# Table of Contents

# Proceedings

MAPS 39 • TUGboat 30:3 • Biuletyn GUST 25 (2009) • Die TEXnische Komödie 1/2010 • EUROTEX 2009

REDACTIE

Taco Hoekwater, hoofdredacteur
Barbara Beeton
Karl Berry

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

**Voorzitter**
Taco Hoekwater
ntg-president@ntg.nl

**Secretaris**
Willi Egger
ntg-secretary@ntg.nl

**Penningmeester**
Ferdy Hanssen
ntg-treasurer@ntg.nl

**Bestuurslid**
Karel Wesseling
karel.wesseling@ntg.nl

**Bestuurslid**
Prof. dr. ir. F. G. J. Absil
frans.absil@ntg.nl

**Postadres**
Nederlandstalige TEX Gebruikersgroep
Maasstraat 2
5836 BB Sambeek

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

**E-mail bestuur**
ntg@ntg.nl

**E-mail MAPS redactie**
maps@ntg.nl

**WWW**
www.ntg.nl

Copyright © 2009 NTG

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

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

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

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

**Productie.** De Maps wordt gezet met behulp van een LaTeX class file en een ConTeXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.40.10 draaiend onder Linux 2.6. De gebruikte fonts zijn Linux Libertine, Inconsolata, schreefloze en niet-proportionele fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

**The TeX Users Group (TUG)**: This publication is also known as issue 30:3 of TUGboat, the journal of the TeX Users Group: http://tug.org/TUGboat

**Polska Grupa Użytkowników Systemu TeX (GUST)**: Tę publikację, przygotowaną przez NTG, Holenderską Grupę Użytkowników TeXa, rozprowadza dla swoich członków Polska Grupa Użytkowników Systemu TeX – GUST jako Biuletyn GUST (ISSN 1230-5650).

**Deutschsprachige Anwendervereinigung TeX e.V. (DANTE)**: This publication is also known as issue 1/2010, 22. Jahrgang of Die TeXnische Komödie, the journal of DANTE e.V.

**Československé sdružení uživatelů TeXu (CSTUG)**: This publication was distributed by the Czechoslovak TeX Users Group to its members.

# Table of Contents

# EuroTEX 2009

In 2005 we had an NTG user meeting on one of the Dutch islands, Terschelling, which was hosted by active TEX users at the Royal Navy. This meeting was organized so well that we didn't have to think twice when Frans Absil offered the facilities of the Netherlands Defence Academy close to The Hague and Delft for a EuroTEX conference. We settled for 2009. As we were also up to the third ConTEXt conference, we decided to combine it with EuroTEX as a parallel program. Taco Hoekwater offered to organize both.

Having visited many TEX conferences I can say with confidence that this was a real nice one. Frequent visitors at TEX conferences agree on the fact that it is good to stay all at the same spot and keep hotel accommodation and conference facilities close together. In this case perfect catering as well as free (!) accommodation and facilities only made it even better.

Of course not all was a serious matter. We had a one-day trip around the middle-west part of the The Netherlands, which included a walking tour around the city of Delft and visiting some famous traditional waterworks, of course including windmills. We had a nice tour around and dinner at one of the old waterline defence fortresses.

Taco did a fine job of organizing everything. He even went so far as attending the required medical training so that we could stay on the premises 24/7. I also want to thank Frans Absil and the Defence Academy for inviting and supporting the TEX community so generously.

Hans Hagen

**A big thank you to our sponsors**

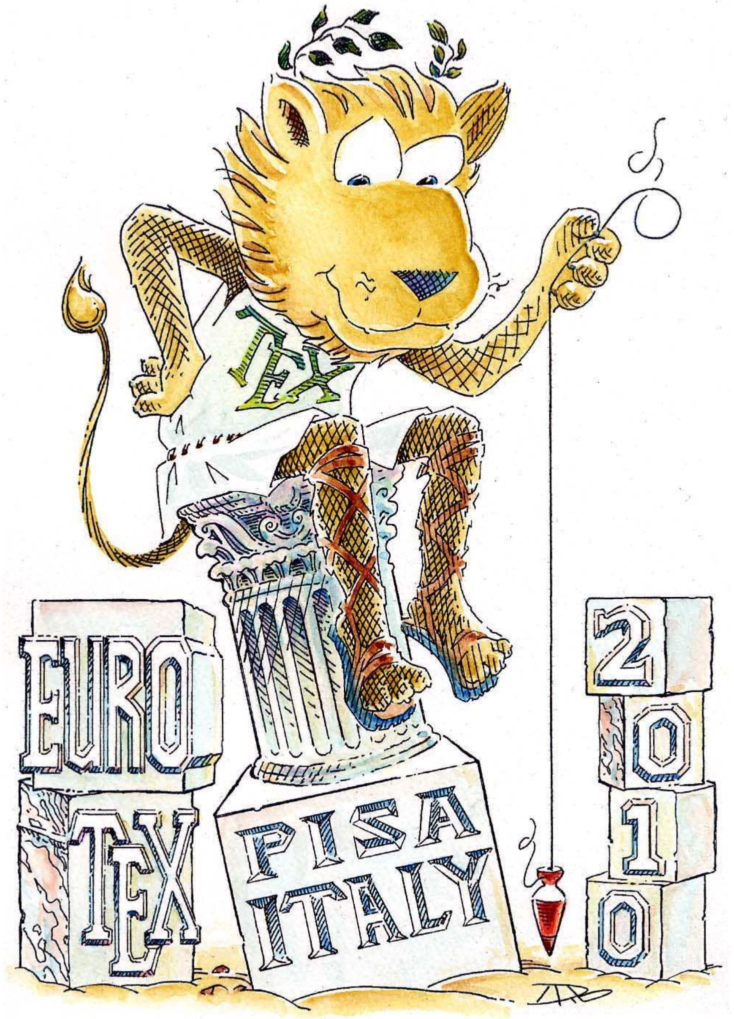
ELVENKIND

BITTEXT

NTG

# EuroTeX 2010

The Italian TeX User Group (GuIT) is very proud to invite you to EuroTeX 2010. The conference will be held from 25 to 29 August at Sant'Anna School of Advanced Studies in Pisa, Italy.



Further information on the registration, programme, accommodation, and social events will be soon available on our website:

http://www.guit.sssup.it/eurotex2010/eurotex.en.php

We hope to see you soon in Pisa.

# 4th ConTeXt Meeting

## September 13–18, 2010
## Brejlov (Prague), Czech Republic

**Meeting**

- meet new TeX friends, present your results and ideas
- get help from the experienced users
- get in touch with the latest developement of ConTeXt and LuaTeX
- Monday evening to Saturday morning

**Place**

- Mill *Brejlov*: a place to work & rest, http://www.brejlov.cz
- on the bank of *Sázava* river, beautiful countryside
- 30 km southeast of *Prague* (near *Týnec nad Sázavou*)
- enjoy swimming in the river, canoeing, walking, or cycling
- taste Czech cuisine, beer & wine
- visit *Prague* on the weekend before or after the meeting

See you in Brejlov!

http://meeting.contextgarden.net/2010

CSTUG

## TUG 2010
## TEX's $2^5$ anniversary!

**Presentations covering the TEX world**

http://tug.org/tug2010 ▪ tug2010@tug.org

**June 28–30, 2010**
**June 29: Introductory LATEX workshop**

**Sir Francis Drake Hotel (Union Square)**
**San Francisco, California, USA**

*With special guest appearances by*
**Donald E. Knuth**
*and other original Stanford TEX project members:*
**David Fuchs, John Hobby, Oren Patashnik, Michael Plass, Tom Rokicki, Luis Trabb-Pardo**

- March 1, 2010 — presentation proposal deadline
- March 26, 2010 — early bird registration deadline
- June 28–30, 2010 — conference and workshop

*Sponsored by the TEX Users Group and DANTE e.V.*

## TUG Institutional Members

American Mathematical Society,
*Providence, Rhode Island*

Aware Software, Inc.,
*Midland Park, New Jersey*

Banca d'Italia,
*Roma, Italy*

Center for Computing Sciences,
*Bowie, Maryland*

Certicom Corp.,
*Mississauga, Ontario, Canada*

CSTUG, *Praha, Czech Republic*

Florida State University,
School of Computational Science and Information Technology,
*Tallahassee, Florida*

IBM Corporation,
T J Watson Research Center,
*Yorktown, New York*

Institute for Defense Analyses,
Center for Communications Research, *Princeton, New Jersey*

Konica Minolta Systems Lab Inc,
*Boulder, Colorado*

MacKichan Software, Inc.,
*Washington/New Mexico, USA*

Marquette University,
Department of Mathematics, Statistics and Computer Science,
*Milwaukee, Wisconsin*

Masaryk University,
Faculty of Informatics,
*Brno, Czech Republic*

MOSEK ApS,
*Copenhagen, Denmark*

New York University,
Academic Computing Facility,
*New York, New York*

Princeton University,
Department of Mathematics,
*Princeton, New Jersey*

Springer-Verlag Heidelberg,
*Heidelberg, Germany*

Stanford University,
Computer Science Department,
*Stanford, California*

Stockholm University,
Department of Mathematics,
*Stockholm, Sweden*

University College, Cork,
Computer Centre,
*Cork, Ireland*

University of Delaware,
Computing and Network Services,
*Newark, Delaware*

Université Laval,
*Ste-Foy, Québec, Canada*

University of Oslo,
Institute of Informatics,
*Blindern, Oslo, Norway*

# TEX Education – a neglected approach

**Abstract**
In this note a proposal about education is made and some education is done. Why not offer a macro writing Master Class, in plain TEX&MetaPost via internet, to endorse Minimal Markup and to strive after readable, adaptable, robust and reusable macros, by using paradigms? The macros are destined to be included in a library for reuse in AnyTEX. Educational issues are: language, awareness, insight, and TEXnique proper. Courseware is indispensable. The personality of the teacher is all important. Self-study is not enough and considered dangerous. A few turtle graphics macros for line-drawing in plain TEX, used for sketching a binary tree and fractals, are given. Knuth's gkppic macros are used for flowcharts. Of both their advantages and disadvantages are mentioned. Graphics with curved lines, via PS and MP, such as envelope, smiley, yin yang, Schröfers opart, and a circle covered by circles are included. 2.5D graphics emulated from Naum Gabo constructive works and Escher's impossible cube, both specified by data in 3D and projected on 2D with the viewing angles as parameters, are supplied. Without explanation Spirals on the sphere and a torus are included. Reuse of macros, pictures, references, tools, formats is relevant with my blue.tex released in about 1995, as an unusual, integrated example, to be re-released under LPPL after review on the TEX Live Distribution DVD of 2010. At the end a suggestion is done to extend MetaPost towards 3D.

**Keywords**
2.5D, automatically numbered items, awareness, BabelTEX, binary and trinary tree, `blue.tex`, ConTEXt, data integrity, education, Escher, Gabo, H-fractal, impossible cube, insight, Malevich, master class, MetaPost, minimal markup, PDF, PostScript, plain TeX, regular surface, reuse, smiley, suprematism, TEX Collection DVD, turtle graphics, yin yang

**Contents**

## What TeX&Co education is available?

Searching the internet with `TeX education` as keywords yielded no off-the-shelf courses. When I added the keyword `tutorial` I found the good-looking LaTeX tutorial from the Indian TeX User group. Possibly the law of diminishing returns applies: our TeX tutorials are provided on the actual TeX Collection DVD and the outdated 4AllTeX CDs, and stored in the TeX archives of old, founded in the pre-WWW time, read before HTML appeared, if not in books. Times have changed. An effective way to make oneself known these days is by WWW pages, with keywords to be spotted by search engines.

With respect to MetaPost I found via Google a nice tutorial by `A. Heck: MetaPost by doing`, published in MAPS of 2005. I know of Hans Hagen's `MetaFun`, but a link to his MetaPost work did not show up in Google. Also the work of Troy Henderson with his MetaPost introduction, embraced by TUG, and his MP-previewer accessible via the internet, is worth mentioning. In fact helped me a lot.

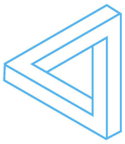
An interesting PostScript tutorial I found under `Dev Central`, which is much in the style of my `Just a little bit of PostScript`, `MAPS96.2`. Dev Central also provides for other interesting tutorials.

History has it, that TeX related courses are offered along with the TUG, EuroTeX, or LUG meetings. With this EuroTeX announcement I missed subscription forms for classes, but maybe that is not special for a EuroTeX nowadays. While this paper was underway the participants of EuroTeX received an invitation to participate in a Math Content Workshop. In the program I found LaTeX for beginners, and open sessions around ConTeXt. At the conference a tutorial on layers in ConTeXt was organized. S. Kroonenberg reported about her TeX network job at the economy department of the University of Groningen. The status and plans for ConTeXt and LuaTeX were discussed in evening sessions. The news with respect to MetaPost, SVG graphics, transparency, multiple precision... was given by Taco Hoekwater.

For me the big thing was that I had missed completely, in the past seven years of my TeX inactivity, the incorporation of OpenType fonts in TeX&Co.

> Conclusion: no TeX&Co classes are offered in general.

If for comparison one searches the internet for courseware or tutorials for ADOBEs Creative Suite, a wealth of entries will appear. The unorganized world out there, in addition to Adobe itself, contributes tutorials, videos and similar to use for free.



### Education in NTG

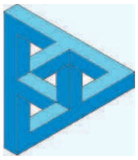
In the beginning of NTG we had working groups. The education Working Group organized a LaTeX course with courseware `Publiceren met LaTeX`, in Dutch. I was the SGML teacher at the Stanford `10 years of TeX and MF` TUG meeting. At Stanford I attended Dough Henderson's MF class. At the conference I met Amy Hendrickson and invited her to teach plain TeX in Holland. Later we had courses with David Salomon and Bogusłav Jackovski as teachers for TeX, respectively MF.[1]

A teacher is usually talkative as must be, but for the intermediate and lower level a teacher must also be a good listener, especially to find out about (mental) blockades, or misunderstandings and remove these.

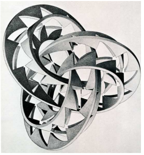
Amy, Bogus, and David were paid for their teaching and enjoyed hospitality at my place at my costs. During the Boston TUG meeting I visited AMS and invited Michael Downes and Ralph Youngen to tell us in Holland about how AMS supports their authors. I studied the clever AMS formats and styles, and criticized their too clever macro writing technique in my `AMS BLUes`.

The 4AllTeX CD effort had a strong educational flavor.

The EuroTeX bus project I consider also educational, where NTG, with the help of GUST's Julita Bolland, facilitated 20+ GUST, 20+ CyrTUG, 6 CSTUG members, and one Hungarian lady, Gy̋ongi, to participate in the Arnhem EuroTeX. Hans Hagen's ConTeXt is widespread and the accompanying wealth of documentation is highly educational.



> For TeX&Co important material can be found via the excellent WWW pages of the TeX user groups. One only has to know that!



M.C. Escher
← Knot
CGL's
→ Sort of

A weak point is the absence of links to **good** (free) TeX tutorials on the WWW.

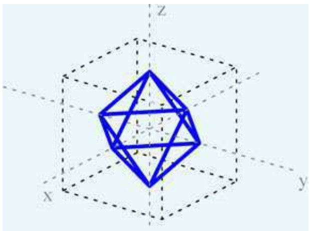
## Is there a need for TEX&Co courses?

Much self-study material is available, to start with the TEXbook and the MFbook, next to excellent tutorials. Courses are nevertheless necessary, IMHO, because of the complexity. Moreover, I consider self-study not sufficient, even dangerous.

Apparently the public does not ask for courses.

## Education material on TEX Live DVD

Nowadays, we have AnyTEX&Co on our PCs at home, such as the LATEX-biased Integrated Development Environment, TEXnicCenter, distributed on the TEX Collection DVD. However, as plain TEXie, I missed in the output dropdown window TEX→pdf and ilks. It should not be so difficult to add that, tho plain TEXies form a minority. Please do. More difficult, I presume, is to provide for MetaPost processing from within TEX. Both are necessary for my proposal with respect to a MasterClass. At the conference I became aware of TEXshop on the Mac OS X+ and Jonathan Kew's TEXworks IDE in the spirit of TEXshop under Linux and Windows XP.



The public domain TEX program, distributed on the TEX Collection DVD, comes along with a wealth of documentation, software and books, which allow for self-study of the open and public domain programs AnyTEX&Co, while for advanced questions one may consult FAQs, or discussion lists with their archived earlier discussions. I consider this a tremendous development, beneficial, except for the lack of standards in TEXing, and that it is not enough for a casual visitor of the internet, who likes to get started, despite the excellent active installation PDF documents in your language. However, self-study can be dangerous, but... in the absence of courses there is no choice. A standard in TEXing in the free TEX world is most likely not realistic, alas.

But...
we might try.

Pluriform macro writing emerged, inhibiting readability, as the most obvious characteristic. No programming paradigms, despite Knuth's example macro writing in the TEXbook: macros which constitute `plain.tex`, and macros to mark up a letter, concert, or a book, and his `gkppic` line-diagram macros in plain TEX, related to LATEXs picture environment,[2] which were used for typesetting his Concrete Mathematics. No path nor picture datastructures, no color and no filling of arbitrary closed curves. The `manmac` macros were created to typeset the TEXbook and ilks, and likely his The Art of Computer Programming œuvre. In 4AllTEX, in the `TeX` directory `GENERIC`, macros are collected, such as the `midnight` suite by van der Goot next to macros by Eijkhout, ... No stress on paradigms. The TEX Collection DVD contains a copy of the Comprehensive TEX archive, the CTAN. Searching with keyword `BLUe` yielded no match.[3]

## Education issues

Education turns around: language, awareness, insight, TEXnique proper, courseware, and the personality of the teacher.

### Language

A fundamental aspect in general education is language, it is the basis. Everybody speaks a language, can read and write, and may publish easily nowadays via the WWW. Language serves a lifetime! Language fluency is a prerequisite for participation in a culture, is fundamental in communication. Language lies at the heart of publications.

