for better ConT_EXt support could be called "a story of success" or "a good reason why one should attend T_EX meetings". His first announcement about the ConT_EXt bundle on the mailing list got no reply. This time, most of the Mac users at the meeting eagerly awaited the upload of the new bundle, in order to try it on their own computers. The bundle supports syntax highlighting and auto-completion of all user-level commands. It also provides an interactive list of arguments that these commands accept, and provides shortcuts for typesetting and previewing.

Patrick Gundlach:
ConT_EXt integration in the TEXTMATE editor

Probably tired of the ease of typesetting PDF documents with ConT_EXt, he has also written support for manual editing of PDF in TEXTMATE, so that cross-reference tables can be calculated automatically. Interested hackers should ask him for details.

Patrick Gundlach:
Creating a PDF document, the hard way

Discussions

All attendees agreed that our shelves lacked a well-written "THE CONT_EXtBOOK" or (online) Cookbook, but no one volunteered to write it. It was also agreed that T_EXSHOW is incomplete. Filling the gaps in this area is mostly up to the users.

Mojca Miklavc:
Documentation

Some suggested to clean up and improve the wiki pages: to create a site index, to point the most important recent changes and advances since the last update of ConT_EXt manual. More samples should be provided, and submitting test files should be made easier.

Sanjoy Mahajan:
Regression testing

The one who makes few mistakes makes little progress. Because ConT_EXt development is progressing apace, a repository with test suites has been set up, and Sanjoy has been developing tools to check for broken functionality between different ConT_EXt releases. Simplifying submissions of test cases should be one of the first steps towards a better quality control before new “ γ releases”, as some jokingly called them.

A Glimpse into the Future

Arthur Reutenauer:
**An Introduction
to OPENTYPE fonts**

Unfortunately, OpenType fonts are not as open as one would expect them to be, but luckily enough both Hans and Patrick have been open enough to ideas and requirements for proper font support in ConT_EXt and on the garden, some of which have secretly been added to live.contextgarden.net during the night before presentation, while Patrick himself was sleeping.

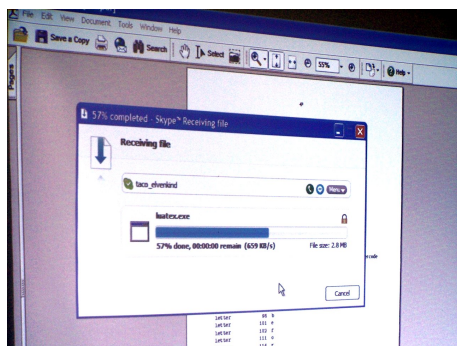
Idris Samawi Hamid:
Oriental T_EX

If typesetting Ancient Greek is problematic – what about Arabic? After (mis)using ALEPH for quite some time, Idris claimed that no existing system was able to meet his requirements to typeset old Arabic texts, and applied for a grant at Colorado State University to fund Taco’s development of LuaT_EX. Without such a grant, we would probably not be seeing the rapid development of LuaT_EX, aiming for the first beta release during TUG conference this year.

We are now eagerly awaiting his next tutorial about writing funding proposals (hopefully applicable to EURO currency as well).



*Hans Hagen,
Taco Hoekwater:*
**LuaT_EX,
ConT_EXt MKIV,
Fonts & Typescripts**



Last but not least, Hans and Taco filled in the remaining time not already reserved for other presentations. Although I’ve seen quite some of their talks already, they always keep us surprising with some really special slides – both in visual appearance and contents. The development has been progressing almost with the light-speed – even during the conference. The picture shows a (non-scheduled) Live demonstration of bugfixing in LuaT_EX during Hans’s tutorial.



The most special thing of the meeting was that most of us have known each other from vivid discussions on the mailing list without ever meeting before. So the introductory discussions that would otherwise start with platitudinous phrases asking for names, location or profession, were usually replaced by hitting the core of the subject:

“Wait! You are the one who did . . . !”

The meeting was a great success and left a deep impression to everyone. Most attendees left Epen saying “See you next year”.

How to fit five nationalities into one car? Well, T_EXies have lots of experience with packing boxes. So as long as they go well with each other, they always find a solution.

Invitation to Slovenia

The second ConT_EXt user meeting, in 2008, will take place in Slovenia, near the Slovenian–Austrian–Italian border, and last probably a day or two longer than this one, in order to leave you more time to discuss the fascinating topics, to exchange ideas, and to breathe fresh air while sightseeing or doing sport. As the date approaches and planning crystallizes, the time and location will be announced on the mailing list and contextgarden.net.