The TEXbook spends several chapters on just typesetting text, deals with the variety of

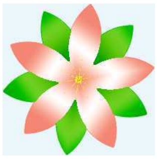
□ type faces
□ accented characters
□ ligatures
□ hyphenation
□ line and page breaking,
□ structuring commands ...

TEX is well-suited for typesetting all sort of languages, not just those based on Latin fonts. A fundamental assumption, next to the basic boxes and glue approach, is that a paragraph is the unit for hyphenation, not the keyboarded lines of input are taken as lines for the output. TEX neglects one `eol`-symbol in a row, treats it as a space. In a revision of TEX, was it 1989?, the `\language` command was introduced to facilitate for various, language specific hyphenation styles.

> TEX arose from Knuth's dream to typeset mathematics beautifully. I presume Knuth was very much surprised by the complexity of typesetting ordinary language, automatically and foolproof.

Typesetting novels by TEX is a trifle. However, book production is culture biased, with variations in layout and the used fonts. Is TeX the only tool available for this task?

Definitely not. Novels are produced by word processors, with excellent spelling checkers, I presume. I have heard of Adobe's Indesign, no hands-on experience as yet, however. MS Word I use quite often for contributions to our local gardening bulletin. These gardeners have not even heard of AnyTeX.



It amazes me that we have no BabelTeX as yet, where the commands and error messages can be instantiated for your language. For example in Russian you would then end with \пока instead of \bye, and just keyboard Cyrillics.

It also surprises me that we don't have 2 communicating windows open simultaneously: one for the editor and the another for the typeset result, as next best to WYSIWYG. Bluesky's TeX enjoyed on the fly compilation, called flash mode.

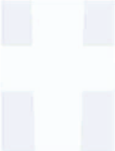
But ...

TeXworks has also the edit and the pdf window open and linked, as I learned at the conference. When an error occurs one is directly led to the line with the error in the source file in the edit window.

*LUGs* The most astonishing aspect of TeX being around is that there have arisen so many Language-biased TeX user groups. This demonstrates a relationship between TeX and languages. It is misleading to think that TeX has only to do with typesetting Math. LUGs have proven that

> a subset of TeX can be used extremely well to typeset ordinary text.

Malevich
Suprematism:
White cross on a
White background
Emulation →



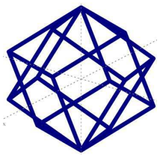
Maybe this fact should be brought to the attention of a casual user, and should not be burdened by all the other complex tasks TeX can do, which might frighten an innocent user.

I have included a picture, and its emulation, of Malevich[4] because he is the father of suprematism, which deletes the superfluous, which I associate with Minimal Markup.

*NTG was founded twenty years ago* PCs were emerging. We could access the computer centre from home by telephone through 1024baud modems. NTG started a fileserver and the digest `TeX-nl`. UNIX was taking off.

No networks nor WWW were in sight. The mainframes or midi's were accessed via terminals. I think that the meetings twice a year and the Minutes and ApPendiceS, MAPS, appearing in time, had a great impact on NTG.

I contacted budding LUGs in Germany, France, England and Scandinavia, and looked for cooperation. We cooperated in organizing EuroTeXs. In my time as president, NTG funded the LaTeX2ε project. Much later NTG took part in funding the Latin and Gyre font projects.



*Jargon* Mathematics from the language point of view is a jargon, with its own symbols, structures, meanings and definitions. In typesetting Math TeX is superb, still unmatched?[5]

But...

Microsoft, with its Cambria project and the use of OpenType Math fonts, may have taken the lead.

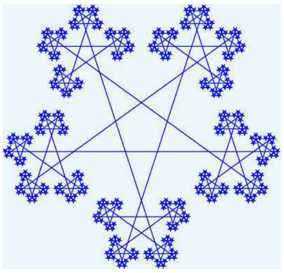
Other jargons can be typeset equally well, but you have to write the equivalent of the intelligent math mode yourself. I have not heard of such an effort.[6]

Jargon books are more complicated than novels, especially with respect to tradition in typesetting the jargon, such as: contents, back matter (an index, tables of contents, pictures, tables, . . . , references) and cross-links. For typesetting Math one must be a mathematician, have enjoyed a Math education, in short one must know the jargon.

But...

That is not enough, at least with the mathematical training I enjoyed. No typesetting conventions of math was ever taught to me. No `Mathematical writing, 1989` by Knuth as main author (who else?) existed. Happily, TeX embodies it all: the subtle spacing around Math symbols, the awareness of operators and how to typeset them in context, the composition of symbols with limits for example, is all taken care of by TeX, though TeX is not perfect. In-line Math and displayed Math may look different. The choices made by TeX, implemented in the Math mode, are wired in, but... parameterized. Math constructs can be marked up by using macros. The Math fonts are not just ordinary fonts, they come along with a suite of `\fontdim` parameters for placement of the glyphs in context by the intelligent Math mode. Using just another, commercial, Math font with the richness of the TeX math mode knowledge is not trivial. OpenType Math fonts come also with a suite of parameters, some the same as in TeX some different, and some beyond TeX's. Work on incorporating OpenType Math fonts for use in TeX, is underway by Bogusław Jackovski et al., and about to be released.

The above observations delineate the AnyTeX&Co users who make full use of the power of TeX⇔MetaPost: those who want to typeset Math (and to a lesser extent other jargons) beautifully, and ... be in complete control. Physics jargon has a lot in common with Math, and I think the Physics typesetting tradition is highly similar.

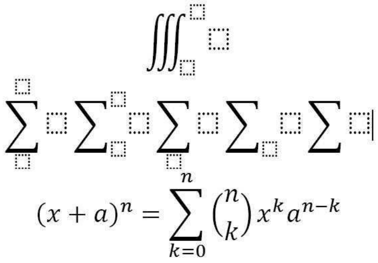


## Awareness

To be aware of what is available in order to choose the right tool for the task at hand is all important and costs-effective.[7]

> To create awareness is a main objective of education, next to acquiring TeXnique proper and learning how to cope with change and the unknown.

Awareness of competing and assisting tools is mandatory. As a WYSIWYG competing tool we have MS Word, with the `Cambria` module for Math, which uses the OpenType Math fonts. I glanced into the Math possibilities of Word 2007, and I'm impressed by what MS has achieved, especially with respect to user friendliness. Below, I have included a snapshot.



$$\iiint_{\square}^{\square}\square$$

$$\sum_{\square}^{\square}\square\ \sum_{\square}^{\square}\square\ \sum_{\square}\square\ \sum_{\square}\square\ \sum\square$$

$$(x+a)^n=\sum_{k=0}^{n}\binom{n}{k}x^k a^{n-k}$$

No confusing details, no markup tags which one must remember, just templates to be filled in, or modified. The dotted squares in the template formulas can be filled in with the needed symbols in a WYSIWYG way. Easy, isn't it? It is hard to convince users that TeX is better, I guess. The Binonium of Newton, with its 'n over k' and 'limits with the summation symbol', looks correctly typeset to me. Is TeX more flexible? Rhetorical question.

But...

MS (Math) looks easier. TeX&Co must watch out, the law of diminishing returns seems to apply.

However, in TeXnicCenter I found similar but less advanced templates. if you click on an icon the LaTeX code, will be inserted in your script, which saves you typing, and which relieves you from remembering the commands. You still have to look at the inserted tags for where to fill in. In MS the fill-in place is marked by empty dotted squares. For example, for the $\overset{\cdots}{\longrightarrow}$ icon TeXnicCenter inserts the LaTeX control sequences

```
\stackrel{}{\rightarrow}
```

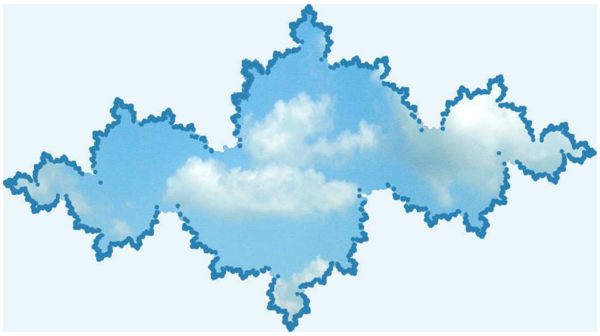
Do we have IDEs with really advanced editors with AnyTeX support, which not only prompt markup tags, but also prompt formula templates to be filled in?

With respect to WYSIWYG, we compromise by providing two communicating windows open: the editor window with the source file and the typeset window with the `pdf` result.

At the conference attention was paid to provide support for OpenType (Math) fonts for use in TeX.

*Lack of awareness* shows up when how to do typesetting tasks are published over and over again, without taking notice of, or mentioning, earlier work nor giving proper credits. Is the TeXworld anarchistic? In the last issue of MAPS, spring 2009, I read about how to typeset an addition table, which is similar to Pittman's approach of long ago of typesetting by TeX a multiplication table. The author did not mention earlier work if not by me, while the title alluded to the past. It is true that it was typeset in ConTeXt, and that is new, I guess. Superfluous in view of my plea to provide macros for creation of the table data in plain TeX to be used in AnyTeX, though in this example case the data are a trifle.

> Ignoring the past is not a scientific tradition.



I have included below the essentials of my (plain, what else?) macros for typesetting a multiplication, addition, . . . table, of a decade ago, as supplied in my `Publishing with TEX` guide, PWT, which accompanies `blue.tex`. The invoke reads

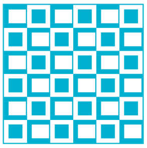
```
\def\dataMT{1\cs 2\cs 3\rs
            2\cs 4\cs 6}
```

```
$$\framed\ruled %...Attributes
  \bluetable\dataMT$$
```

The creation of the (general) data can be done as follows

```
% Creates 1 2 3
%         4 5 6
\def\rows{\c1
   \cols \advance\r1
   \ifnum\r>\mr \swor\fi
   \rs\rows}
%
\def\cols{\te\r\multiply\te\c\the\te
   \advance\c1
   \ifnum\c>\mc \sloc\fi
   \cs\cols}
%
\def\sloc#1\cols{\fi}%terminator
\def\swor#1\rows{\fi}%terminator
\def\rs{\par}%row separator
\def\cs{ }   %column separator
%
\mr2 \mc3 \rows %invoke 23 table data
```

TEX macro writing of the past, the present, or the future?



One might argue that it just generates the data, and that the complexity of markup is absent.

This is done on purpose adhering to the separation of concerns adage. More general, in my bordered table macros, I first generate the data and then do with the data whatever I have to do. A consequence of this approach is that the border of a table, which contains usually information about the entries, is handled separately. In `blue.tex` a table is composed of border, data, and caption or footer, much in the spirit of Knuth's `\bordermatrix`. By attributes one can specify framing and ilks.

> My minimal markup macro for creation of the data for the multiplication table is like Knuth's robust macros timeless, and that is what I ask you to do: write timeless, robust, mean and lean macros in plain TEX, the lowest common subset of all TEXs, to be reused in AnyTEX.[8] However, in this special case the data can just be supplied, but that is not the issue. OK, you are right we should start with creating a library of reusable parts.

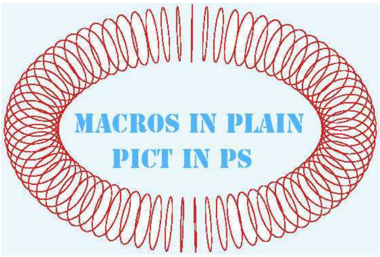
*Awareness of other tools* Phil Taylor in his recent `\parshape pre-processor`[9] starts with telling that he used HTML, because 'HTML can flow text around an embedded (rectangular) image in a totally straightforward manner, requiring nothing more than...'. He continues that he would like to use TEX and provided macros, well... his pre-processor. TEX has a more powerful, general mechanism for placing images in a paragraph.

But...
HTML, Word. . . are simpler to use. Be aware.

Sveta uses Photoshop interactively for a.o. coloring. MetaPost and ilks allow for coloring in a workflow. I don't know how to achieve by MetaPost the effects Sveta can do in Photoshop.

*Libraries for macros and pictures* are necessary for reusing parts. AnyTEX is a preprocessor of TEX! TEX itself has a built-in preprocessor called macro expander. MetaPost is a preprocessor of PostScript, and even can produce SVG, with the MetaFont ingenuities inherited. So at the basis is plain TEX and PS.

A nicety in the table chapter of PWT, next to the wealth of examples similar to those given in the TEXbook, and some more, is a little spreadsheet functionality, which can be used in (budget) table markup, to enhance data integrity. It automates addition or subtraction of table entries, which is not that trivial because TEX does not come with the ordinary calculator functionalities.[10]



This is similar to my plea of long ago in the IFIPWG2.5: write portable numerical (library) algorithms in the lowest higher-level language, FORTRAN, for use in all other higher level languages. Moreover those FORTRAN algorithms could be highly optimized, for example the matrix routines of $O(n^3)$ complexity, with $n$ the order of the matrix. At the computer center of the Groningen University we spelled out the interfacing from PASCAL, ALGOL(68), and Simula to FORTRAN, and supplied users with this information, on-line, together with the availability of FORTRAN numerical libraries. We even contracted CDC to adapt their ALGOL68 compiler towards

FORTRAN interfacing. Realistically, I expect that my plea will be partially obeyed . . . again.

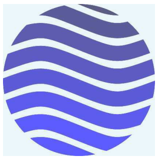
Data integrity is all important. I paid attention to data integrity in among others my bridge macros, where once a card is played, it can no longer show up in the diagrams. Data integrity was also on my mind when I created macros for typesetting crosswords.
BTW, in the suprematistic Lozenge below Mondiaan was nearly right in dividing the sides according to the Golden Ratio. This Lozenge of 1925 was the last in a series ending with this minimal one. Others have some colored parts or more lines.

P. Mondriaan
Lozenge
Composition
with two lines

The bundling of the various macros, pictures, references, tools gave rise to my BLUe collection, nicknamed after Ben Lee User in the TEXbook.

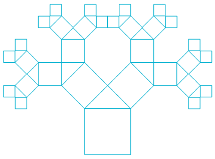
Like Knuth's plain etc macros and Berry's eTEX macros, my BLUe collection is composed of parts to be reused in anyTEX. Within BLUe what is needed can be obtained by selective loading, similar to retrieval from a database. One only loads what is needed! Even simpler, when not in the context of BLUe, is just to copy what you need, as I did for this note, see later.

But...
that is not simple enough, a library with ready to use modules is what we need.



From the absence of my BLUe in the FAQs of the UKTUG, the TEX archives, and the TEX collection DVD, I presume that the TEX community missed the reuse-of-parts aspect of BLUe. Partly true, as I became aware at the conference: the bottleneck is the copyright.

In TEX education language fluency is a prerequisite. Teach how to typeset ordinary language, technical jargon, e.g. mathematics, next to awareness of similar tools, the pro and cons of competitors.



## Insight

Characteristics of insight are

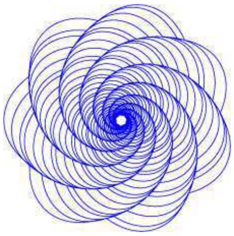
□ Abstraction
□ Separations of Concerns, SoC
□ Parameterization
□ To foresee the future
□ To use TEX&Co
□ To adhere Minimal Markup, Suprematism
□ To use Paradigms
□ To reuse parts

*Dijkstra in the past mentioned that abstraction* is our only mental tool to master complexity. As computer users we abstract all the time.

*pdfTEX violates SoC* adage. I experience a retrograde. Inserting a color for example does not obey the scope rules in pdfTEX. So the goodies of the past are annihilated. Why not keep the past achievements upright? I understand that we don't have the broad oversight Knuth had, and sin against all kinds of situations we don't foresee. Add whatever you want.

But...
without disturbing the achievements of the past, please. It is no good that a casual user like me is used as a guinea-pig. Test your materials thoroughly before releasing, please. Adhere to the practice of β-releases, such that a casual user is warned.

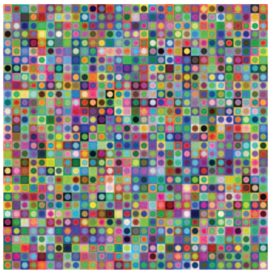
*MetaFont is the big example of parameterization* where each glyph is characterized by dozens of parameters. To handle gracefully related parameters Knuth invented the `suffix` concept, as far as I understand it is a unification of the common index and the PASCAL record, in the minimal style. In creating pictures it is a good habit to parameterize for the size, because then we can selectively scale. The line thickness is not altered if the size is changed. By blunt overall scaling the line thickness also changes, which might not be what you want.



*Knuth forecasted the future* by saying that he would use TeX a hundred years after the birth of TeX with the same quality as that of the beginning days.

*Using paradigms in macro writing* will increase readability.

One needs time and courage to invest in the use of plain TeX and MetaPost, which will serve a lifetime, and will enrich your insight. Learning just a little bit of PostScript will not harm. You will be surprised by what you can achieve by it, with Adobe Photoshop as finishing touch for (interactive) coloring or removing hidden lines.



On the other hand if the majority of the TeX community spends time and energy on LaTeX, ConTeXt, LuaTeX, in general on successors of TeX, ... it is hard to stay with plain TeX, to stay with Knuth, which means without development.
However, if one thinks a little deeper, it is an ill-posed rhetorical suggestion.

> TeX is a fixed point, only the environment changes

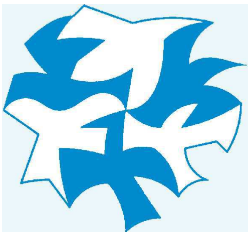
Adaptation to PS is for example taken care of by the driver `dvi(2)ps` and ilks, and pdf output can be obtained by `Distiller` or `Acrobat` (not tested though by me for a document), or just one of the various `pstopdfs`. TeX commands for handling colors and inclusion of graphics are dictated by the drivers and have to be included in `\specials`. I have no problems at all to leave MetaFont for MetaPost, because... well, again an ill-posed suggestion. I don't leave MetaFont, I just don't use it for graphics any longer, I'm not a font designer. Well,... again partially true: I'll continue to use MetaFont as my poor man's limited MetaPost previewer.

MetaFont and MetaPost have different purposes: the first one is aimed at bitmap font design, the second at creating PS graphics for inclusion in AnyTeX or `troff`. The MetaFontbook is still needed because of the unusual concepts `suffix`, `vardef`, `primarydef`, `secondarydef`, and `tertiarydef`, which MetaPost has taken over from MetaFont, and which I don't grasp completely, yet.

A middle road is provided by ConTeXt, which also comes with a wealth of documentation and is actively supported by its author Hans Hagen and colleagues, for example Taco Hoekwater.

> A real breakthrough would be an interactive TeX, which I would use immediately.

One can look upon this as what Apple did. They adopted UNIX as the underlying OS, and built their nice GUI on top. Comes TeXshop close?



*Minimal markup* As said in my PWT guide, I favor to start with just text in a WYSIWYG way. Once you have your contents more or less right, have it spell-checked, and only then insert as few markup tags as possible. The result is what I call a Minimal Marked up script.

But...
do you have the patience to work along these lines? In reality this is my logical way of working. In practice I insert already markup once I have a reasonable version. Or, do you favor to rush into code and start with `\begindocument`...etc, without having written a word of the contents yet? Marvin Minsky drew attention to this already in his Turing Award lecture of long ago 'Form versus Content.' This approach, to markup at the end and use as little as possible of TeX, is next best to WYSIWYG-TeX, and my main reason to practise Minimal Markup.

Below the essentials of my Minimal Marked up script, obeying the 20%-80% adage, for this paper is given.

```
\input adhocmacros

\author ...

\abstract ...

\keywords ...

\head Script

...
\subhead
```

```
...
\jpgD ...
...\ftn ...
\bye
```

In order to mark up in the above spirit, I have borrowed from BLUe the following macros[11]

```
\def\keywords#1\par{...}
\def\abstract#1\par{...}
\def\((sub)sub)head#1\par{...}
\def\ftn#1{...}%#1 footnote tekst
\def\beginverbatim \def\endverbatim
\def\beginquote    \def\endquote
```

while `\jpgD` was just created for the occasion. Handy is the `\ftn` macro, which takes care of the automatic numbering of the footnotes. While working on this note, which a.o. emphasizes the use of Minimal Markup, I adapted the `\ftn` macro, such that the curly braces around the footnote text are no longer needed: just end the footnote by a blank line or a `\par`, implicit or explicit.

Also convenient is the functionality of a Mini-ToC. For the latter all you need is to insert

```
%In \head
\immediate\write\toc{#1}
%In subhead
\immediate\write\toc{\noexpand\quad#1}
%In subsubhead
\immediate\write\toc{\noexpand\qquad#1}
```

Of course

```
\newwrite\toc
\immediate\openout\toc=\jobname.toc
```

must be supplied in the adhoc macros as well. Reuse on the fly!

But...

A LATEXie would shrug shoulders, because (s)he has got it all already. True!

But...

at the loss of minimal markup. A BLUe user has both the macros and the minimal markup. A matter of choice, choose what you feel comfortable with.

> If you like to concentrate on contents, clean scripts, abhor the curly braces mania, to err less and less, then Minimal Markup is for you.

## Knuth's approach

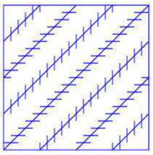
What astonishes me most is that Knuth's `plain.tex` is not embraced by the majority. His basic approach should be taught, because in TEX, well in automated digital typesetting, there are so many subtle things, where you will stumble upon sooner or later, which cannot be shielded away from you by AnyTEX, completely. Just pushing the buttons—inserting by an IDE prompted markup tags—is not enough.

How come that users did not adopt Knuth's `plain.tex`? Is it impatience, because mastering the TEXbook with `plain.tex` embodied takes time, and much more when not guided by a skilful teacher?

History has it, that first gains were preferred by adopting LATEX, which dares to explain less, keeps you unaware, which emphasizes the structure of documents, though not so rigorous as SGML, in a time when structuring whatever was en vogue. I'm not saying that structuring is wrong, not at all.

But...

one should not overdo it, one should not suggest it is the one and only. Keep eyes open, be on the alert for other aspects. On the other hand LATEX comes with a lot of packages, nowadays.



When the minimal markup attitude is adopted, one does not need that many markup instructions! The structure is already there, in what you have to say, no need to overdo it. For this note I used basically a handful of structural macros, well... a few more, to be replaced by the ones used by the editor.

> Knuth was right from the very beginning, though... not completely!

John Plaice commented on my presentation

'... that it was not possible to praise Dijkstra and Knuth in the same sentence as the two held completely opposite points of view with respect to programming languages. When Dijkstra published his 'Go to considered harmful'(CACM 11(3):147-148, 1968), Knuth defended the goto statement with his 'Structured Programming with go to Statements' (Computing Surveys 6(4):261–301, 1974).

According to Plaice, Knuth consistently supported the use of low-level languages for programming. In writing his TAOCP series of books, Knuth used his own assembler, the MIX, which he updated to a RISC machine, the MMIX, for the latest edition. His TEX: The Program book regularly makes use of gotos. The register model used in TEX programming is that of an assembler and the syntax is COBOLish.

Knuth's TEX macro language has been criticized publicly by Leslie Lamport, who stated that if he had known that TEX would have survived for so long, that he would

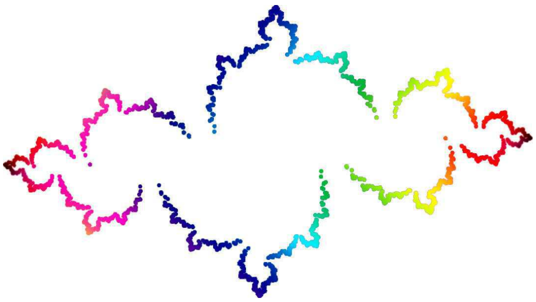
have fought much more strongly with Knuth for him to create a more usable language.'
Fair enough! Especially easier languages.

But...
I don't see what is against Knuth's attitude. Just a different approach.

Remember...
There Is More Than One Way To Do It.
I appreciate the features of high-level structured programming, with no, well... little, but well-defined side-effects, like for example exception handlers.

But...
when I learned Algol68 at the time, I was much surprised, because it seemed to me one big side effect. For me Knuth[12] knows what he is talking about, and he like nobody else, produced marvelous errorless tools, so SuPerBe documented

$$\text{S}_{\text{uper}}\text{uP}_{\text{leasanttoread}}\text{erB}_{\text{eyondthoroughness}}\text{e}$$



## TEX&MetaFont

Thirty years ago the twin TEX&MF was born. TEX is aimed at typesetting beautiful Math by computer. MF was developed for providing the needed (Computer Modern bitmap) fonts.

Knuth was apparently so convinced of the correctness of his programs that he would reward every reported bug in TEX with a check of 1$, to start with, and he would double the reward each time an error was reported. We all know the exponential growth behavior of this. In the Errors of TEX, Software Prac&Exp 19,7 1989, 607–685, Knuth mentions all the errors he corrected, to begin with those found while debugging.
TEX had the following limitations in its design, on purpose.

□ No WYSIWYG
□ No Color
□ Poor Graphics
□ No Pictures (inclusion)
□ No Communicating Sequential Processes[13]

How to overcome?
Thanks to `\special`, PS, PDF, the drivers and pdf(Any)TEX we have the following ways out, in order to compensate for what we miss

Script $\xrightarrow{\text{TEX}}$ `.dvi` as default

Script $\xrightarrow{\text{TEX}}$ `.dvi` $\xrightarrow{\text{dvi2ps}}$ `.ps` for color and graphics

Script $\xrightarrow{\text{TEX}}$ `.dvi` $\xrightarrow{\text{dvi2ps}}$ `.ps` $\xrightarrow{\text{ps2pdf}}$ `.pdf`

or directly, the popular one-step

Script $\xrightarrow{\text{pdf(Any)TEX}}$ `.pdf`

I favor the multi-step way, the 3rd, which is in the UNIX tradition of cooperating 'little' languages, where processes are 'piped' in a chain, and which adheres to the Separations of Concerns adage. With respect to TEX&MetaPost we talk about well-designed and time-proven programs.

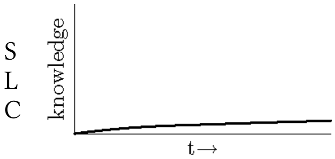
I don't believe that one person, or even a group, can write a monolithic successor of TEX, of the same quality as TEX and the time-proven cooperating little languages like `dvi(2)ps`, MetaPost, Acrobat and ilks.

In the direct use of pdfeTEX I stumbled upon..., well... undocumented features?

## Drawbacks of TEX

It is interesting to read chapter 6 of the TEXbook again about running TEX. History! Is it? Still reality for plain TEXies?
Next to the limiting choices made in the design of TEX there are the following drawbacks

□ TEX SLC 120+% Energy[14]

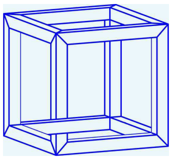


□ After so many years AnyTEX&Co lack some sort of stability. Me, for example, I don't have MetaPost running in an IDE. The TEXnicCenter is not perfect, does not provide for menus suited for plain TEX, too smart editor...
□ - No GUI[15]
But...
□ TEX&Co 99+%Quality
□ TEX&Co gives you full control

### Drawbacks of MetaFont

Nowadays, when we ask in TEX for a TEX-known font of different size, it is generated on the fly.

MetaPost, which sounds like a successor to MetaFont, was intended for creating PS graphics to be included in TEX documents, and not for creating PS fonts, despite `&mfplain`, which is not enough. MetaFont's bitmap fonts are outdated, because of the significant memory it requires and because it is not scalable. The gap was filled in 2001 by Bogusłav Jackovski et al. by releasing MetaType. Latin Modern is the PS successor of TEX's native bitmap Computer Modern fonts. Work is underway for use of OpenType (Math) fonts in TEX.



### Literate programming

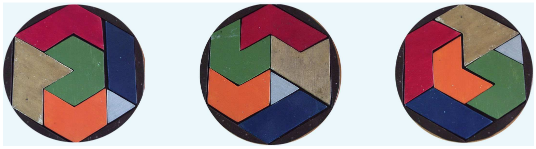
I like to characterize literate programming by

- □ Aims at error-free and documented programs
- □ Human logic oriented, not imposed by computer
- □ Programming and documentation are done simultaneously
- □ Relational instead of hierarchical

The computer science department of the Groningen University pays attention to program correctness issues, mainly for small programs, heavily biased by the loop invariance techniques of the 70-ies. No real-life approach suited for large programs like Knuth's literate programming approach, by which TEX was implemented.

But...

There's More Than One Way To Do It[16]



Even at Stanford I could not find offerings for TEX classes nor classes for literate programming.
No need? Still ahead of time?

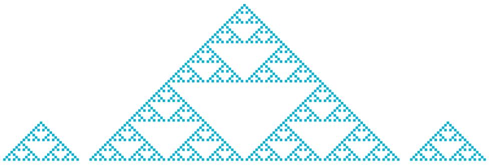
> I consider education in literate programming important, which should be provided by computer science departments in the first place.

### TEX Collection DVD

The DVD will lead you to the use of LATEX or ConTEXt. The IDE TEXnicCenter allowed me to open projects and process either

LATEX → `.dvi`, or
LATEX → `.pdf`, or
LATEX → `.ps` → `.pdf`

The Center does not provide buttons for processing plain TEX with a format file of your own, at least that is not clear to me. I had to fool the system: opened a template, threw all that I did not need away and processed my minimal plain TEX job as LATEX!
Clumsy! Did I overlook something? The possibility to adapt TEXnicCenter was on my mind for a couple of days. At last, I undauntedly defined an `output profile` and selected `pdfeTeX.exe`. Indeed, glory, my `Hello World!\bye` job worked, only the viewer Acrobat did not open automatically with the result file. The resulting `.pdf` file was stored in the same directory as the source file, so I could view it nonetheless. Not perfect as yet, because I would like to have the result opened in Acrobat automatically. Nevertheless, encouraging. I was surprised that `\magnification` did not work in pdfeTEX. I also stumbled upon that `\blue` defined as the appropriate `\pdfliteral` invoke, did not obey the scope rules. The editor in TEXnicCenter is too smart: \'e is changed into é, unless you insert a space. I reported this, no answer as yet.



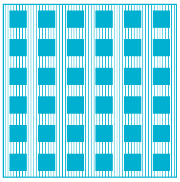
This paper asks among others to include in TEXnicCenter also buttons for processing by Knuth's `plain.tex`. It would not harm to include as example the minimal TEX job

```
Hello world!
\bye
```

or the TEX classic `story.tex`, to demonstrate the workflow

TEX→ `dvi`, `pdf` (or `ps`).

View buttons for either `.dvi`, `.ps` or `.pdf` results are nice.



I must confess that I was put off by the TEX Collection after roughly a decade of my TEX inactivity. The no longer maintained 4AllTEX CD of 1999 allowed me to TEX this paper under Vista(!) with a handful of layout (preprint) macros, to be replaced by the macros of the editor of the proceedings. The outdated 4AllTEX CD contains a lot of interesting macros. Ingenious is the refresh option by just clicking the preview window in `windvi`, which alas, does not work properly under Vista. For pdf as output I had to get familiar with TEXnicCenter. This is in contrast with for example Adobe: when you order software such as the Creative Suite, it is turnkey and works.

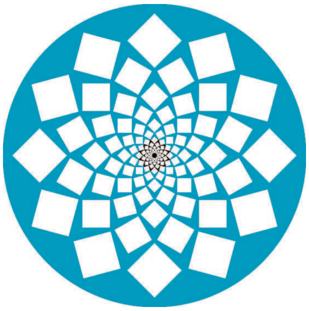
## TEXing Paradigms Master Class

The TEX arcane has become complex, very complex, and in my opinion too complex, if not for the number of languages one has know, as reported by Marek Ryćko at the BachoTEX2009.

But...
in the spirit of UNIX, or LINUX as you wish, many little languages are unavoidable.

I favor to simplify. Educate the ins-and-outs of `plain` as basis, as a vehicle for digital typography, with a wink to `manmac` when real-life book production is at stake. A beginners' course on how to use TEX is not necessary, because of the excellent tutorials, and the TEX Collection installation (active) PDF document in your language to get LATEX, proTEXt or ConTEXt running. How to run MetaPost is not yet provided for, did I miss something?

As already proposed a decade ago, I favor a class on minimal markup, on macro writing paradigms in plain TEX, which I would propose nowadays as a Master Class, not on esoterics, but on macros we all need now and then, which are full of details worthwhile to know. In my macros you will not find 15 `\expandafters` in a row. Triads? Yes!



The prerequisite is that participants are AnyTEX users, who just want to broaden their understanding, who want to gain a confident way of robust macro writing, who like documents to be embellished by minimal markup.

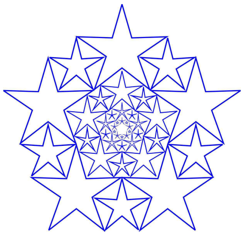
This Master Class I would organize via the internet, although a TEX course via the internet is not new. Times have changed. The teacher, or conductor, does not have to be the most knowledgeable TEXie, just the coordinator, like a chairman in a meeting. Attendees and coordinator commit themselves for a year or so, and have once a month a multiple participants session on the internet, with as many in between contacts as one can handle. The coordinator provides for exercises, where ideas to be worked out can come from participants too. In order to secure commitment the course is not for free, and the coordinator, under contract, is paid. A user group, for example NTG, backs it up and warrants continuity, for example with respect to the coordinator. For such a TEXing Paradigms activity I have material of 10 years ago as starting point, but would like to see added sessions on

□ the use of the various `\last`⟨...⟩s,
□ virtual fonts,
□ active documents, read hypertexts, and
□ TEX with calls to MetaPost on the fly.

My old headings paper will be revised with emphasis on the three generations of headings already

1. Just as in plain TEX
2. Provide for running heads in the headline and provide for a ToC creation, like in `manmac`
3. Provide for hypertext cross-referencing links and PDF bookmarks.

Of course, we might look into the CTAN for examples from the community.



The gain for participants is to master paradigms, to acquire a robust and minimal TEX macro writing technique, just like Knuth. Absolute necessary is that the TEX community backs up such a Master Class, by providing turnkey, mutual communicating TEX&MetaPost on the fly, distributed for example via the TEX Collection DVD. Although I am not in a good health, and have already said good-bye to most of my TEX materials, I am available to conduct such a class, provided there is a stand-in, and turnkey TEX⇔MetaPost IDEs for PCs.

Another dream of me is a (hyperbolic) geometry class supported by MetaPost as tool.

M.C. Escher
Limit Circle III

### TEXing Paradigms beneficial?

In the sequel I will argue why a Master Class, or is it a sort of internet workshop, for TEXing paradigms is beneficial.

It would be great if one could write a macro `\jpg` just at the moment one needs it. Reality is, I must confess, that it is not yet, well nearly, the case for me. Knuth, I was told in the past, can just do that.

In 1994 I published a note on the extension of plains `\item` macro in MAPS to provide for automatic numbering.

After my period of TEX inertia I looked at it again, and I was much surprised. I missed the following considerations, also in the revision of 1996:

1. I missed the criterion that its use should be similar, and then I mean really similar, to Knuth's `\item`, with its minimal markup.
2. I missed the reasoning why it is useful to supply such a macro. Would not just the straightforward markup
   ```
   \item1 text1
   \item2 text2
   etc
   ```
   make the need for a macro `\nitem` superfluous?
3. I overlooked that it was all about: a list of labeled indented paragraphs, each paragraph as usual ended by a `\par`—or the synonym `\endgraf`—inserted by either `\item`, `\itemitem`, `\smallbreak`, . . . `\bigbreak`, . . . , or implicitly by just the minimal markup of a blank line!

The point is: a good teacher would have drawn my attention to the ingenuity of Knuth's minimal markup, and strongly suggested not to write such a macro. Moreover, he would argue that there is hardly a need for it. MAPS was not reviewed, so neither a referee was in sight. Self-study is not enough, one needs guidance by a master. Little, first knowledge, which is usually acquired by self-study, is dangerous.

ConTEXt and LATEX provide for automatically numbered lists.

But...
within an environment, which is a clear and secure but different approach. It does not strive after utmost minimal markup.