Mojca Miklavc



Participants:

Front row (from left to right): Luigi Scarso (it), Mojca Miklavec (si), Zofia Walczak (pl), Taco Hoekwater (nl), Duncan Hothersall (uk), Arthur Reutenauer (fr)
 Back row: Bernd Militzer (de), Willi Egger (nl), Thomas Engel (de), David Roderick (uk), Oliver Buerschaper (de), Patrick Gundlach (de), Mari Voipio (fi), Steffen Wolfrum (de), Luuk Beurskens (nl), Alexander S. Berdnikov (ru), Jano Kula (cz), Idris Samawi Hamid (us), Hans Hagen (nl), David Kastrup (de), Michael Guravage (nl), Karel Horák (cz)

Missing from the photo: Sanjoy Mahajan (us), Tobias Burnus (de), Thomas A. Schmitz (de), Jelle Huisman (nl)

EuroBachoT_EX 2007

Michael Guravage

Saturday

Jerzy Ludwichowski, GUST president and conference organizing committee chair, opened the conference by welcoming all the participants, and encouraged everyone to enjoy the proceedings in the spirit of the conference – “Paths to the Future”.



The first speaker was **Jonathan Kew**, who related the history and current status of X_YT_EX. After its initial appearance in the spring of 2004 on Mac OSX, a version supporting OpenType fonts appeared the following year. X_YT_EX for Linux was announced at BachoT_EX 2006, and was quickly followed in June by a version for Windows. This year marked a milestone for the X_YT_EX project, in that it became an integral part of T_EX Live distribution.

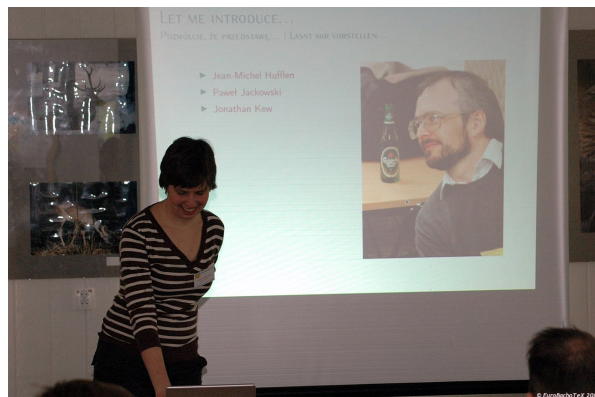
Key features of the current X_YT_EX implementation include its native Unicode support, improved integration with existing macro packages and its smart inclusion of hyphenation patterns.

Looking ahead, future releases will support host operating system fonts (OpenType, TrueType and PostScript) with no T_EX-specific setup. Another new feature is inter-character tokens – inserting arbitrary tokens in-between adjacent characters based on character classes. This allows one to easily mix scripts and fonts, or insert spacing to stretch text. Finally, to better support non-Latin scripts minority languages, and scripts not yet in Unicode, X_YT_EX will support SIL’s Graphite font system.

Taco Hoekwater began his presentation by announcing that Metapost version 1.0 is now available. New features include, file name templates, new color types, i.e. cmyk, grey-scale and marking-only and an improved manual. To overcome the various size limitations of the current implementation, Metapost version 1.1 will incorporate dynamic array allocation and provide greater numeric precision. To obviate problems with existing input files, this new version will appear as a separate binary. Finally, in the next year or so Taco anticipates making Metapost functionality available as a library.

The title of **Hans Hagen’s** presentation was “Beware of too much tokenspeak”. T_EX consumes characters which, in turn, become tokens, and then nodes. Hans gave us a glimpse into how LuaT_EX, at the node list level, is simplifying and streamlining previously complex pieces of T_EX. So much so that he has been able to retire moderate pieces of existing ConT_EXt code. Consistent with the theme of the conference, Hans described how LuaT_EX provides a genuine opportunity to embrace the future.

Joanna Ludmiła Ryćko introduced the T_EX Clinic. The clinic began last year at BachoT_EX, and was open to anyone at the conference seeking relief from a nagging T_EX complaint. A number of T_EX clinicians were introduced and put at the disposal of the participants.





Johannes Große presented MathPSfrag – a tool that replaces existing labels in Encapsulated PostScript graphics with LaTeX generated labels. MathPSfrag extends PSfrag, allowing both automatic and fine grained manual control over label content and placement.

Siep Kroonenberg presented her epspdf conversion utility. Written in Ruby and Ruby/Tk, and using Ghostscript and pdftops, epspdf offers both command line and graphical user interfaces for a round-trip conversion between PostScript and PDF.

Zofia Walczak demonstrated several basic and advanced features of the the Portable Graphics Format (PGF) package. Written by Till Tantau at the Institute for Theoretical Computer Science at the University of Lübeck, PGF is partitioned in three layers: system, basic and front-end. TikZ is a front-end for PGF. It provides access to all the features of PGF, and is intended to be easy to use. If you look closely you will see it has borrowed part of its syntax from both METAFONT and PSTricks.

Norbert Preining stood in for **Jim Hefferon** and described a new ‘experimental’ procedure for uploading software to CTAN. The workflow includes upload, approve and install steps resulting in TDS compliant

bundles. A means for updating package meta-data is also present.

Jean-Michel Hufflen introduced us to XSL-FO, comparing and contrasting corresponding LaTeX and XSL-FO structures.