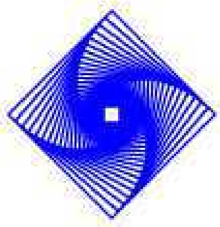
If you have to mark up a document with long lists of numbered paragraphs a minimal `\nitem` macro, similar in use as `\item`, can be handy. I would propose the following `\nitem` nowadays, where I took care of my perceived impossibility at the time by redefining `\par` at an appropriate local place. The sequence of numbered paragraphs, to be marked up by `\nitems` is enveloped by a group behind the scenes, with the advantage that one can stay ignorant of the hidden counter.

```
\newcount\itemcnt
\def\nitem{\begingroup
    \def\par{\endgroup\endgraf}
    \def\nitem{\advance\itemcnt1
            \item{\the\itemcnt}}%
    \nitem}
%\par has to be replaced by \endgraf
\def\item{\endgraf\hang\textindent}
\def\itemitem{\endgraf\indent
    \hangindent2\parindent \textindent}
```

Another, maybe overlooked[17] nicety is to have a `\ftn` macro, which maintains and uses a footnote counter. The user can just supply the contents of the footnote, does not have to worry about the (hidden) counter. My recent created Minimal Markup variant does not need curly braces around the (1-paragraph) footnote text, just end the text by a blank line. No

```
\futerelet\next\fo@t
```

needed.



If not convinced by my arguments a Master Class is beneficial by definition.

## Examples of macro writing

To show you what I have on my mind in macro writing I have supplied a few macros.

### Tough exercise

Glance at my recent solution of the TEXbook tough exercise 11.5, which is more clear and more direct than the one given in the TEXbook, IMHO, illustrating the

First-In-First-Out paradigm, as published in MAPS92.2 revised 1995, titled `FIFO and LIFO sing the BLUes--Got it?` To end recursion a (classical) sentinel is appended and macro tokens are gobbled, the latter instead of Knuth's multiple use of `\next`. The assignment inhibits processing in the mouth, which in general I do not consider that relevant. This gobbling up of macro text in the mouth I use abundantly, e.g. in my TEX macro for quicksort, as published in MAPS96.2. It also shows that sometimes we have to test for spaces and that sometimes the %-character is mandatory, especially when inadventory spaces turn you down.

My current solution reads

```
\def\fifo#1{\ifx\ofif#1\ofif
            \else \ifx\space#1\space
                  \else\blankbox{#1}%
                  \fi
            \fi
            \fifo}
\def\ofif#1\fifo{\fi}
%
\def\blankbox#1{\setbox0=\hbox{#1}
   \hbox{\lower\dp0
      \vbox{\hrule
       \hbox{\vrule\phantom{#1}\vrule}
            \hrule}}}
%
\def\demobox#1{\leavevmode\fifo#1\ofif}
%with use
\demobox{My case rests.
Have fun and all the best.}
```

Is the use of `\ifx` essential, or could `\if`, respectively `\ifcat`, have been used?

Earlier, I had a solution where I read the line word by word using a space as parameter separator, circumventing explicit testing for spaces. So, at least three variants are available for discussing the pro-and-cons. I am curious for GUST's approach, because they have the bounding boxes of each character in GUST as part of their logo. Undoubtedly ingenious.

Knuth uses the functionality in as well the TEXbook ch18, to illustrate his explanation of the transformation of a math formula specification into a sequence of (boxed) atoms, as the MetaFontbook ch12 on Boxes. Reuse, aha!

### Wind macros

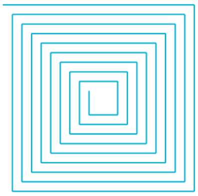
Despite the powerful MetaPost, I will talk for the first time about my plain TEX turtle line drawing graphics macros of ≈13 years ago, which I have used for a variant solution of the TEXbook exercise 22.14 (see later), but also for drawing some fractals, e.g. my favorite the Binary tree and the H-fractal (to be introduced later), a square spiral, and a Pythagorean tree, as well as a variant of the Sierpinsky triangle given at the end, as application of Knuth's dragon figures approach in appendix D of the TEXbook. The included smileys are new, they replace the old ones in `pic.dat`, because like Hans Hagen I guess, I'll do all my graphics in MetaPost or PS from now on. Revision, aha!

The wind macros can still be handy for sketches in TEX alone, although I programmed already some fractals in PS directly. If time permits I might finish my fractals note, with the graphics in PS and MP.

`\N`, `\E`, `\S`, and `\W`, draw a line element from the point (`\x`, `\y`) of size as supplied by the argument in the direction North, East, South, or West, respectively. The line element is encapsulated in a box of size 0, meaning the drawing does not change the reference point.

```
\def\N#1{\xy{\kern-.5\linethickness
  \vbox to0pt{\vss
  \hrule height#1\unitlength
   width\linethickness}}%
\advance\y#1\unitlength}
%
\def\S#1{\advance\y-#1\unitlength{\N{#1}}}
%
%\E and \W read similar, see my
%Paradigms: the winds and halfwinds. MAPS 96.1
%
\def\xy#1{%Function: place #1 at \x, \y
   \vbox to0pt{\kern-\y
   \hbox to0pt{\kern\x#1\hss}\vss}}
```

`\xy` is similar to Knuth's `\point` macro of Appendix D of the TEXbook.



A straight application of the wind macros is the above shown (inward) square spiral, which is drawn by the macro listed below, where the number of windings is supplied in the counter `\k`, and the coordinates (`\x`, `\y`) contain the starting point for the drawing. Note that during the drawing one has not to be aware of the coordinates, they are implicit. In order to display a centralized figure supply

$\backslash x = -\backslash k * \backslash unitlength$

$\backslash y = \backslash k * \backslash unitlength$

and enclose it in a box of height, width and depth

$.5\backslash k * \backslash unitlength.$

```
\def\inwardspiral{{\offinterlineskip
\loop\E{\the\k}\advance\k-1
     \S{\the\k}\advance\k-1
     \W{\the\k}\advance\k-1
     \N{\the\k}\advance\k-1
\ifnum\k>4 \repeat}}
```

### Contest

I needed for the graphics inclusion in this paper, and in the slides, a minimal markup macro \jpg.

During my presentation I launched a minimal markup problem. I wanted to replace the following markup with optional width... height...

```
$$\pdfximage height..width...
             {filename.jpg}
  \pdfrefximage\pdflastximage$$
```

by either the minimal markup

```
\jpgD filename
or
\jpgD width... filename
or
\jpgD height... filename
or
\jpgD height... width... filename
```

No square brackets, no curly braces, and even no explicit file extension, because it is already in the macro name.

I challenged the audience to write such a macro. There would be two winners one appointed by me and one by the audience.[18]

*And... the winner is...* Péter Szabó, by me and by the audience, with the following solution

```
\def\jpgfilename#1 {%
\egroup  %end \setbox0\vbox
$$\pdfximage%
\ifdim\wd0=0pt\else width\wd0\fi
\ifdim\ht0=0pt\else height\ht0\fi
{#1.jpg}%
\pdfrefximage\pdflastximage$$
\endgroup}
```

```
\def\jpgD{%
  \begingroup
  \setbox0\vbox\bgroup
  \hsize0pt \parindent0pt
  \everypar{\jpgfilename}%
  \hrule height0pt }
```

Elegant, to associate the optional parameter with a \hrule and measure it by putting it in a box. Elegant it is.

But...

I must confess, it took me some time to understand it. A Master Class would have been beneficial for me, to learn to understand these kinds of macros more quickly ☺.

The near winner was Bernd Raichle with a thorough, straight-forward solution, not avoiding the pitfall, and which is a bit too long to include in this note.

But...

it shows a superb, elaborate parsing technique looking for width.. height... and then take appropriate action.

Post conference Phil Taylor came up with a near, but intriguing solution and I myself also boiled up one. Both are supplied in the Appendix I, because much can be learned from them. Both neglect the (unintended) pitfall to concentrate on the parsing of the optional parameters. Phil and I just pass on the optional parameters, if any, to \pdfximage.

## 2D Graphics

Line drawings with incremental difficulties are included and discussed, such as

□ straight line drawings by the wind macros and by gkppic in plain TEX, and PS, such as fractals and flowcharts

□ graphics with curved lines by PS and MP, such as graphics composed of oblique lines (and spurious envelopes), circles, and general curves, to be drawn by splines.

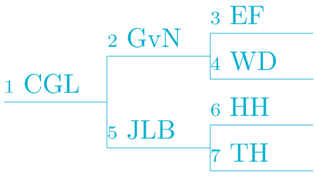
### Binary tree

I consider my binary tree macro as very beautiful, because it can be programmed so nicely in TEX or PS, and has such interesting applications. I consider it much in the spirit of Knuth's Turing award lecture Computer Programming as an Art.

```
\def\bintree{\E{\the\kk}%
   \ifnum\kk=2 \eertnib\fi
   \divide\kk2 {\N{\the\kk}\bintree}%
               \S{\the\kk}\bintree}%
\def\eertnib##1\bintree{\fi}%terminator
```

This mean and lean macro from the past can be adapted to your needs.

The above \bintree I used for a Turtle graphics, non-\alignment, solution of the TEXbook exercise 22.14, which reflects the binary structure. En-passant, NTG's VIPs replace the original names given in the TEXbook.



The previous drawing is obtained via

```
%labels in preorder
\def\1{CGL}
\def\2{GvN}\def\5{JLB}
\def\3{EF}\def\4{WD}
\def\6{HH}\def\7{TH}
%
$$\unitlength2ex\kk8 \chartpic$$
%
%with adaptation to insert the leaves
%
\let\Eold\E
\def\E#1{\global\advance\k1
  \xytxt{ \csname\the\k\endcsname$_\the\k$}
  \Eold8}}
```

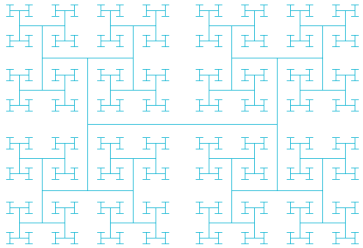
Remarks. The (educational) indices at the nodes are inserted to identify the nodes, to make the (order of) traversal explicit. The replacement text of \1 will be placed at node 1, etcetera. Adding the leaves, the initials, is done by adapting the \E macro and invoking \xytxt, which places text at the point (\x, \y). \chartpic encapsulates the Binary Tree picture in a box of appropriate size. See my earlier more elaborate note in MAPS96.1.

The variant PS code of the binary tree macro reads

```
%!PS -Bintree, cgl~1995-
%%BoundingBox: 0 0 600 600
/Bintree{/k exch def drawN
  /k k 2 div def
   k 1 gt {%
gsave drawE k Bintree grestore
      drawW k Bintree}if
   /k k 2 mul def}def %end BT
/drawN{0 k rlineto currentpoint
               stroke translate
   0 0 moveto}def
/drawE{k 0 rlineto
    currentpoint stroke translate
    0 0 moveto}def
/drawW{k neg 0 rlineto
    currentpoint stroke translate
    0 0 moveto}def
200 400 moveto 1 setlinewidth
.5 .5 scale 128 Bintree
showpage
```

I hope you will experiment with my binary tree codes, and please let me know if you have some nice, mean and lean use for it.
A more complicated use of the wind macros is the H-fractal macro as given below

```
\def\hfractalpic{%Size(2,1)*2\kk\unitlength
  \def\hf{\ifnum\level>0
             {\nxt1\hf}\nxt3\expandafter\hf
          \fi}%
  \def\nxt##1{\advance\dir##1
     \ifnum3<\dir\advance\dir-4 \fi
     \ifcase\the\dir \N{\the\kk}%
         \or         \E{\the\kk}%
         \or         \S{\the\kk}%
         \or         \W{\the\kk}%
     \fi
     \multiply\kk17 \divide\kk24
     \advance\level-1 }%
\dir=0 \hf}%end hfractalpic
```

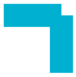


My PS variant is even more concise and reads

```
%!PS -H-fractal  cgl aug2009-
%%BoundingBox: 0 0 600 600
/Hfractal{/k exch def
  gsave draw
  /k k 2 mul 3 div def
  k 1 gt {90 rotate k Hfractal
        -180 rotate k Hfractal}
      if
  /k k 3 mul 2 div def
  grestore}def %end Hfractal
/draw{0 k rlineto
currentpoint stroke translate
0 0 moveto}def
%
300 0 moveto 1 setlinewidth
.3 .3 scale
512 Hfractal
showpage
```

With respect to graphics I favour PostScript, and why not write yourself a little bit of mean and lean PostScript? The above macros in TEX can be useful for sketches and fractals.

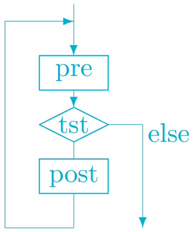
But...
when linethickness becomes noticeable, they suffer from a severe disadvantage of ugly line joinings like the notch



In MetaPost these notches can be circumvented by using suitable pens. Using in PostScript appropriate values for setlinejoin, setlinecap, or the use of closepaths for contours, will help you out.

### Flowchart

An example of the use of gkppic macros is the following diagram for the loop.



The code I borrowed from BLUe's pic.dat, adapted for the occasion with inserting \bluec and \bluel. Not nice, so another reason for doing it all in MetaPost.

```
\def\blueflowchartlooppic%
{\bgroup\unitlength=3ex%3.8ex
 \xoffset{-.5}\yoffset{-.3}%
 \xdim{5}\ydim{7.5}%
\beginpicture
%\ifmarkorigin\put(0,0)\markorigin\fi
\put(0,0){\line(1,0){2}}%
\put(0,0){\line(0,1){6}}%
\put(0,6){\bluec\vector(1,0){2}}%
\put(2,6.5){\bluec\vector(0,-1){1.5}}%
\put(1,4){\framebox(2,1){\bluec pre}}%
\put(2,4){\bluec\vector(0,-1){.5}}%
%\put(2,3){\rhombus(2,1)1{tst}}%
\put(1,3){\bluec\line(2,1){1}} %lu
\put(1,3){\bluec\line(2,-1){1}} %ll
\put(3,3){\bluec\line(-2,1){1}} %ru
\put(3,3){\bluec\line(-2,-1){1}} %rl
\put(2,3){\makebox(0,0){\bluec tst}}%
%
\put(2,2.5){\line(0,-1){0.5}}%
\put(1,1){\framebox(2,1){\bluec post}}%
\put(2,1){\line(0,-1){1}}%
%
\put(3,3){\line(1,0){1}}%
\put(4,3){\vector(0,-1){3}}%
\put(4,2.5){\kern2pt{\bluec else}}%
\endpicture\egroup%
}% end blueflowchartlooppic
```

Before the invoke I defined \bluel as well as \bluec and used \bluec in the definition and \bluel before the invoke. Not so nice, but imposed by PDF.

*Disadvantages* of the gkppic macros and LATEX picture environment is that coloring is tricky when using \pdf(Any)TeX: elements of a font are colored by supplying **k** to \pdfliteral and lines or fills are colored by supplying **K** to \pdfliteral. Moreover, one has to put the pictures drawn by the wind macros in a box of appropriate size, sort of Bounding Box, to ensure space. A nuisance.

```
\def\bluec{\pdfliteral{1 0 0 0 k}}
%versus
\def\bluel{\pdfliteral{1 0 0 0 K}}
```

When we use the multi-step processing line via PS we don't have that disadvantage. Below a test example.

```
pretext
\special{ps: 0 0 1 setrgbcolor}%blue
abc
\hrule
def
\special{ps: 0 0 0 setrgbcolor}%black
posttext
\bye
```

Process the .dvi via dvips and view the .ps, to verify the result. In TEXnicCenter I have added the Output Profile eTEX→PS→PDF. Convenient.
Note the explicit switching back to black, because the TEX scope rule is not obeyed, of course. Another way to prevent that the rest will appear in blue, is to insert gsave in the first and grestore in the second \special.

*My MP code for the flowchart* which made use of Hobby's boxes.mp.

```
input  boxes.mp;
prologues:=3;
%outputtemplate:="%j-%3c.eps";
beginfig(0)
boxit.boxa (btex \hbox to 60pt%
     {\strut\hss  pre\hss}etex);
boxit.boxb (btex \hbox to 60pt%
    {\strut\hss   tst\hss}etex);
boxit.boxc (btex \hbox to 60pt%
     {\strut\hss post\hss}etex);
boxa.c=(100,180);
boxb.c=(100,140);
boxc.c=(100,100);
boxa.dx=boxb.dx=boxc.dx=5;
boxa.dy=boxb.dy=boxc.dy=5;
drawoptions(withcolor blue);
drawboxed (boxa, boxc);
%draw contents of b and the diamond shape
draw pic.boxb;
draw boxb.w--boxb.n--boxb.e--boxb.s--cycle;
%The arrows
drawarrow boxa.s--boxb.n;
drawarrow boxb.s--boxc.n;
z1-boxb.e=z2-boxc.e=(20,0);
```

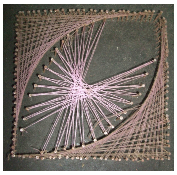
```
drawarrow boxb.e--z1--z2;
z3=boxb.w-(20,0);
z4=boxc.w-(20,0);
drawarrow boxc.w--z4--z3--boxb.w;
endfig
end
```

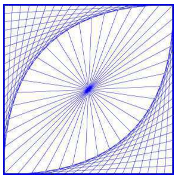
Note how the diamond diagram is obtained from the information created by the boxit invoke. Henderson's previewer does not allow the use of boxes.mp. If one can write the above flowchart with the use of gkppic macros, one could have coded it equally well direct in PS. To code in PS the generality incorporated in the boxes.mp is substantial more work, and unnecessary in view of the neat boxes.mp macros, which come with the PD MetaPost. In Goossens' et al. Graphics Companion the metaobj work of Roegel is mentioned as a generalization. To understand the boxes.mp macro is a good exercise in becoming familiar with the suffix concept, a unification of index and record, which Knuth introduced to handle conveniently the many related parameters in a glyph. By the way, the resulting PS is not standard PS, but named purified PostScript which is incorrect, because of the used cmr fonts. The coupling of the TEX cmr fonts to PostScript fonts, an old problem, is done by the prologues:=3;. The name of the resulting output file can be composed to one's wishes by assigning an appropriate string to outputtemplate.[19] The filename with extension .eps facilitates previewing via Acrobat. For previewing via CSview4.9 it does not matter. The inclusion of the string (100,100) translate as value of special hinders previewing: the Bounding Box is incorrect, the calculation of the BB does not account for this PostScript inclusion. I have the impression that the PostScript code is just passed through to the output. Inserting the desired font via special would not work either. I found it convenient to name boxes beginning with box..., to avoid name clashes. See for more details and possibilities the (latest version of the) MetaPost manual.

> If only the TEX Live DVD would have provided installation directions for MetaPost.

### Oblique lines

When my youngest daughter was at school, she produced the following object created by 'oblique lines' with spurious envelopes (left). My emulated result in MP is at the right, and created just for this note.

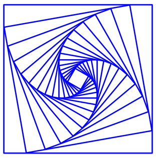




The curves, not quarter circles by the way, suggested by the oblique lines which connect points on the sides, are spurious. Below I have included the MP code.

```
beginfig(1); numeric s; s=5cm;
path l, u, cl, cr;
%...
cl=(-s,-s){up}..(-.5s,.5s).. {right}(s,s);
cr=(s,s){down}..(.5s,-.5s)..{left}(-s,-s);
for t= 0 step 0.5 until 20:
draw point t/10 of cl --
     point t/10 of cr withcolor blue;
endfor;
pickup pencircle scaled 3pt;
draw l..u--(s,-s)--cycle withcolor blue;
endfigure;
```

If we rotate and shrink appropriately we get the interesting figure



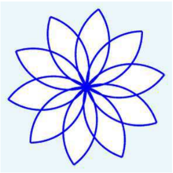
In Goossens' et al. The LATEX Graphics Companion this figure is shown as an example of a recursive object.[20]

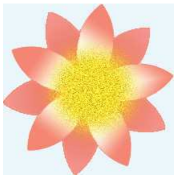
```
beginfig(0);
u=1cm;
drawoptions(withcolor blue);
for k=0 upto 15:
draw((u,-u)--(u,u)--(-u,u)--
  (-u,-u)--cycle)rotated 10k;
u:=.86u;
endfor
endfig;
```

These figures are a prelude to Gabo's 3D regular surfaces.

### PostProcessing by Photoshop

My wife, Svetlana Morozova, 'shoot-shoot' post processed the PS flower (left) interactively via Photoshop into the nice colored flower (right).





The PS code for the flower is given on the next page.

```
%%!PS Flower. CGL june 96
%%BoundingBox: -25 -25 25 25
0 0 1 setrgbcolor
100 100 translate
/r 18 def
10{r 0 r 90 180 arc
   0 r r 270 360 arc
   36 rotate}repeat
stroke
showpage
%%EOF
```

Have a try in coloring it. For comparison I have included the MP code below.

```
beginfig(1);
r=18;
path p;
drawoptions(withcolor blue);
p= (0,0){up}...{right}(r,r){down}
    ...{left}cycle;
for i=1 upto 10:
draw p rotated 36i;
endfor
endfig
end
```

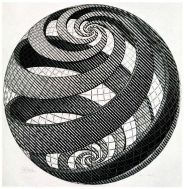
Note. In MP we don't have to translate the origin. The connection between the knots is tight, by three dots. Interesting is that PS can draw circles while MP approximates.

## PostScript spaces

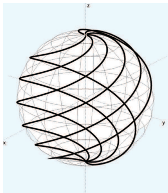
MF and MetaPost are nice graphical languages, with convenient and powerful high-level instructions. Postscript employs the notion of a 2D user space and a 2D device space, the latter can rotate. MetaPost has a path and a picture data structure, which can be rotated, well... transformed. Another difference is that PostScript is stack oriented and MetaPost enjoys a declarative language with a thorough syntaxes.

Negative from MetaPost is that the resulting PostScript code may not be concise and readable. Values of the MP variables are given in the resulting PS and not their symbolic names. Loops are unwinded.

Raw PostScript can be included via MetaPost's `special`, however. The best of both worlds?[21]



M.C. Escher
← Bolspiraal
CGL's
→ Sort of



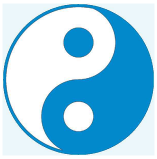
## Yin Yang

A neat timeless MetaPost code I borrowed from Hobby

```
beginfig(1);
u=1cm;
path p;
p = (-u, 0)..(0,-u)..(u,0);
fill p{up}..(0,0){-1,-2}..{up}cycle;
draw p..(0, u)..cycle;
endfig;
```

without the 'eyes'.[22]

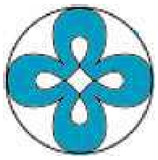
Below my enriched PostScript variant of Hobby's MetaPost code, for the real Yin Yang picture.



```
%!PS-Adobe- Yin Yang. cgl July 2009
%%BoundingBox: -25 -25 25 25
/R 25 def     /hR R 2 div def
/mR R neg def /mhR hR neg def
/r R 5 div def /mr r neg def
/rcircle {translate % center on stack
         r 0 moveto 0 0 r 0 360 arc
}def
0 mR moveto 0   0  R    270  90 arc
            0   hR  hR   90 270 arcn
            0   mhR hR   90 270 arc
fill
R 0 moveto 0 0 R 0 360 arc
stroke
gsave 0 hR rcircle fill grestore
gsave 0 mhR rcircle
      1 setgray fill
grestore
```

It differs from the MP code, because there is no direction specification in PostScript. Procrusting direction has to be done by control points in general, which are not needed in this special case. If the small discs are omitted—Hobby's original code—the PS code is not that much longer than the MP code. Orientation is immaterial.

A picture with a genuine use of control points is the Malbork window below.



Syntactic sugar? Yes, but the PS code can directly be used in documents, to be inserted by the `dvi(2)ps` driver, alas not directly by pdf(Any)TEX. Strange.

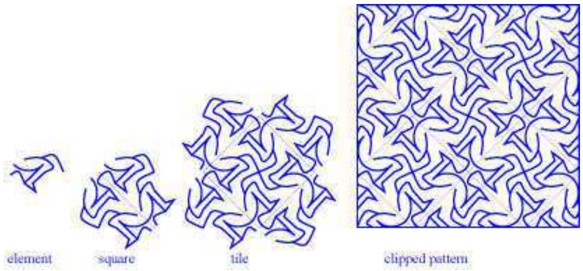
It is tempting to go for `.pdf` all the way and I was advised to convert PS pictures into PDF, which is easy via `Acrobat`, or the older `Distiller`, `Illustrator`, `Photoshop`?, or...No big deal.

However, I could not control the placement of `.pdf` pictures in pdfeTEX.[23] For this paper, and my slides, I have converted all `.eps` pictures into `.jpg`.[24]

The `.jpg`s could be integrated smoothly in the document by pdfeTEX, and was the way of picture inclusion with pdfeTEX, for EuroTEX09, because it worked and I think `.jpg` is more appropriate for pictures, despite its non-scalability without quality loss

But,..

maybe PDF, SVG or PNG is better, no hands-on with the latter two formats as yet. However, I believe— biased by the SoC principle and because I can include `.ps` pictures in my document— that the more-steps processing, via `dvi(2)ps` is better. I will explore that later.



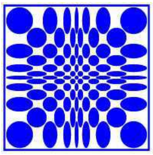
### Smiley

A simple example in MP of the use of pens is the following MP code for the smiley ☺

```
beginfig(1);
u=1cm;
color yellow; yellow=(1, 1, 0);
fill fullcircle scaled 20u
                 withcolor blue;
pickup pencircle scaled 4u;
draw ( 4u, 4u)   withcolor yellow;
draw (-4u, 4u)   withcolor yellow;
pickup pencircle scaled 1.5u;
draw (-7u,-u){1,-10}..(0,-7u)
 ..{1,10}(7u,-u) withcolor yellow;
endfig;
```

### Schröfers opart

Placement of (deformed) circles in a square has been done by the artist Schröfer in an opart way. My emulation follows.



```
%!PS -Schroefer's Opart cglnov1996-
%%BoundingBox: -190 -190 190 190
200 200 translate
/s 5 def %BB for 1 with s=5
/drawgc{gsave
   r c translate
   r abs 5 div s add
   c abs 5 div s add scale
   0 0 1 0 360 arc
   fill
grestore}def%end drawgc
%
/indices [30 21 14 9 5 2 0
   -2 -5 -9 -14 -21 -30] def
/Schroefer{/flipflop true def
indices{/r exch s mul def
  gsave indices{/c exch s mul def
        flipflop{drawgc}if
        /flipflop flipflop not def
         }forall
  grestore
}forall
-38 s mul dup moveto
0 76 s mul rlineto
76 s mul 0 rlineto
0 -76 s mul rlineto
closepath 5 setlinewidth stroke
}def%end Schroefer
gsave .5 .5 scale Schroefer
grestore
showpage
```

### EuroTEX94 battleship logo

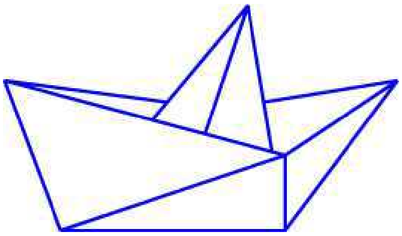
In order to illustrate the calculation of intersection points in PS I have borrowed the EuroTEX94 battleship logo.

In general we need MF or MP for solving (linear) equations within a graphics context. However, the calculation of the intersection point of 2 straight lines we learned already at highschool, albeit without pivotting strategy, for numerical best results. So, the knowledge for solving

2 equations in 2 unknowns in PS is there. I spent a little time on writing the PS code for it, ≈15 years ago, just after the 1994 EuroTEX, to write the logo in PS when MP was not yet in the public domain. I did not realize at the time that it was important, because people believe, have the prejudgement, that we can't solve equations in PS, or at least they apparently think that we don't have a def for it. Indeed, we should create a library of `.ps` defs, or maybe it exists already somewhere?

What we miss so far is that we can't specify in PS the equations implicitly as we can in MF and MP. No big deal.

From the specified points on the stack in PS we can form the coefficients for the equations, leave these on the stack and solve the equations. No more than high school knowledge is required, and... a little endurance to solve the equations, as I did maybe some 30 years ago for the HP handheld calculator, which also uses a stack and Polish Reverse Notation.



The data in PS code reads

```
%Data
/p0{0 0}def
/p1{3 s mul 0}def
/p2{4.5 s mul 2 s mul}def
/p3{3 s mul s}def
/p4{-.75 s mul 2 s mul}def
/p5{p0 top p3 p4 intersect}def
/p6{p0 p1 mean top p3 p4 intersect}def
/p7{top p1 p3 p4 intersect}def
/p8{p2 p5 top p1 intersect}def
/p9{p8 dup 0 exch top p0 intersect}def
/top{2.5 s mul 3 s mul}def
```

To specify all the points I needed a PS def `intersect` for calculating the intersection point of 2 lines determined by 4 points.
Points `p1 p2 p3 p4` → `x y`

```
/p1{0 0}def /p2{10 0}def ...
%
/p {p1 p2 p3 p4 intersect}def
%
/intersect {%p1 p2 p3 p4 -> x y
makecoef 7 3 roll
makecoef
solveit}def %end intersect
%
/makecoef{%z1 z2 -> e a b
4 copy       %x1 y1 x2 y2 x1 y1 x2 y2
4 -1 roll mul
3 1 roll mul sub
5 1 roll 3 -1 roll sub
            %(y2x1-y1x2) x1 x2 y2-y1
3 1 roll sub%(y2x1-y1x2) y2-y1 x1-x2
}def %end makecoef
```

As last piece the definition of `solveit`

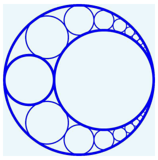
```
/solveit{%e a b f c d  -> x y
%Equations: ax + by = e   p=pivot
%           cx + dy = f
%pivot handling  %e a b f c d
1 index abs      %e a b f c d |c|
5 index abs      %e a b f c d |c| |a|
gt {6 3 roll} if %exchange 'equations'
%stack: e a b f c d or f c d e a b,
%first is in comments below
exch 4 index     %e a b f d c a
div              %e a b f d p
6 -1 roll dup 6 1 roll 3 1 roll
                 %a e b f e d p
4 index exch     %a e b f e d b p
dup 4 1 roll     %a e b f e p d b p
mul sub          %a e b f e p (d-b.p)
4 1 roll mul sub  exch div
%a e b (f-e.p)/(d-b.p) = a e b y
dup 5 1 roll mul sub exch div exch
}def %stack: x y
```

Finally, the drawing of the battleship

```
%Battleship
-2 s mul 0 translate
0 0 1 setrgbcolor
p0 moveto p1 lineto p2 lineto p3 lineto
                    p0 lineto closepath
p1 moveto p3 lineto p4 lineto p0 lineto
p5 moveto top lineto p6 lineto
p6 moveto top lineto p7 lineto
p2 moveto p8 lineto p4 moveto p9 lineto
stroke
```

### Circle covered by circles

This example is included because it demonstrates that even in PS we can solve nonlinear equations.



Essential in this code is the definition of `Bisect` for zero finding of a nonlinear function.

```
/Bisect{%In  0<=l<u f(l)<0 f(u)>0
        %Out l<=d<=u u-l<eps f(l).f(u)<=0
/fd f def
fd 0 lt {/l d def}
        {/u d def}ifelse
u l sub eps gt
  fd 0 ne and %l-u>0&f/=0
{Bisect}if
d}def %end Bisect
%
/l ...def   /u ...def
/d{.5 l u add mul}def
/f{% f: d-->f(d)
     ...       } def
```

For the complete code and the formulas for the midpoints of the circles see my Tiling note of the mid 90-ies.

## 2.5D Graphics

For drawing 3D objects in PS I discern the following spaces

□ 2D PostScript Device Space
□ 2D PostScript User Space

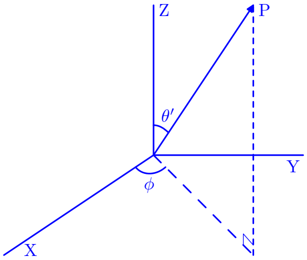
I added

□ 3D User Space, the data
and
Project 3D US
onto 2D PostScript US

By 2.5D graphics I mean an image of a 3D object, displayed as 2D graphics, obtained from 3D data specifications and projection onto 2D.

The projection formula reads

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -\cos\phi & \sin\phi & \\ -\sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



The (full) transformation matrix can be understood as to be composed of 2 rotations: first around the z-axis and second around the transformed x-axis, such that the z-axis coincides with the view direction $\vec{OP} \perp$ the new xy-plane. We can omit the 3rd, the transformed z-coordinate, because it is no longer visible in the (orthogonal) projection. The factorization of the projection matrix is

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sin\theta & \cos\theta \\ 0 & \cos\theta & -\sin\theta \end{pmatrix} \begin{pmatrix} -\cos\phi & \sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

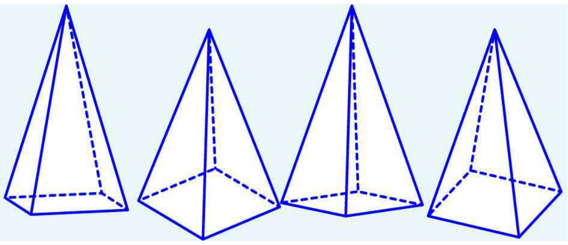
Coded in PS as a `Point to Pair` def, `ptp`, the projection formula reads

```
/ptp{/z exch def/y exch def/x exch def
x neg a cos mul y a sin mul add
x neg a sin mul b sin mul y neg a cos mul
       b sin mul add z b cos mul add}def
```

Later, in the MP code for Gabo's `linearii`, the MP vardef for `Point to Pair` is given.

### Pyramids

Hobby in his 'A user's manual for MetaPost' draws a pyramid. Below I have drawn pyramids starting from 3D data as function of the viewing angles.



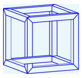
The PS code reads

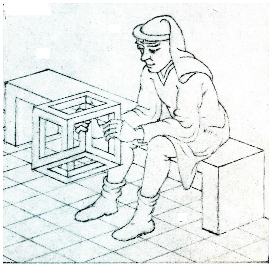
```
%!PS-Pyramid in projection, cglaug2009-
%%BoundingBox: 0 0 300 100
/ptp{/z exch def/y exch def/x exch def
x neg a cos mul y a sin mul add
x neg a sin mul b sin mul y neg a cos mul
       b sin mul add z b cos mul add}def
%
/r 20 def /hr r 2 div def
/z1{r neg r    0 ptp}def
/z2{r neg dup  0 ptp}def
/z3{r r neg    0 ptp}def
/z4{r r        0 ptp}def
/top{0 0 r 4 mul ptp}def
%
/pyramid{z1 moveto z2 lineto z3 lineto
   [2]1 setdash stroke
z3 moveto z4 lineto z1 lineto
top moveto z1 lineto
top moveto z3 lineto
top moveto z4 lineto stroke
top moveto z2 lineto
   [2]1 setdash stroke}def%end pyramid
%
30 300 translate
0 0 1 setrgbcolor%blue
1 setlinecap 1 setlinewidth
%
15  25  65{/a exch def
30  -20 10{/b exch def
           pyramid
           57 0 translate}for}for
showpage
```

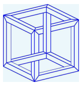
### Escher's impossible cube

As student I was asked by my professor to (re)make Escher's[25] Impossible Cube.

I decomposed it into two parts, in timber, and put them together such that in projection the impossible cube was obtained. I photographed it and handed the photo to my professor.[26]





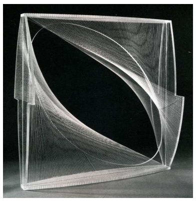


I'm happy that after so many years, I had the guts to emulate the cubes.

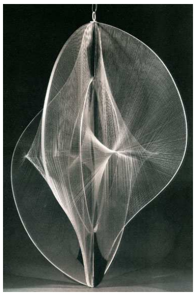
I consider each corner of the (impossible, non-mathematical) cube as a cube itself, with its 8 corners as data points, which yields in total 64 data points. After projection I could connect the appropriate points and make the (impossible) cube. First, I did the erasing and adjusting in Photoshop, but a little later I drew the impossible cube in PS alone, which is not completely general as function of the viewing angles. A bit tedious. The code is too long and too tedious to be included here.