Grzegorz Murzynowski introduced gmdoc, a package for documenting LaTeX style files. It differs from its predecessor by emphasising compact ‘minimal’ markup.

Grzegorz Murzynowski continued by describing his gmverse and gmcontinuo packages. The former provides right alignment for long and broken lines of verse. The latter allows typesetting paragraphs *in continuo*, marked not with a new line and indent but continuously, marked with only the ¶ sign.

In the last talk for Saturday, **Marek Ryćko** argued for a fine, or finer, grained component architecture for TeX functionality. He hopes that focusing on interfaces to facilitate integration will be the tipping point for TeX development.

The weather was clear and cool throughout the week. So it was under the stars and a waxing moon that, later



that evening, we enjoyed the annual bonfire; replete with food, drinks, songs, and of course fire breathing pyroTeXnics.

Sunday

This year we found the accomodation not quite ready for guests. For instance, there were no curtains and toiletpaper was missing as well. It took us a while to find out that all cloth and paper was being used in the “make yourself some paper” workshop given by **Grażyna Jackowska** that ran in parallel to the talks. As the conferences advanced, the participants had to become more careful where they walked because handmade paper was hanging on trees everywhere.

Andrzej Tomaszewski began the second full day of talks by describing the various conditions and limitations he encountered while producing “The Master of Life Arteries of the Greater Warsaw,” a jubilee book for the Warsaw Municipal Water Authority.

Dorota Cendrowska presented several, oft disregarded, design criteria to consider when typesetting enumeration for inclusion in printed text and multimedia presentations.

Jerzy Ludwichowski described his and **Karl Berry**’s work on consolidating the GUST SOURCE and NON-SOURCE font licences into the single GUST Font License (GFL). The result is a license that is legally identical to the LaTeX Project Public License (LPPL), which the FSF and Debian already accept as a legitimate free software license.

Jean-Michel Hufflen described how MIBibTeX strives to be a better BibTeX. Starting in 2000, MIBibTeX originally was written in C, but has been reimplemented recently in Scheme, a Lisp dialect. Jean-Michel anticipates MIBibTeX’s first public release in May of this year.

Next, **Jean-Michel Hufflen** showed how lexicographical order relations are language-dependent, and how MIBibTeX addresses this issue in the context of multilingual bibliographies. Bibliography styles can be unsorted or sorted. However, the bst language’s sort function is suitable for English only. MIBibTeX uses nbst and scheme which together allows one to sort European Languages in correct lexicographic order.

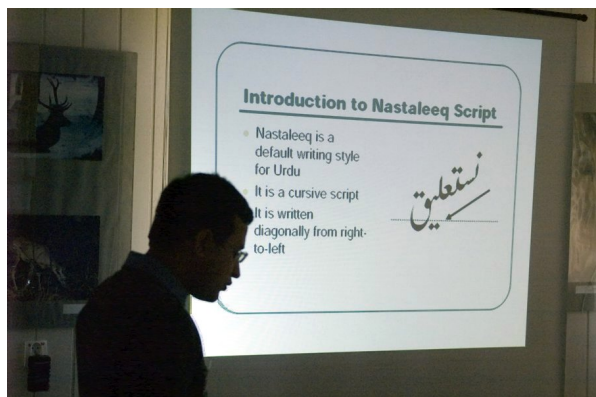
David Kastrup described how to download, install and use the Emacs AUCTeX package. You can retrieve the latest version of AUCTeX from <http://www.gnu.org/software/auctex>. And for the stouthearted, the source code for a pre-release version of Emacs 22 is available from <http://alpha.gnu.org/gnu/emacs/pretest>

Péter Szabó demonstrated dvdmenuauthor, a collection of tools, including pdfLaTeX and xpdf, used to create menus for dvdauthor – an excellent low level tool for creating video DVDs on Unix systems.

Norbert Preining described how the Debian “etch” release contains both TeX Live 2005 and teTeX – in parallel, and how both system administrators and regular users can benefit from side-by-side TeX distributions. Norbert concluded with a preview of TeX Live 2007 and further developments regarding TeX on Debian.



The presentation of **Atif Gulzar** and **Shafiq-ur Rahman**, who are from Pakistan, began by explaining how Urdu is used by some sixty million people in twenty countries. Urdu is based on an Arabic script with Nastaleeq as its most prevalent writing style. Nastaleeq is highly contextual – written right-to-left and top-to-bottom. Atif constructed an Omega virtual font containing 827 glyphs, and used Omega external OTPs in a two pass solution to achieve the appropriate ligature placement and kerning. From the more than twenty thousand valid ligatures in Urdu, Atif was able



to correctly render and place a subset of approximately seven thousand ligatures.

Hossam A. H. Fahmy presented his joint paper with **Amir M.S. Hamdy** about their aim to create a font suitable for typesetting the Qur'an. Using examples from existing fonts, he explained many of the problems that one encounters when attempting to digitize a calligraphic script like Arabic. The second part of the talk focused on a detail of that: how to simulate a real-world pen nib in METAFONT.