### Gabo's constructive art

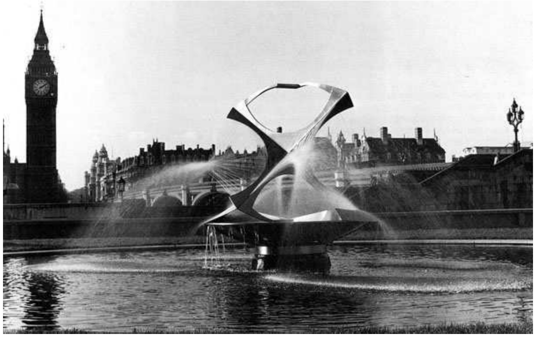
Long ago, I was still a student, I visited the Tate Gallery in London, and was captivated by the constructive art of Gabo,[27] especially his `linearii` of the early 1940-ies.



Naum Gabo
← Lineari
→ Linearii



I also passed by his (temporarily nonworking) fountain in front of the St Thomas hospital opposite the Big Ben, on the other bank of the river Thames.



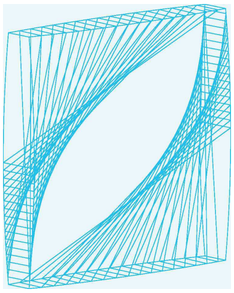
In the fountain, the regular surface is formed by jets of water, and changes dynamically, because it rotates, due to the 'action is reaction' principle.

After many years, I all of a sudden had an idea of how to imitate this fountain in my garden, for my quarter circle pond. A very remote sort of imitation, but... funny. Too alternative to be included here, maybe on the slides for BachoTEX2010, for fun.

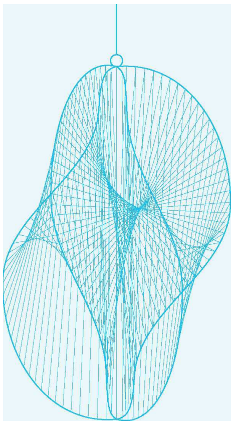
With my youngest daughter I imitated Gabo's `linearii` in plastic.

In ≈1995 I emulated `lineari, linearii` in MF, and adapted the code towards MetaPost. For this conference I looked at the pictures again. Sveta and I adjusted them with thinner lines and colored them blue.

They came out more beautiful than before, even nicer than the photo's of the objects, IMHO. A matter of taste?



Naum Gabo
Lineari



Naum Gabo
Linearii

Of linearii I have included the MP code below

```
beginfig(1);
proofing:=1;
size=75;
path p[];
def pointtopair(expr x,y,z)=
(-x*cosd a + y*sind a,
 -x*sind a * sind b -y*cosd a * sind b
  + z*cosd b)
enddef;
%
%Path construction
%
%basic path (the shape of the boundary)
%can be molded, can be constrained etc
p1:= (0,3size){right}..
  {down}(1.1size,1.75size){down}..
  (.35size,.75size)..(.175size,.375size)..
  {left}origin;
%path with regular---nearly so---
%distributed points
n:=0;%number of points along the curve
p10:= point 0 of p1 hide(n:=n+1)..
   for t:=1 upto 19: hide(n:=n+1)
   point .05t of p1..endfor
 point 1 of p1 hide(n:=n+1)..
   for t:=1 upto 13: hide(n:=n+1)
   point 1+t/14 of p1..endfor
 point 2 of p1 hide(n:=n+1)..
   for t:=1 upto 3: hide(n:=n+1)
   point 2+t/4 of p1..endfor
 point 3 of p1 hide(n:=n+1)..
   for t:=1 upto 3: hide(n:=n+1)
   point 3+t/4 of p1..endfor
   origin;
%viewing angle parameters
b:=-10; a:=60;
%Project the nodes and create
%‘paths in space’ the boundaries
p100:= for k=0 upto n-1:
    pointtopair(0,xpart(point k of p10),
                  ypart(point k of p10))..
    endfor pointtopair(0,0,0);
p200:= for k=0 upto n-1:
    pointtopair(xpart(point k of p10), 0,
                ypart(point k of p10))..
    endfor pointtopair(0,0,0);
p300:= for k=0 upto n-1:
    pointtopair(0,-xpart(point n-k of p10),
          3size-ypart(point n-k of p10))..
    endfor pointtopair(0,0,0);
p400:= for k=0 upto n-1:
    pointtopair(-xpart(point n-k of p10),
     0, 3size-ypart(point n-k of p10))..
    endfor pointtopair(0,0,0);
%
%Drawing
%
%MetaPost approach: black background
%                   and whitedraw
%Black background
fill (-1.5size,-size)--(-1.5size,5size)--
     (1.5size,5size)--(1.5size,-size)--cycle;
%
%Below white drawing
drawoptions(withcolor white);
%
pickup pencircle scaled .5pt;
%Top ring and hang up (rope)
draw point 0 of p100..
     point 0 of p100 + (0,.1size)..cycle;
draw point 0 of p100 + (0,.1size)..
     point 0 of p100 + (0,3size);
%Draw boundary curves
draw p100; draw p200; draw p300; draw p400;
%
%Draw (partially hidden) regular surfaces
pickup pencircle scaled .1pt;
for k=0 step 1 until n:
  draw point k of p200..point n-k of p300;
endfor
for k=0 upto n:
  draw point k of p400..point n-k of p100;
endfor
%erase the ‘hidden’ parts of the lines
%erase fill p100..reverse p200..cycle;
%MetaPost might blackdraw this
%fill p100..reverse p200..cycle
%     withcolor black;
%Front
pickup pencircle scaled .1pt;
draw p100; draw p200;
draw point 0 of p100--origin;
```

```
%
%Draw regular surface which is in sight
for k=0 step 1 until n:
  draw point k of p100..point n-k of p200;
endfor
%Clip to boundary, mod July 2009
clip currentpicture to (-1.5size,-size)--
     (-1.5size,5size)--
     (1.5size,5size)--(1.5size,-size)--
     cycle;
endfig;
end
```

Mathematically, I love the above included regular surfaces due to Gabo, because they are constructed from 1-dimensional data, the bounding curves in 3D. The necessary parameterized projection technique also facilitates animation by changing the viewing angles.

For the first time I emulated the real Gabo by not erasing the 'hidden' lines. In reality they are not hidden, because the object is made of transparent perspex. I lied a bit in the past, because when I did not erase the hidden lines the reverse video picture looked too much blurred by detail. For this conference I fine-tuned the picture with thinner lines and in blue, which looks OK to me.

## Reuse

Sooner or later one arrives at the situation to organize the wealth of macros, pictures, references, tools and ilks for reuse. This gave rise to my BLUe collection. The idea in BLUe is that all the macros you use most of the time are bundled into a blue.tex kernel. The rest is split up into: tools, formats, pictures, references, addresses,... of which BLUe will reuse parts on the fly, unaware of the filing system of the computer. Reuse is a general important aspect, and ipso facto in the lifecycle of document parts.

$$\begin{tikzcd}
\text{Produce} \arrow[r] & \text{Distribute} \arrow[r] & \text{Consume} \arrow[d] \\
\text{reuse} \arrow[u] & \text{retrieve} \arrow[u] \arrow[l] & \text{store} \arrow[l]
\end{tikzcd}$$

With a monolithic collection you have it all, all the time. I decided to adhere to the kernel&modules adage, and to use only from a module what is needed, realized by a selective loading technique, similar to the one of M. Diaz, as mentioned in appendix D of the TEXbook.

One might argue that this economy has become more and more irrelevant, because of the enormous increase of computer speed and memory, since the birth of TEX. Partly true: sometimes parts conflict, e.g. one either formats a report or an article, and in general it is safe to avoid possible conflicts.

To illuminate this note, I have reused pictures be it from pic.dat or from separate PostScript files.

Sometimes reuse implies some extra work.

I also reused the commands included in \loadtocmacros together with \pasteuptoc for a mini-ToC to keep track of the structure while creating this note. En-passant the Contents at the beginning was obtained.

This proceedings submission differs from the pre-proceedings one, because working on the slides gave feedback. The 2.5D GABO's as well as the Escher Cube have earned a place for their own.

An invoke of the one-part \bluepictures followed by one or more \picturenames will load the indicated (TEX) pictures and make them available under their names. At the time I did not construct a library of PostScript pictures, because I did not know how to supply these to \epsfbox. There is no pspic.dat, alas. If time permits I will think it over.

Another aspect is the search path of drivers, if only they looked into TeX input or texmflocal; where to put pspic.dat?

It is not so handy to put the pictures directory in the same place as the main document. I do not know how to add search paths.

If only \pdfTEX could handle PostScript...

As alternative to \bluepictures one can use the underlying two-part macros, sort of environment

```
\beginpictures
   \picturename1
   ...
\endpictures
```

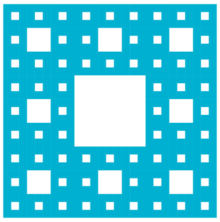
but, alas TEX has no garbage collector, it would not save on memory, it only reduces the possibility of conflicts.

Similar structures hold for tools, formats, references, . . .

In 2002 I worked on macros for the automatic generation of PDF bookmarks for the ((sub)sub)heading titles. It worked, even presented them at the EuroTEX, if I'm not mistaken.

But...

I did not finish the macros, maybe I should. I noticed that in 2005 A. Heck did not provide for bookmarks in his MetaPost note published in MAPS, also available on his WWW site. Is there a need? Rhetorical question.



The above Sierpinski picture was done in TeX with its pseudo filling via rules, also called **black** boxes by Knuth. TeX lacks the filling of an arbitrary closed curve, if not for coloring it.[28] Hyperlinks were also invented long after the birth of TeX. `pdf`TeX makes that all possible.[29] I missed what extras eTeX, or NTS as successors of TeX have brought us. I hope to learn at this conference what LuaTeX is all about.

> For me plain TeX mutual communicating with MetaPost, with PDF as result is sufficient, and even better when PS pictures can be included.

Maybe I can work around it by including PS in MetaPost with SVG or PDF out, or by the multi-step route via `dvi(2)ps`. It is so strange that the world outside has adopted PS and we don't in pdf(Any)TeX.[30]

## 3D metaPost?

It should not be too difficult to extend MetaPost with triples, (x, y, z), in analogy with color, for 3D data. Transformations can then be defined by the 12-tuple $T_x$, $T_y$, $T_z$, $T_{xx}$, $T_{yy}$, $T_{zz}$, $T_{xy}$, $T_{xz}$, $T_{yz}$, $T_{yx}$, $T_{zx}$, $T_{zy}$. In matrix notation

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} + \begin{pmatrix} T_{xx} & T_{yx} & T_{zx} \\ T_{xy} & T_{yy} & T_{zy} \\ T_{xz} & T_{yz} & T_{zz} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

## Conclusions

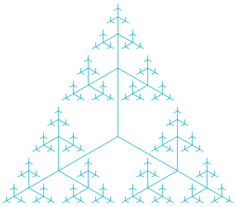
Whatever your tool in computer-assisted typesetting, a Master Class on macro writing in plain TeX and MetaPost is worthwhile, along with discussing tools for similar and complementary tasks to increase awareness, insight and productivity.

A course in literate programming would be great too.

### Wishes

May my turtle graphics macros find a niche, and may my BLUe collection and Paradigms notes be saved from oblivion and kept alive via distribution on the TeX Collection DVD.[31]

May the TeX Collection maintainers, the TeXnicCenter authors, and Jonathan Kew in his TeXworks, support the plain TeX and MetaPost users as well as the LaTeX and ConTeXt users.



### Hopes...

TeX&Co world will

□ Adopt Knuth's Plain & Adobe's PS
□ Adhere to Paradigms: SoC...

*I learned a lot* at and after the conference.

> TeXies, who have been out of it for a while, are well re-educated at an EuroTeX meeting.

LuaTeX and XeTeX are TeX engines, which a.o. provide support for Unicode and OpenType fonts. New OpenType fonts for use in LuaTeX and XeTeX are Latin Modern and TeX Gyre, the latter based on a free counterpart of Adobe's basic set of 35 fonts and the former on Computer Modern. Both aim at greatly increasing the number of diacritical characters in the freely available collections of fonts.[32]

Interesting for me with respect to Cyrillics.

## Acknowledgements

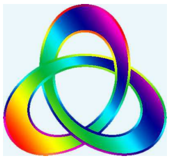
Thank you NTG for inviting me for this conference. Because of this invitation I contributed this paper, refreshed my TeX knowledge, polished my pictures undusted `blue.tex`, and tried to get a modern communicating TeX⇔ MetaPost IDE running on my PC.[33]

I hope you enjoyed reading this paper and if only one idea sparked up in your mind, I am happy. If the walk along history lane, enriched by my perspectives, did not interest you, hopefully the included pictures pleased you.

Thank you Jos Winnink for running the above `linearii.mp` after 10+ years again, as well as the orphan `lineari.mp`, which I also created in MetaFont in 1996, and just before this conference transcribed into MetaPost from memory, since I no longer had the original.[34]

Thank you Taco Hoekwater for procrusting this note into the proceedings, and for your advice in getting me TeXing again.

The `linearii` picture, I consider the most beautiful 2.5D graphics I ever made with TeX, MetaFont, MetaPost, or PostScript, although `lineari` comes close.
The doughnut picture was tricky.

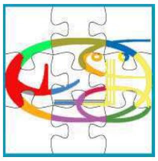


I needed the equation solver of MetaFont, and thanks to my familiarity with splines I could reuse the MetaFont splines in PostScript. Recently, I made in MP a variant, see Appendix II.

The cat picture below is my oldest and my first exercise in MetaFont. I drew it already while at high school. Some years ago, I recast it into a wall sculpture composed of broken mirror pieces.

Cat
← drawing
sculpture →

I also made a puzzle of the cat drawing. Coloring the contours of the cat differently in each piece yielded a difficult, misleading puzzle: one has to concentrate on the drawing and not on the colors, nor the shape of the pieces.



Is this all? No, there is some more, but this is enough for the moment.
My case rests, have fun and all the best.

Zoals
mijn dochters
vroeger
analoog
puzzelden,
zo zullen
mijn kleinkinderen
in de toekomst
electronisch
nog plezier
aan Escher
beleven, en
ongetwijfeld
meer

## Notes

1. Both had courseware: NTG's Advanced TEX course: Insight & Hindsights, respectively MetaFont: practical and impractical applications.
2. The LATEX picture environment and the gkpmac suite I consider outdated, because of the inconsistency when using colors and because it is much better and more convenient to do all your graphics in PS directly or via MetaPost.
3. At the conference I was reminded that BLUe is under copyright. I heard from the LPPL licensing and that seems enough for distributing it on the TEX Live DVD. I agreed with Manfred Lotz to modify the copyright into LPPL, and to look over BLUe for release on TEX Live DVD of 2010. As much as possible, for example the Publishing with TEX guide, can already be released on the 2009 DVD as well as all my notes published in MAPS.
4. Kazimir Malevich, 1878–1935. Russian painter, born in Kiev.
5. At the conference Ulrik Vieth talked about the need for finetuning of math afterwards, especially with respect to a.o. adjusting spacing, aligning subscripts, and adapting for the size of delimiters. In an earlier paper OpenType Math Illuminated, BachoTEX 2009, he details with OpenType Math compared to TEX and MS Cambria release in Word 2007.
6. Hans Hagen reported about the Oriental TEX project, which to me looks like an Oriental mode. Hans confirmed that one can look at it that way.
7. This is an old issue. In the past we had the expression American Screwdriver, meaning using your tool for everything. TEX is not an American Screwdriver.
8. I did not say that one should work in plain or PS all the way. Of course one can start and write macros in whatever high level language is available. I do wish, when you are finished with the macros, that you translate them into plain, respectively PS, and include them in a library for reuse.
9. BachoTEX2009.
10. Courtesy Phil Taylor, who published the macros for doing the calculation by using dimension variables.
11. \author is absent, because in BLUe the author is known, is default, you don't have to fill it in each time, similar holds for affiliation. BLUe is a personalized collection of macros.
12. When as a student I became member 1024 of the Nederlandse Rekenmachine Genootschap, I received The Art of Computer Programming I. My heroes next to Knuth are G. Ploya, G.E. Forsythe, Knuth's predecessor, C. Lanczos, F.L. Bauer, H. Rutishauser, P. Henrici, R.W. Hamming, L. Wall, and H.A. Lauwerier my Dutch applied Math professor. He would have loved my PS Impossible Cube, for sure.
13. Tony Hoare in the 70-ies coined the term in his famous paper.
14. SLC mean Slow, Steep, Strenuous Learning Curve.
15. Since TEX was invented we have witnessed a tremendous development in computers, and how to use computers. The command line interface belongs to the past, we all use computers via GUIs. Why not have a Word-like document preparation system with TEX as open, well-documented kernel, which can be accessed for advanced tasks?
16. Courtesy L. Wall.
17. Because LATEX, ConTEXt, BLUe and ilks have that, of course.
18. I did not say that one could start with the filename, because I consider that against what people are used to, and makes the problem a trifle. The specification was not watertight, because I preferred a more or less open problem, to stimulate creativity.

19. My MetaPost1.005 did not yet provide it.
20. This rotating shrinking squares and a few other pictures, which I borrowed from H.A. Lauwerier's 'Meetkunde met de microcomputer', such as the Koch fractal, the Hilbert curve, the Jura fractal, Escher's knot,. . . and published in MAPS in the mid-90-ies, in my 'Just a little bit of PostScript', 'Graphics & TEX—a reappraisal of Metafont', or 'Tiling in PostScript and Metafont—Escher's wink', I found back, without reference and translated in MetaPost.
21. As yet not! Be aware that the PostScript code is just handed through, not taken notice of.
22. The numeric equation, `u=1cm`, looks a bit strange, looks more like a dimension à la TEX. It becomes clear when you realize that `cm` is just a scaling factor.
23. The problem is that generally I got a picture per page, and I did not know how to trim the picture to its bounding box. After the conference I found out how to trim these pictures in Acrobat 7 professional: select the picture, copy it to the clipboard, and then click create PDF and select From Clipboard Image.
24. The conversion was generally done by opening the `.pdf` pictures in `Photoshop`, trim them and save them as `.jpg`. Later I became aware of the `prologues:=3;` statement, which yields a.o a picture trimmed to the Bounding Box.
25. M.C. Escher, 1898–1972, Dutch artist.
26. I must have the negative somewhere, but can't find it, alas. I'll give it another try.
27. Naum Gabo, 1890–1977. Born Naum Borisovich Pevsner. Bryansk. Russian Constructivist.
28. I colored the picture by post processing in Photoshop. A work flow approach is simpler via the use of the earlier defined `\bluel` and switching back via `\black`.
29. The term hypertext was coined by TeD Nelson during the 1960s, the concept can be traced to Vanneger Bush in 1945. See the Hypertext issue of the CACM july, 1988.
30. I read in the PDF manual the excuse that PDF is a successor of PS, but does not have a programming language built in???
31. This wish will be fulfilled, as reported earlier.
32. Courtesy Bogusłav Jackovski et al. and Ulrik Vieth.
33. Still a wish. But... Wybo installed Ubuntu Linux for me on my laptop, with TEXworks, so I can explore that. Will give me enough work to do.
34. After the conference the NTG discussion list told me how to run the MetaPost utility supplied on TEX Live. Open a new directory, copy `cmd.exe` into it as well as your `filenamme.mp`. Double click the `cmd.exe` icon and type after the prompt `mpost filename.mp`. Another suggestion was to use MetaPost from within the `SciTE` editor. It would have been handy if the `readme`, which comes with the TEX Live, would have contained information on how to use the various utilities. Don't assume a casual user knows it.

Kees van der Laan
Hunzeweg 57, 9893PB Garnwerd, Groningen
kisa1@xs4all.nl

## Appendix I: Contest Solutions

*Phil Taylor* submitted the following mean and lean post conference solution to the Contest, based on what is said on TeXbook p204. He told me that the working of # at the end of the parameter list is little understood, but very useful.

Indeed, I had to look up that functionality and do the replacement to see what it does. A paradigm, though I don't know at the moment where Knuth used it. Phil's solution is a near solution, because curly braces are still needed around the filename.

But...

mean and lean it is, and I reused it already, adapted, for getting a list of all PS pictures used in this paper.

I realized, and use it as an aid in remembering, that the # at the end of the parameter list is a workaround for having a curly opening brace as separator.

```
\def \jpg #1#%
  {\def \next
      {\immediate \pdfximage #1
            {\the \toks 0 .jpg}
       \pdfrefximage \pdflastximage
       }
  \afterassignment \next
  \toks 0 =
  }
%Use
\jpg width 300pt height 30pt {P-Taylor}
```

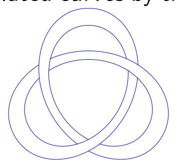
*I* came up with the solution below, which I will add to my FIFO paradigms list in my TeXing Paradigms collection. Of course, I used this one for my purpose.

```
\def\jpgD#1\par{\def\scaling{}
\fifow#1 \wofif          %Sentinels
\pdfximage\scaling\expandafter{\fn}
$$\pdfrefximage\pdflastximage$$}
%
\def\fifow#1 #2{\process{#1}#2
\ifx#2\wofif\expandafter\wofif\fi
\fifow#2}
%
\def\wofif#1\wofif{}
%
\def\process#1#2{%
\ifx#2\wofif\def\fn{#1.jpg}%
\else\edef\scaling{\scaling\space#1}%
\fi}
```

Both solutions circumvent the pitfall of parsing the arguments. The height... and width... if present, are just passed through.

## Appendix II: Escher's knot

The Escher's knot figure, I consider highly instructive. The latest version, in MP, makes use of the symmetry, while the hidden lines are **not** removed: the figure is (re)drawn with only the visible curve pieces. For constructing the pieces it is handy first to draw the complete picture wit `dotlabel` commands included, for the points P, Q, R, and the intersection points a, b, c, d. Construct with the aid of this picture the visible pieces anew from the calculated curves by the use of `cutbefore`.



```
u=5cm;
%Ext points P, Q, R, counter clockwise
pair P, dP;   P:=(0,u);  dP:=( 1,  0);
pair Q, dQ;   Q:= P rotated 120;
pair R, dR;   R:= P rotated 240;
dQ:=(-1,  1.73205);dR:=(-1, -1.73205);
path PQ, QR, RP, pq, qr, rp,
     PQa, PQb, pqa, pqb;%pieces
drawoptions(withcolor blue);
PQ = P{dP}.. .5R{dR}..{dQ}Q;
QR = PQ rotated 120;
RP = PQ rotated 240;
pq = PQ scaled .8;
qr = QR scaled .8;
rp = RP scaled .8;
%draw PQ..QR..RP;%No hidden lines removed
%draw pq..qr..rp;%No hidden lines removed
%Just the pieces instead of
%hidden lines removal
pqa = pq cutafter QR;
pqb = pq cutbefore qr;
draw pqa; draw pqb;
draw pqa rotated 120; draw pqb rotated 120;
draw pqa rotated 240; draw pqb rotated 240;
%a similar approach as above did not work, bug?
PQ:= PQ cutbefore QR;
PQa:= cuttings;
PQb:= PQ cutbefore qr;
draw PQa; draw PQb;
draw PQa rotated 120; draw PQb rotated 120;
draw PQa rotated 240; draw PQb rotated 240;
```

Note the names used: a path PQ is the 'line' between points P and Q. Very handy, this naming convention taken over from good old classical geometry. It facilitates reading of the code. With `cutbefore` and `cutafter` it seems more natural to draw just the visible pieces instead of erasing hidden parts.

# LuaTEX lunatic

And Now for Something Completely Different
– Monty Python, 1972

**Abstract**
luatex lunatic is an extension of the Lua language of luatex to permit embedding of a Python interpreter.
A Python interpreter hosted in luatex allows macro programmers to use all modules from the Python standard library, allows importing of third modules, and permits the use of existing bindings of shared libraries or the creation of new bindings to shared libraries with the Python standard module ctypes.
Some examples of such bindings, particularly in the area of scientific graphics, are presented and discussed.
Intentionally the embedding of interpreter is limited to the python-2.6 release and to a luatex release for the Linux operating system (32 bit).

**Keywords**
Lua, Python, dynamic loading, ffi.

## History

I met luatex sometime around November 2006, and I started to play with it to explore the possibility of typesetting xml documents in an alternative way than the traditional approach based on xsl stylesheet plus xslt processor.

My first intention was to typeset a wikipedia xml dump [4] which is compressed with bzip2; given that I mainly use Python for my programming language, I quickly found `python-bz2` and then Gustavo Niemeyer's "personal laboratory" [15] where I have discovered `Lunatic Python` [14].

To avoid confusion, here `Python` means `CPython`, the C implementation of the Python language [49]. There are other implementations of the Python language: for example Jython [41] and IronPython [37]. According to [6] "the origin of the name (is) based on the television series Monty Python's Flying Circus."

In March 2007 I started to investigate the possibility of integrating Lunatic Python with luatex [57] and in August 2007 I made the first release of luatex-lunatic [20], just around the birth of my second daughter Martina (09/08/07, [33]).

During the 2$^{\text{nd}}$ ConTEXt meeting [21] I found that luatex was stable enough to finalize the project, so I remade all steps and some new examples too (ConTEXt meetings are good places for these kinds of things).

Examples are now hosted at contextgarden [35] while [20] remains for historical purposes.

## Motivations & goals

TEX is synonymous with portability (it's easy to implement/adapt TEX *the program*) and stability (TEX *the language* changes only to fix errors).

We can summarize by saying that "*typesetting in TEX tends to be everywhere everytime*."

These characteristics are a bit unusual in today's scenario of software development: no one is surprised if programs exist only for one single OS (and even for a discontinued OS, given the virtualization technology) and especially no one is surprised at a new release of a program, which actually means bugs fixed and new features implemented (note that the converse is in some sense negative: no release means program discontinued).

Of course, if we consider the *LATEX-system*, i.e. LATEX and its most used packages, this is not frozen at all: just see the near-daily announcements from CTAN. pdfTEX also changes to follow pdf releases.

With `luatex-lunatic` I adopt this point of view: LuaTEX or more specifically LuaTEX & ConTEXt-`mkiv` as a **tool** for publishing content, with some extent to content management. As a tool, it is no surprise if there are "often" (for a TEX user) new releases, given that we can have a LuaTEX update, or a ConTEXt-`mkiv` update, or a Lua update, or a Python update, or a library update for which the Python binding exists; and, of course, if made with no "cum grano salis", no surprise if this can become quickly unmanageable.

The price to pay is the potential **loss of stability**: the same document (with the same fonts and images) processed with a new release can produce a different output.

With regard to portability, the LuaTEX team uses libtool: *GNU Libtool simplifies the developer's job by encapsulating both the platform-specific dependencies, and the user interface, in a single script. GNU Libtool is designed so that the complete functionality of each host type is available via a generic interface, but nasty quirks are hidden from the programmer* [39]), while in Lua and Python support for dynamic loading is a feature of the languages, i.e. there is a (Lua/Python) layer that hides the details of binding.

Thus stated, due to the lack of resources, I have no plan in the immediate future to investigate any OS other than Linux, so this article will cover this OS only; or, stated in another way, there is a potential **loss of portability**.

We can summarize saying that "*typesetting in luatex-lunatic is here and now*", where *here* stands for "a specific OS" and *now* for "with this release". Actually *here* stands for "Linux 32 bit", and *now* stands for `luatex -snapshot-0.42.0.tar.bz2` with ConTEXt-`mkiv` current 2008.07.17; probably both will already be old by the time this is printed.

Another motivation has emerged during the development of luatex-lunatic: the possibility to use ConTEXt-`mkiv` as a sort of literate programming tool for a specific context.

It is well known that CWEB is a way to tangle together a program written in a *specific* programming language (C) with its documentation written with a macro markup language, TEX; luatex-lunatic and ConTEXt-`mkiv` can be used to tangle together a program written in an (almost) *arbitrary* programming language with its documentation written with a *high level* macro markup language, ConTEXt-`mkiv`.

Put in another way: currently an application calls TEX or LATEX (i.e. it creates a process) to obtain a result from a tex snippet (for example to show a math formula); instead luatex-lunatic with ConTEXt-`mkiv` calls the application by dynamic loading (i.e. it does not create a process) to obtain the result to insert into tex source.

For example one can use luatex-lunatic ConTEXt-`mkiv` to typeset a math formula, and the binding for the evaluation of the same formula (there are several symbolic-math Python modules already available).

We will see more about this later, when we will talk of Sage.

We want to find the smallest set of patches of the luatex codebase, or, better, we want to avoid:

1. constraints of any sort to the development team;
2. massive modifications of the code base;
3. radical modification of the building process.

## Lunatic Python

There is no better site than [14] to explain what is Lunatic Python:

Lunatic Python is a two-way bridge between Python and Lua, allowing these languages to intercommunicate. Being two-way means that it allows Lua inside Python, Python inside Lua, Lua inside Python inside Lua, Python inside Lua inside Python, and so on.

...

The bridging mechanism consists of creating the missing interpreter state inside the host interpreter. That is, when you run the bridging system inside Python, a Lua interpreter is created; when you run the system inside Lua, a Python interpreter is created.

Once both interpreter states are available, these interpreters are provided with the necessary tools to interact freely with each other. The given tools offer not only the ability of executing statements inside the alien interpreter, but also to acquire individual objects and interact with them inside the native state. This magic is done by two special object types, which act by bridging native object access to the alien interpreter state.

Almost every object which is passed between Python and Lua is encapsulated in the language specific bridging object type. The only types which are not encapsulated are strings and numbers, which are converted to the native equivalent objects.

Besides that, the Lua side also has special treatment for encapsulated Python functions and methods. The most obvious way to implement calling of Python objects inside the Lua interpreter is to implement a `__call` function in the bridging object metatable. Unfortunately this mechanism is not supported in certain situations, since some places test if the object type is a function, which is not the case of the bridging object. To overwhelm these problems, Python functions and methods are automatically converted to native Lua function closures, becoming accessible in every Lua context. Callable object instances which are not functions nor methods, on the other hand, will still use the metatable mechanism. Luckily, they may also be converted in a native function closure using the asfunc() function, if necessary.

According to [68], page 47, a *closure* is "a function plus all it needs to access non-local variables correctly"; a non-local variable "is neither a global variable nor a local variable". For example consider `newCounter`:

```
function newCounter()
 local i = 0
 return function()
         i = i+1
         return i
        end
end
c1 = newCounter()
print(c1()) --> 1
print(c1()) --> 2
c2 = newCounter()
print(c2()) --> 1
print(c1()) --> 3
print(c2()) --> 2
```

`i` is a non-local variable; we see that there is no interference between `c1` and `c2`—they are two different closures over the same function.

It’s better to track a layout of installation of luatex-lunatic on a Linux box.
Let’s set up a home directory:
`HOMEDIR=/opt/luatex/luatex-lunatic`
Next:

1. download and install python-2.6.1 (at least) from [49]. Assuming `$HOMEDIR/Python-2.6.1` as build directory, let’s configure python-2.6.1 with

   ```
   ./configure
        --prefix=/opt/luatex/luatex-lunatic
        --enable-unicode=ucs4
        --enable-shared
   ```

   and install it. After installation we should end in a “Filesystem Hierarchy Standard”-like Filesystem (cf. [46], except for `Python-2.6.1`), i.e. something like this:

   ```
   $> cd $HOMEDIR && ls -1X
   bin
   include
   lib
   man
   share
   Python-2.6.1
   ```

   It’s also convenient to extend the system path:

   ```
   $> export PATH=
            /opt/luatex/lunatic-python/bin:$PATH
   ```

   so we will use the python interpreter in `$HOMEDIR`.
2. download luatex source code from [43]; we will use `luatex-snapshot-0.42.0`, so let’s unpack it in `$HOMEDIR/luatex-snapshot-0.42.0` . For uniformity, make a symbolic link

   ```
   $> cd  $HOMEDIR
   $> ln -s luatex-snapshot-0.42.0 luatex
   ```

   It’s convenient to have a stable ConTEXt minimals distribution installed (cf. [23]) under `$HOMEDIR`, i.e. `$HOMEDIR/minimals`, so that we will replace its `luatex` with our `luatex-lunatic`. Remember to set up the environment with

   ```
   $> . $HOMEDIR/minimals/tex/setuptex
   ```

   We don’t build it now, because `build.sh` needs to be patched.
3. download luatex-lunatic from [3], revision 7, and put it in `lunatic-python`, i.e.

   ```
   $> cd $HOMEDIR
   $> bzr branch lp:lunatic-python
   ```

   We must modify `setup.py` to match luatex installation (here "<" stands for the original `setup.py`, ">" stands for the modified one; it’s a diff file):

   ```
   1c1
   < #!/usr/bin/python
   ---
   > #!/opt/luatex/luatex-lunatic/bin/python
   14,16c14,16
   < LUALIBS = ["lua5.1"]
   < LUALIBDIR = []
   < LUAINCDIR = glob.glob("/usr/include/lua*")
   ---
   > LUALIBS = ["lua51"]
   > LUALIBDIR = ['/opt/luatex/
      luatex-lunatic/
      luatex/build/texk/web2c']
   > LUAINCDIR = glob.glob("../
      luatex/source/texk/web2c/luatexdir/lua51*")
   48a49
   >
   ```

When we build lunatic-python, we will end with a `python.so` shared object that will be installed in the `$HOMEDIR/lib/python2.6/site-packages` directory, so it’s convenient to prepare a `python.lua` wrapper like this one:

```
loaded = false
func = package.loadlib(
"/opt/luatex/luatex-lunatic/lib/python2.6/
site-packages/python.so","luaopen_python")
 if func then
        func()
        return
 end
if not loaded then
        error("unable to find python module")
end
```

Before building, we must resolve the dynamic loading problem; again from [14]

> . . . Unlike Python, Lua has no default path to its modules. Thus, the default path of the real Lua module of Lunatic Python is together with the Python module, and a `python.lua` stub is provided. This stub must be placed in a path accessible by the Lua require() mechanism, and once imported it will locate the real module and load it.
>
> Unfortunately, there’s a minor inconvenience for our purposes regarding the Lua system which imports external shared objects. The hardcoded behavior of the

loadlib() function is to load shared objects without exporting their symbols. This is usually not a problem in the Lua world, but we're going a little beyond their usual requirements here. We're loading the Python interpreter as a shared object, and the Python interpreter may load its own external modules which are compiled as shared objects as well, and these will want to link back to the symbols in the Python interpreter. Luckily, fixing this problem is easier than explaining the problem. It's just a matter of replacing the flag RTLD_NOW in the loadlib.c file of the Lua distribution by the or'ed version RTLD_NOW|RTLD_GLOBAL. This will avoid "undefined symbol" errors which could eventually happen.

Modifying luatex/source/texk/web2c/
luatexdir/lua51/loadlib.c
is not difficult:

```
69c69
< void *lib = dlopen(path, RTLD_NOW);
---
> void *lib = dlopen(path, RTLD_NOW|RTLD_GLOBAL);
```

(again "<" means original and ">" means modified).

According to dlopen(3) - Linux man page (see for example [18]),

The function dlopen() loads the dynamic library file named by the null-terminated string filename and returns an opaque "handle" for the dynamic library. If filename is NULL, then the returned handle is for the main program. If filename contains a slash ("/"), then it is interpreted as a (relative or absolute) pathname. Otherwise, the dynamic linker searches for the library as follows (see ld.so(8) for further details):

□ (ELF only) If the executable file for the calling program contains a DT_RPATH tag, and does not contain a DT_RUNPATH tag, then the directories listed in the DT_RPATH tag are searched.
□ If the environment variable LD_LIBRARY_PATH is defined to contain a colon-separated list of directories, then these are searched. (As a security measure this variable is ignored for set-user-ID and set-group-ID programs.)
□ (ELF only) If the executable file for the calling program contains a DT_RUNPATH tag, then the directories listed in that tag are searched.
□ The cache file /etc/ld.so.cache (maintained by ldconfig(8)) is checked to see whether it contains an entry for filename.
□ The directories /lib and /usr/lib are searched (in that order).