All the news about pdf \TeX version 1.40 was brought to use by **Martin Schröder**. Most prominent among the new features are the ability to create compressed object streams, support for JBIG2-encoded images, and the addition of a colorstack à la dvips. The colorstack feature is already in use in the new releases of the hyperref package, and solves the LaTeX problem of the text color disappearing at a page break.

Karel Horák walked us through the history of the háček – or caron, if you prefer – in Czech typesetting. He showed us not only an objective historical progression of the symbol shape, but also many forms that occur in actual fonts. Some few he considered good, some more not so good, most all are apparently simply hideous and out of touch with Czech tradition. The likely cause is that the big font foundries never considered asking a Czech typographer for an opinion.

Hàn Thế Thành also talked mostly about accents, but in this case about the ones used in Vietnamese. The writing system is based on the Latin alphabet, but it has great many accented characters to denote sounds that are not differentiated in the roman alphabet. His Vn \TeX package is a complete solution for typesetting Vietnamese, including support for large number of fonts, some of which he created himself.



The day ended with two presentations by **Tomasz Łuczak**. The first talk was about the LyX document processor (see www.lyx.org), the second talk about how to convert wiki markup into \TeX source. Unfortunately, both talks were given in Polish, and even with the simultaneous English translations provided by kind members of the audience it was hard to follow.

Monday

There were no lectures scheduled for Monday. Instead, we took an excursion to Toruń where we visited the District Public Library – Copernican Library and toured the town. After which we drove on to Chełmno where we enjoyed a scrumptious dinner and music before returning home.

Toruń, situated astride the Vistula (Wisła) river, has been an important regional and trading center since medieval time. A member of the Hanseatic League, Toruń boasted a fleet of one hundred and fifty ships, whose trade allowed Toruń's prosperity to rival that of Brugge, Copenhagen and London. UNESCO designated the Gothic buildings of Toruń's Old Town a World Heritage Site in 1997.

At the Copernican Library we were treated to a sample of the treasures of their collection, including a first edition of Copernicus's "Revolutionibus Orbium Coelestium", or "The Revolution Of The Heavenly Orbs" which appeared in print in 1543. Lastly, we were shown a recent reproduction of Gutenberg's Bible. The exemplar is one of 180 copies, matching Gutenberg's original number. Each exemplar was made using the same materials and techniques as the originals, including individual letter variations (font expansion) that Gutenberg used to achieve aesthetic interline spacing.



Our tour of Toruń's Old Town began at the historic Town Hall under a statue of Nicholas Copernicus with the inscription, "He moved the earth, and made the sun stand still." We visited several churches and historical landmarks before ending where we started.

We were running late, so it was late in the afternoon when we arrived in Chełmno, a town located on seven hills, and one of Europe's best examples of defensive architecture. Chełmno's several churches date from the thirteenth and fourteenth centuries. On the fourteenth of February each year, the inhabitants ostentatiously celebrate Saint Valentine's Day since the local parish church has kept the saint's reliquary for many centuries.

After a short stroll through the town, we retired to a local restaurant where we enjoyed a delicious buffet dinner. Entertainment was provided by a group of musicians including Bogusław Jackowski's daughter.

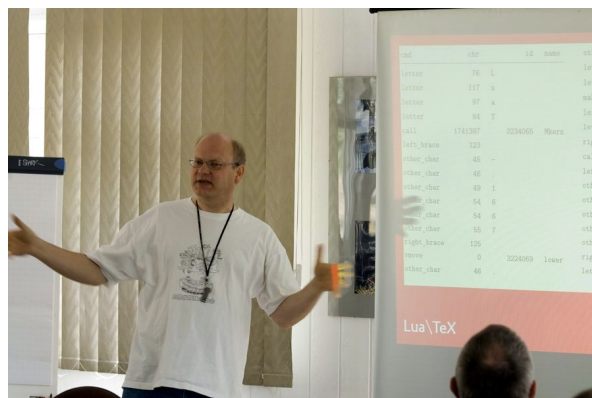


Tuesday

In the first presentation Tuesday, **Hàn Thê Thành** presented a summary of font-related topics in pdfTeX. Some, like font expansion and margin kerning, are

already documented in the pdfTeX manual. The rest are scattered across README and example files, e-mails and mailing lists. For the first time, all these topics were brought together in one place. Topics include adjusting letter and inter-word spacing, adding additional kerning before or after certain characters from a font, Unicode support for browser cut, paste and search actions and sub-fonts – a mechanism for supporting CJK languages.

Hans Hagen began his presentation by describing the issues driving the development of ConTeXt's font system, namely switching between different font styles and sizes, and proper font handling in math mode. To make font switching easier, ConTeXt can assemble a collection of different fonts into a single structure called a typescript. For example, a typescript might use palatino-regular as the default serif font, palatino-sans as the sans font, courier as the mono space font and euler as the math font. Instantiating this typescript would make these fonts available when using the commands `\rm`, `\ss`, `\tt`, and `$$` respectively.



Hans concluded by describing how the trend toward OpenType fonts, consistent user interfaces and DTP-like functionality will continue to inform where and how ConTeXt controls fonts – and vice versa.

Taco Hoekwater explained how LuaTeX, with its native support for OpenType fonts, will obviate the need for static font metric files. Currently LuaTeX implements a few dozen callbacks at strategic points in TeX. When populated, callbacks will override TeX's default behaviour with custom code. Taco demonstrated how, when using OpenType fonts, LuaTeX callbacks invoke code that extract the font metric information directly from the OpenType font itself.

Grzegorz Murzynowski identified two differing opinions concerning the TeX & Co. logos. The first group contends that the font is part of a logo, and

therefore the combination is inviolate. The second group contends that a logo should be typeset in the same font as its context. For the latter group Grzegorz suggests several slight modifications to the LaTeX logo to make it fit better with various fonts.

Sam Guravage, the youngest speaker ever to address a BachTeX conference, explained how he uses TeX for all his school assignments. Sam enumerated what he found easy in TeX e.g. sectioning and lists, and what he found difficult e.g. figures and error messages. Sam's conclusion was that TeX makes his work look better, and looking better meant higher grades.



David Kastrup began a series of talks by introducing `qstest` – a LaTeX macro package for writing regression tests. The idea is that a user can include a number of tests in his `.dtx` files and use pattern and keyword lists to specify which tests should be run; either when his package is loaded or while running a separate test file through LaTeX. The `qstest` package, together with the `typedtx` documentation format and `docstrip`, allows one to integrate unit testing and documentation in a single `.dtx` file.

David Kastrup continued with a discussion of the `makematch` LaTeX macro package. Factored out of the `gstest` package, `makematch` matches patterns with wildcards against a list of targets.

David Kastrup concluded his series of talks by explaining how the `bigfoot` macro package, originally written as a footnote apparatus for text-critical editions, can benefit the ordinary LaTeX user. For example, default footnote behavior bypasses TeX's global pagebreak optimization whenever a footnote does not completely fit on one page. In contrast, footnote breaks in `bigfoot` are reconsidered for each possible breakpoint of the main text. This means TeX will find the optimum combination of breaks in main and footnote texts.



Robustness, optimization, color continuity and paragraph footnotes are just a few reasons why LaTeX users might consider using `bigfoot` to replace TeX's native footnote apparatus.

Klaus Höppner walked us through the process of creating PostScript Type 1 fonts from METAPOST sources using `MetaType1`. Created by Bogusław Jackowski, Janusz Nowacki and Piotr Strzelczyk, `MetaType1` is a collection of tools including METAPOST, `t1utils` and AWK; together they are used to generate PostScript Type 1 AFM, TFM and PFB files. Though documentation was scarce, `MetaType1` proved to be the correct tool for the job.

Petr Sojka and Michal Růžička explained how they generated PDF, HTML and XHTML+MathML output from a single LaTeX source file. While many single-source publishing approaches begin with XML, the amount of mathematics involved made TeX the only viable input format. By enforcing a strict separation of form and content, and modifying the TeX4ht sources, the authors were able to realize individual workflows for each output format.

Péter Szabó reflected on his experience compiling various conference proceedings – including those of last year's EuroTeX conference. Péter described how the judicious use of procedures and tools can clarify and simplify the work of authors, editors and printers. Revision control software, mailing lists, shell scripts, utilities, instant messaging and of course TeX, can be combined to realize effective publication workflows.

David Kastrup described DocScape Publisher, an XML oriented database publishing system from QuinScape GmbH. At its core, DocScape uses LaTeX, pdfTeX, and David Carlisle's `xmltex`. Current applications include financial reports, a variety of product catalogs, and online excerpts.

Karel Piška described procedures and programs he has developed for comparing and viewing font elements. His workbench can be downloaded from: <http://www-ep.fzu.cz/~piska/tfcpr.html>. From this set, Karel demonstrated several tools:

cprpk, cprpkt1, cprpkt1c, cprtckpk and cprpkpk:

tools for comparing two bitmapped representations of a glyph pair at two different resolutions.

prfkrn, prfkrna, cprkrn and cprkrna: tools for comparing kerning pairs in two (or three) relative T_EX fonts, or in two releases of one font.

prfof and cprofof: tools for comparing and proofing outline fonts.

In his second presentation, **Karel Piška** applied his tools to analyze and verify the Latin Modern fonts. His results included examples of individual letter defects and inconsistencies. Interestingly, he found an inordinately large number of kerning pairs; the majority of which he thinks are not relevant to any language. Through his exacting work, Karel is improving the quality of the fonts we use everyday.



Janusz M. Nowacki unveiled his complete set of Latin glyphs for the Cyklop font. Designed and cast in lead in Warsaw in the 1920s by J. Idźkowski, Cyklop is a very heavy sans-serif two-element font. Originally produced only in the oblique form, in sizes from 8 to 48 pt, Cyklop is used for newspaper titles, posters, forms, labels and invitations. In addition to the new Latin glyphs, Janusz has added a complete new upright variant.

To round out the day, an informal reception was held in the lecture hall, where participants could enjoy a glass of wine, pleasant conversation, and an exhibition of black and white prints taken by Janusz Nowacki.

Wednesday

Paweł Jackowski presented this years crop of T_EX beauties and oddities, sixteen in total. You have to see these pearls to believe them. The entire collection can be found at: <http://www.gust.org.pl/pearls>.



Ross Moore spoke about his experience typesetting articles for The Journal of The Australian Mathematical Society. Leveraging the interactive capabilities of PDF, AMS journal articles, available free online, now incorporate lots of useful meta data that readers would otherwise have to research themselves.

To enlighten our path to the future, **Arthur Reutenauer** recounted T_EX's recent history. Subtitled "Pax T_EXnica – The program on which the sun never sets", Arthur described how, from T_EX78 to Aleph, X_YT_EX and LuaT_EX, the various T_EX engine extensions and macro packages have gradually enabled us to typeset every language and script of the world – well almost.

Ulrik Vieth presented an overview of the T_EX historic archive, an archive of historic T_EX distributions and packages hosted on the TUG FTP server (<http://ftp.tug.org/historic/>). T_EX's history spans thirty years now, and while its early history is well documented, the history of various macro packages, fonts, and systems like METAFONT and METAPOST must often be pieced together from anecdotal evidence.

After thirty years, the history of T_EX remains an interesting topic of research. The archive contains a wealth of information, but gaps still exist. Contributions are welcome, especially those about (pdf)T_EX and Latin Modern fonts.

Bogusław Jackowski, Jerzy Ludwichowski and Janusz M. Nowacki described the current status of the two large font projects being developed by T_EX User Groups: Latin Modern and T_EX Gyre.

The Latin Modern fonts project was begun in 2002. Based on Computer Modern, the Latin Modern family currently consist of seventy two text and twenty math fonts; available in both OpenType and PostScript Type 1 formats.

The Gyre font project that was begun in 2006 aims to supplement the thirty three URW++ fonts distributed with GhostScript to cover all Latin languages, similarly as the LM fonts do. Hinting is improved and files in OpenType format are provided. Extensions to the math capabilities are planned for the near future.

Here are those T_EX Gyre fonts which have already been given new names:

Original name : Gyre name	
Avantgarde	: Adventor
Bookman	: Bonum
Courier	: Cursor
Helvetica	: Heros
Palatino	: Pagella
Century Schoolbook	: Schola
Times	: Termes
Zaph Chancery	: Chorus

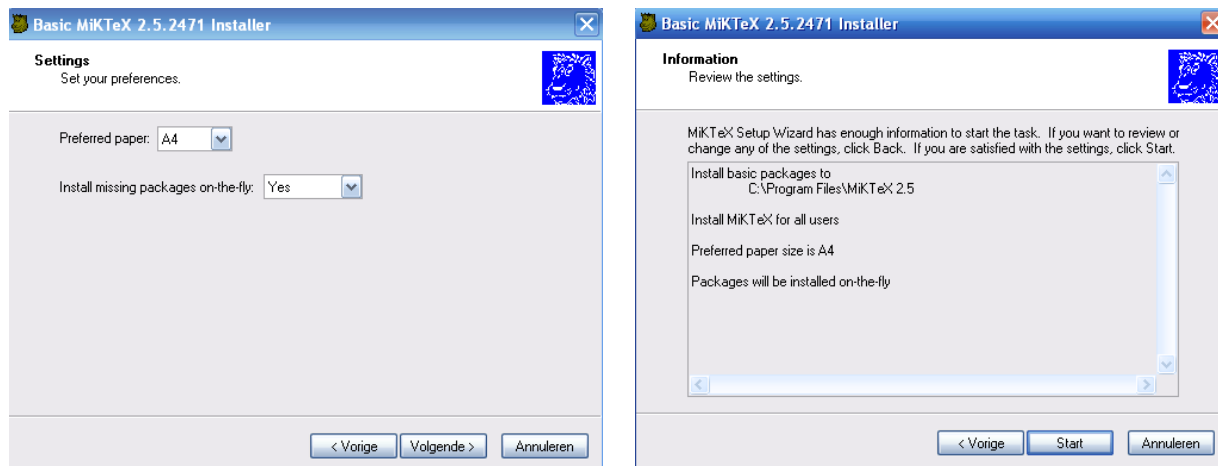
The Latin Modern and Gyre project pages are found on the <http://www.gust.org.pl> website. The respective folders are /projects/e-foundry/latin-modern and /projects/e-foundry/tex-gyre.

Recalling Niklaus Wirth's statement that "algorithms plus data structures equal programs", **Marek Ryćko** demonstrated how to realise Lisp like structures and methods in T_EX. Marek argued that a clean and consistent approach to handling lists of elements will make programming T_EX simpler, and T_EX programs, i.e. macros, more reliable.

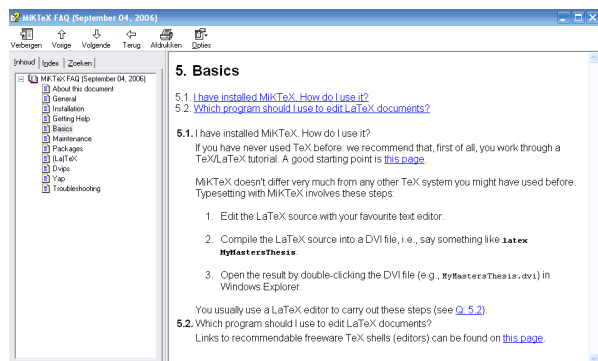


Jerzy Ludwichowski concluded the conference proceedings by thanking the organizers, authors and participants. And as a particular encouragement, the GUST board awarded Sam the award for the best conference presentation. The award was impressed on one of the handmade paper sheets.

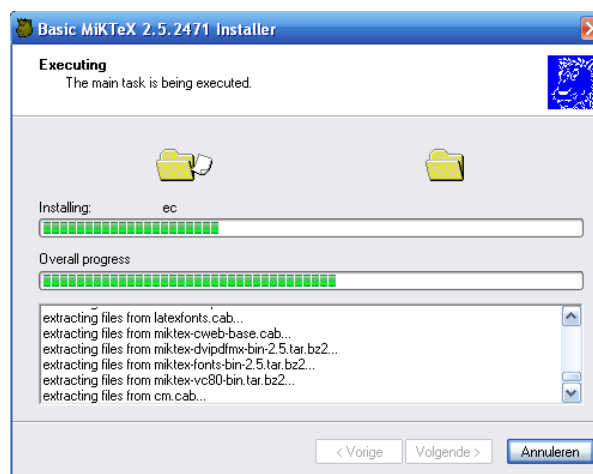
Michael Guravage
(Hans Hagen, Taco Hoekwater)



Figuur 3. De keuzes



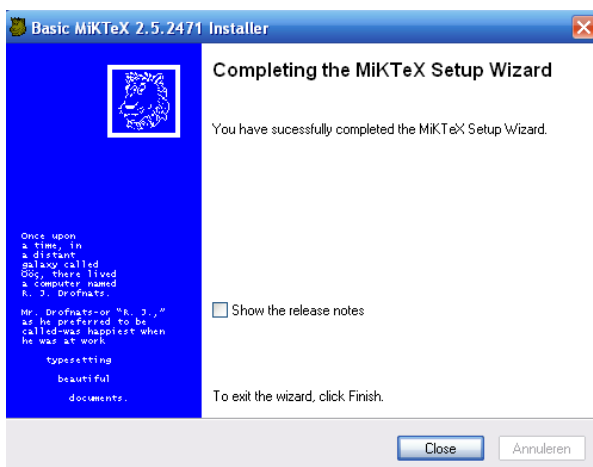
Figuur 4. Een pagina uit de FAQ



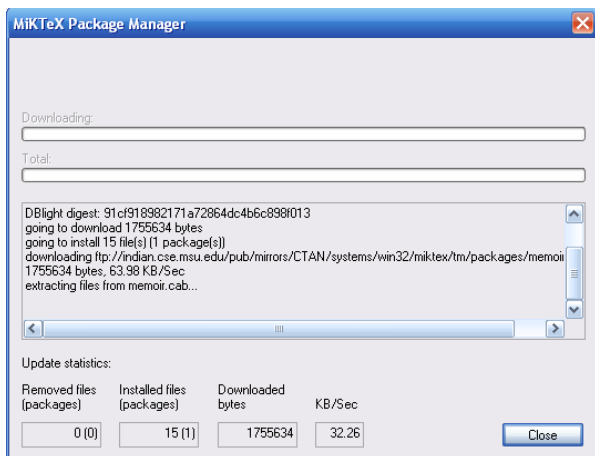
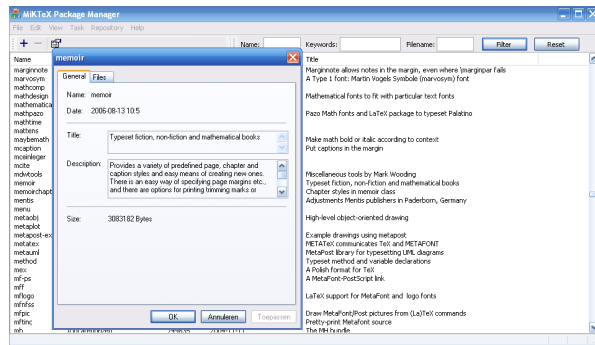
de vraag (figuur 3) of ontbrekende pakketten zonder omhaal mogen worden opgehaald van internet of dat me dat eerst moet worden gevraagd en ik kies voor “on the fly” ophalen. Mist er wat, meteen doorpakken, installeren en verder met m’n document.

Dan passeren wat schermen (figuur 5) waarin ik steeds voor “Start”, “volgende”, of “Sluiten” kies en daarna is de installatie klaar. Wat nu? Ik verwacht een icoon op het XP-bureaublad, met daarachter een “gezicht” van MiKTeX of een WinEdt-achtige editor die alles overkoepelt, maar dat bleef uit. Nu lijkt het eerst alsof de MiKTeX installatie zichzelf met succes heeft verstoep.

In het XP-startmenu vind ik wel “MiKTeX 2.5” en een van de onderdelen daar is een MiKTeX FAQ. Daarin staat bij hoofdstuk 5 (figuur 4) de vraag die volgens mij bovenaan hoort: “I have installed MiKTeX. How do I use it?” Het antwoord is “you can now say ‘tex



Figuur 5. Wat er zoals passeert tijdens de installatie

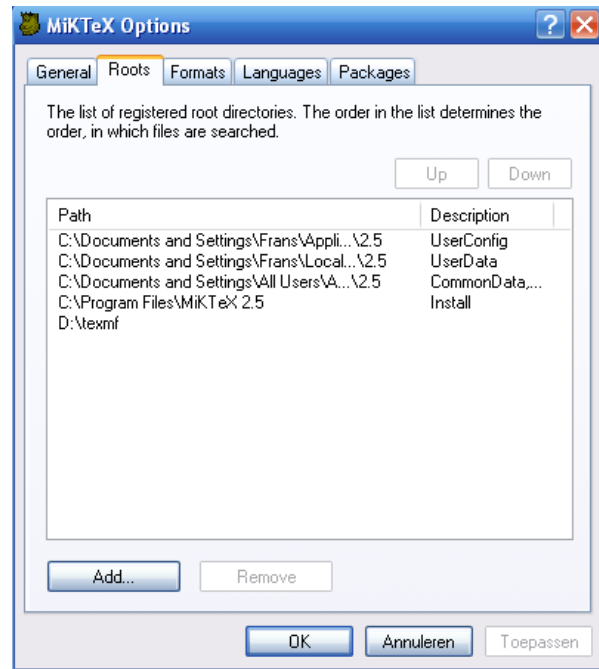


Figuur 6. Package manager

filename' — maar er staat niet bij hoe je dat tegen je machine moet 'zeggen'. Een simpel scherm en een aantal meteen-werkende voorbeelden van brieven, boeken, beamer-presentaties en artikelen zou toch voor de hand liggen met zoveel knappe koppen aan het werk voor de \TeX distributies.

In dezelfde MiKTeX map staat ook een Package Manager (figuur 6) en die ziet er simpel maar erg veelbelovend uit. Een pakket als 'memoir' staat er, bijvoorbeeld. De laatste tijd heb ik daar een paar maal over gehoord maar ik zag er tegenop in het pakket te gaan zoeken, dan te zien hoe de installatie ook alweer moet (de DTX ervan vinden, die compileren alsof het een document is en dan de resultaten ergens in de texmf structuur plaatsen, ik herinner me dat het zo ongeveer moet maar zou het ook lukken?) De Package Manager kan het voor me ophalen en installeren, lekker makkelijk!

Er staat ook een DVI previewer in de MiKTeX map maar ik heb al zoveel jaren geen DVI gebruikt... zouden er nog mensen zijn die DVI willen in plaats van



Figuur 7. MiKTeXinstellingen

PDF?

Een "settings" programma in de MiKTeX map is een van de dingen die ik zou verwachten als menu-onderdeel van "het" MiKTeX programma op mijn desktop. Wie eerder was vergeten te kiezen voor "on-the-fly" ophalen van ontbrekende pakketten kan dat hier goedmaken en wat voor mij belangrijk is: er is een tab met de naam "roots" (figuur 7) waar je kunt aangeven waar je je eigen vertrouwde texmf boeltje hebt neergezet, als je bijvoorbeeld al eens op een andere computer met \TeX aan de slag bent gegaan. Daar ben ik blij mee want op mijn Apple heb ik in de loop van de tijd allerlei eigen materiaal opgeslagen, aangepaste stylefiles bijvoorbeeld om mijn briefhoofd met logo mee te maken, en inder tijd van Wybo Dekker gekregen bestanden om old-style cijfers te kunnen gebruiken in Times en Palatino. (<http://www.servalys.nl/tex/index.html>). Ik kopieer die texmf map met alles eronder naar de D: schrijf op de XP en geef in "roots" op dat daar gezocht mag worden.

Voor mij voelt het toch aan alsof ik iets heb waar ik "niet bij kan", een super fiets zonder zadel en stuur. Pas als ik het vertrouwde WinEdt ophaal van www.winedt.com en installeer, zit er een kop op de MiKTeX installatie: WinEdt heeft in de gaten dat MiKTeX er is en functioneert als cockpit voor het opstellen, compileren en bekijken van \TeX documenten.