If the library has dependencies on other shared libraries, then these are also automatically loaded by the dynamic linker using the same rules. (This process may occur recursively, if those libraries in turn have dependencies, and so on.)

One of the following two values must be included in flag:

□ RTLD_LAZY
Perform lazy binding. Only resolve symbols as the code that references them is executed. If the symbol is never referenced, then it is never resolved. (Lazy binding is only performed for function references; references to variables are always immediately bound when the library is loaded.)
□ RTLD_NOW
If this value is specified, or the environment variable LD_BIND_NOW is set to a non-empty string, all undefined symbols in the library are resolved before dlopen() returns. If this cannot be done, an error is returned.

Zero or more of the following values may also be ORed in flag:

□ RTLD_GLOBAL
The symbols defined by this library will be made available for symbol resolution of subsequently loaded libraries.
□ RTLD_LOCAL
This is the converse of RTLD_GLOBAL, and the default if neither flag is specified. Symbols defined in this library are not made available to resolve references in subsequently loaded libraries.
□ RTLD_NODELETE (since glibc 2.2)
Do not unload the library during dlclose(). Consequently, the library's static variables are not reinitialised if the library is reloaded with dlopen() at a later time. This flag is not specified in POSIX.1-2001.
□ RTLD_NOLOAD (since glibc 2.2)
Don't load the library. This can be used to test if the library is already resident (dlopen() returns NULL if it is not, or the library's handle if it is resident). This flag can also be used to promote the flags on a library that is already loaded. For example, a library that was previously loaded with RTLD_LOCAL can be re-opened with RTLD_NOLOAD | RTLD_GLOBAL. This flag is not specified in POSIX.1-2001.
□ RTLD_DEEPBIND (since glibc 2.3.4)
Place the lookup scope of the symbols in this library ahead of the global scope. This means that a self-contained library will use its own symbols in preference to global symbols with the same name contained in libraries that have already been loaded. This flag is not specified in POSIX.1-2001.

If filename is a NULL pointer, then the returned handle is for the main program. When given to dlsym(), this handle causes a search for a symbol in the main program, followed by all shared libraries loaded at program startup, and then all shared libraries loaded by dlopen() with the flag RTLD_GLOBAL.

External references in the library are resolved using the libraries in that library's dependency list and any other libraries previously opened with the RTLD_GLOBAL flag. If the executable was linked with the flag "-rdynamic" (or, synonymously, "–export-dynamic"), then the global symbols in the executable will also be used to resolve references in a dynamically loaded library.

If the same library is loaded again with dlopen(), the same file handle is returned. The dl library maintains reference counts for library handles, so a dynamic library is not deallocated until dlclose() has been called on it as many times as dlopen() has succeeded on it. The _init routine, if present, is only called once. But a subsequent call with RTLD_NOW may force symbol resolution for a library earlier loaded with RTLD_LAZY.

If dlopen() fails for any reason, it returns NULL.

Nevertheless this is not enough: reference manual [66] says (page 23):

*Dynamic loading of* .so *and* .dll *files is disabled on all platforms.*

So we must “enable” it and we must ensure that the `luatex` executable is linked against `libdl.so` because this contains the `dlopen()` symbol; also we must ensure that all the Lua functions involved in a `dlopen()` call must be resolved in the `luatex-lunatic` executable.

Assuming that we are always in `$HOMEDIR`, we must modify
`source/texk/web2c/luatexdir/am/liblua51.am`
and `source/texk/web2c/Makefile.in` .
For
`source/texk/web2c/luatexdir/am/liblua51.am`:

```
12c12
< liblua51_a_CPPFLAGS += -DLUA_USE_POSIX
---
> liblua51_a_CPPFLAGS += -DLUA_USE_LINUX
```

while for `source/texk/web2c/Makefile.in`:

```
98c98
< @MINGW32_FALSE@am__append_14 = -DLUA_USE_POSIX
---
> @MINGW32_FALSE@am__append_14 = -DLUA_USE_LINUX
1674c1674
<   $(CXXLINK) $(luatex_OBJECTS) $(luatex_LDADD)
$(LIBS)
---
>   $(CXXLINK) $(luatex_OBJECTS) $(luatex_LDADD)
$(LIBS) -Wl,-E -uluaL_openlibs -fvisibility=hidd
en -fvisibility-inlines-hidden -ldl
```

The last patch is the most important, so let’s examine it more closely. Essentially, we are modifying the linking phase of building process of `luatex` (switch `-Wl,-E`) by adding `libdl` (switch `-ldl`) because `libdl` contains the symbol `dlopen` as stated before.

The switch `-uluaL_openlibs` tells the linker to consider the symbol `luaL_openlibs` even if it’s not necessary for building luatex-lunatic. In fact `luaL_openlibs` is coded in `lunatic-python/src/luainpython.c` and it needs to be resolved at runtime only when luatex-lunatic wants to load the Python interpreter.

So, even if `luaL_openlibs` is a function coded in `$HOMEDIR/luatex/source/texk/web2c/luatexdir/lua51/linit.c`, it’s not used by luatex, so the linker discards this symbol because it’s useless.

According to `ld(1)`:

□ -u symbol
> Force symbol to be entered in the output file as an undefined symbol. Doing this may, for example, trigger linking of additional modules from standard libraries. -u may be repeated with different option arguments to enter additional undefined symbols. This option is equivalent to the “EXTERN” linker script command.

It’s possible to examine how symbols are resolved runtime by setting `LD_DEBUG=all`; for example

```
$> export LD_DEBUG=all;
$> luatex python.lua &>python.LD_DEBUG;
$> export LD_DEBUG=
```

Here we are assuming a correct final luatex lunatic `luatex` and the `python.lua` wrapper seen before.
The file `python.LD_DEBUG` will show something like this:

```
3736: symbol=luaL_openlibs;
        lookup in file=./luatex-lunatic [0]
3736: binding file /opt/luatex/luatex-lunatic/
       lib/python2.6/site-packages/python.so [0]
       to ./luatex-lunatic [0]:
        normal symbol ‘luaL_openlibs’
```

Without the `-uluaL_openlibs` linker flag, we will see something like this:

```
4033: symbol=luaL_openlibs;
     lookup in file=./luatex-lunatic-0.42.0.-test [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/tls/i686/cmov/libm.so.6 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/tls/i686/cmov/libdl.so.2 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/libreadline.so.5 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/libhistory.so.5 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/libncurses.so.5 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/tls/i686/cmov/libc.so.6 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/ld-linux.so.2 [0]
4033: symbol=luaL_openlibs;
lookup in file=/opt/luatex/luatex-lunatic/lib/
  python2.6/site-packages/python.so [0]
4033: symbol=luaL_openlibs;
 lookup in file=/lib/tls/i686/cmov/libpthread.so.0 [0]
4033: symbol=luaL_openlibs;
    lookup in file=/lib/tls/i686/cmov/libutil.so.1 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/tls/i686/cmov/libc.so.6 [0]
4033: symbol=luaL_openlibs;
      lookup in file=/lib/ld-linux.so.2 [0]
4033: /opt/luatex/luatex-lunatic/lib/python2.6/
      site-packages/python.so:
     error: symbol lookup error:
     undefined symbol: luaL_openlibs (fatal)
4033:
4033: file=/opt/luatex/luatex-lunatic/lib/python2.6/
     site-packages/python.so [0]; destroying link map
```

And near the bottom we can see this error: `symbol lookup error: undefined symbol: luaL_openlibs (fatal)`.

The last two switches, namely `-fvisibility=hidden` and `-fvisibility-inlines-hidden`, are gcc switches (not linker switches) and again they are related with symbols, more precisely with symbols collisions. Consider this: in `$HOMEDIR/luatex/source/libs/libpng` there is a libpng library (currently vers. 1.2.38). This library, once compiled, will be merged by the linker into the `luatex` executable, and hence into the `luatex-lunatic` executable too. Now, we can build a Python binding to another libpng library or, better, we can import a Python module (e.g. PythonMagickWand, an interface to ImageMagick®, see [40]) that has a binding to its own libpng library. In this situation, at runtime the dynamic loader will resolve for the Python module the symbols of libpng from luatex libpng, instead of those from its own libpng. Now, we cannot guarantee that these two libraries are the same, because we cannot replace the libpng of luatex (see near the end of the preceding section "Motivation & goals") and, of course, we cannot replace the libpng library from the Python module with the one from luatex, because the last one can be patched for luatex only. So, we have symbols collisions (see [9]): almost for sure, a symbol collision will cause a segmentation fault, and the program abort.

More information about this can be found starting from the already cited [9], especially [69]. A good text is also [63].

A solution can be this: "hide" to the "outside" all symbols that aren't necessary for dynamic loading of shared objects. For standard `luatex`, this means "hide all": for `luatex-lunatic`, this means "hide all but not symbols from lua", otherwise we will not be able to use `loadlib`. It's not so difficult to "hide all": just patch the `build.sh` script of luatex sources by adding

```
28a29,36
> CFLAGS="-g -O2 -Wno-write-strings
           -fvisibility=hidden"
> CXXFLAGS="$CFLAGS
            -fvisibility-inlines-hidden"
> export CFLAGS
> export CXXFLAGS
```

The hardest part is to "unhide" the Lua part. We can proceed in this manner: collect the result of the patched `build.sh` in an out file:

```
$> cd $HOMEDIR/luatex; ./build.sh &> out
```

Then locate in out *all* the lines about Lua and remove the `-fvisibility=hidden` flag: for example

```
gcc -DHAVE_CONFIG_H -I.
-I../../../source/texk/web2c -I./..
-I/opt/luatex/luatex-lunatic/
      luatex-snapshot-0.42.0/build/texk
-I/opt/luatex/luatex-lunatic/
      luatex-snapshot-0.42.0/source/texk
-I../../../source/texk/web2c/luatexdir/lua51
-DLUA_USE_LINUX -g -O2
-Wno-write-strings
-fvisibility=hidden
-Wdeclaration-after-statement
-MT liblua51_a-lapi.o
-MD -MP -MF .deps/liblua51_a-lapi.Tpo
-c -o liblua51_a-lapi.o
‘test -f
’luatexdir/lua51/lapi.c’
|| echo
’../../../source/texk/web2c/’‘
 luatexdir/lua51/lapi.c
mv -f .deps/liblua51_a-lapi.Tpo
        .deps/liblua51_a-lapi.Po
```

will become

```
gcc -DHAVE_CONFIG_H -I.
-I../../../source/texk/web2c -I./..
-I/opt/luatex/luatex-lunatic/
      luatex-snapshot-0.42.0/build/texk
-I/opt/luatex/luatex-lunatic/
      luatex-snapshot-0.42.0/source/texk
-I../../../source/texk/web2c/luatexdir/lua51
-DLUA_USE_LINUX
-g -O2 -Wno-write-strings
-Wdeclaration-after-statement
-MT liblua51_a-lapi.o
-MD -MP -MF .deps/liblua51_a-lapi.Tpo
-c -o liblua51_a-lapi.o
‘test -f
’luatexdir/lua51/lapi.c’
|| echo
’../../../source/texk/web2c/
’‘luatexdir/lua51/lapi.c
mv -f .deps/liblua51_a-lapi.Tpo
 .deps/liblua51_a-lapi.Po
```

After that, recompile luatex

```
/bin/bash ./libtool
--tag=CXX
--mode=link
./CXXLD.sh -g -O2
-Wno-write-strings
-fvisibility=hidden
-fvisibility-inlines-hidden
-o luatex
luatex-luatex.o
libluatex.a libff.a
libluamisc.a libzzip.a
libluasocket.a liblua51.a
/opt/luatex/luatex-lunatic/
```

```
   luatex-snapshot-0.42.0/build/libs/
    libpng/libpng.a
/opt/luatex/luatex-lunatic/
   luatex-snapshot-0.42.0/build/libs/
    zlib/libz.a
/opt/luatex/luatex-lunatic/
   luatex-snapshot-0.42.0/build/libs/
    xpdf/libxpdf.a
/opt/luatex/luatex-lunatic/
   luatex-snapshot-0.42.0/build/libs/
    obsdcompat/libopenbsd-compat.a
libmd5.a libmplib.a
lib/lib.a
/opt/luatex/luatex-lunatic/
   luatex-snapshot-0.42.0/build/texk/
    kpathsea/libkpathsea.la
-lm -Wl,-E
-uluaL_openlibs
-fvisibility=hidden
-fvisibility-inlines-hidden
-ldl
```

Of course it's better to edit a `trick.sh` from `out` (see [34]) that will do all the work, paying the price of ~20 minutes of cut and paste for every new luatex release for preparing this trick file.

After executing `$HOMEDIR/luatex/trick.sh` we will have an *unstripped* `luatex` binary in `$HOMEDIR/luatex/build/texk/web2c` so we are ready for the final step. It's better not to strip it, because we can track problems more easily.

4. we copy `luatex` into the `bin` directory of ConTEXt minimals and remade formats:

   ```
   $> cp $HOMEDIR/luatex/build/texk/web2c/luatex
   $HOMEDIR/minimals/tex/texmf-linux/bin
   $> context --make
   ```

   And in the end we must build the lunatic-python shared object:

   ```
   $> cp $HOMEDIR/lunatic-python
   $> python setup.py build && python setup.py
   install
   ```

We can now make a simple test; let's save this in `test.tex`:

```
\directlua{require "python";
sys = python.import("sys");
tex.print(tostring(sys.version_info))}
\bye
```

Next let's run callgrind, a tool of valgrind (see [30]), to generate a *call graph* [5]:

```
$> valgrind --tool=callgrind
    --callgrind-out-file=test-%p.callgrind
    --dump-instr=yes
    luatex --fmt=plain --output-format=pdf test.tex
```

To see and analyze this call graph we can use kcachegrind [13]: see appendix at page 53 for the graph centered at `main` function, with Min. node cost=1% , Min. call cost=1% .

## Examples

### Image processing

*ImageMagick.* `ImageMagick` is "*a software suite to create, edit, and compose bitmap images. It can read, convert and write images in a variety of formats (over 100) including DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PhotoCD, PNG, PostScript, SVG, and TIFF. Use ImageMagick to translate, flip, mirror, rotate, scale, shear and transform images, adjust image colors, apply various special effects, or draw text, lines, polygons, ellipses and Bézier curves.*" (See [40].) There are two bindings in Python, and we choose the `PythonMagickWand` [48], a ctypes-based wrapper for ImageMagick.

According to [50] ctypes is a foreign function library for Python. It provides C compatible data types, and allows calling functions in DLLs or shared libraries. It can be used to wrap these libraries in pure Python. ctypes is included in Python.

This simple script create a 200×200 pixel image at 300dpi with a shadow:

```
import PythonMagickWand as pmw
pmw.MagickWandGenesis()
wand = pmw.NewMagickWand()
background = pmw.NewPixelWand(0)
pmw.MagickNewImage(wand,200,200,background)
pmw.MagickSetImageResolution(wand,118.110,118.110)
pmw.MagickSetImageUnits(wand,
            pmw.PixelsPerCentimeterResolution)
pmw.MagickShadowImage(wand,90,3,2,2)
pmw.MagickWriteImage(wand,"out.png")
```

i.e., something like this:

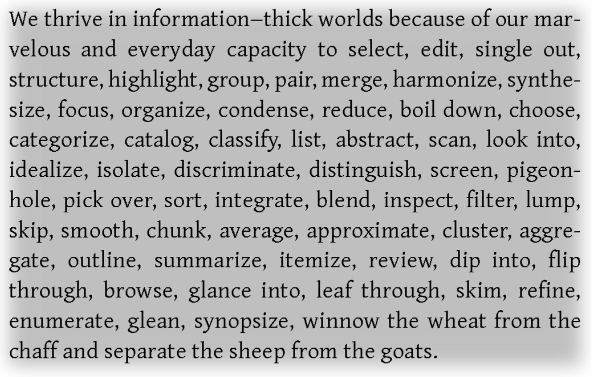
Suppose we want to use it to generate a background for text, i.e.

```
\startShadowtext%
\input tufte
\stopShadowtext%
```

Let’s now look at luatex lunatic and ConTEXt-mkiv in action for the first time:

```
\usetypescriptfile[type-gentium]
\usetypescript[gentium]
\setupbodyfont[gentium,10pt]
\setuppapersize[A6][A6]
\setuplayout[height=middle,topspace=1cm,
  header={2\lineheight},footer=0pt,backspace=1cm,
  margin=1cm,width=middle]
%%
%% lua layer
%%
\startluacode
function testimagemagick(box,t)
  local w
  local h
  local d
  local f
  local res = 118.11023622047244094488 -- 300 dpi
  local opacity = 25
  local sigma = 15
  local x = 10
  local y = 10
  w = math.floor((tex.wd[box]/65536 )
            /72.27*2.54*res)
  h = math.floor(((tex.ht[box]/65536)+
               (tex.dp[box]/65536))
            /72.27*2.54*res)
  f = string.format("%s.png",t)
  --
  -- Call the python interpreter
  --
  require("python")
  pmw = python.import("PythonMagickWand")
  wand = pmw.NewMagickWand()
  background = pmw.NewPixelWand(0)
  pmw.MagickNewImage(wand,w,h,background)
  pmw.MagickSetImageResolution(wand,res,res)
  pmw.MagickSetImageUnits(wand,
       pmw.PixelsPerCentimeterResolution)
  pmw.MagickShadowImage(wand,opacity,sigma,x,y)
  pmw.MagickWriteImage(wand ,f)
end
\stopluacode
%%
%% TeX layer
%%
\def\testimagemagick[#1]{%
\getparameters[Imgk][#1]%
\ctxlua{%
  testimagemagick(\csname Imgkbox\endcsname,
             "\csname Imgkfilename\endcsname")}%
}
%%
%% ConTeXt layer
%%
\newcount\shdw
\long\def\startShadowtext#1\stopShadowtext{%
\bgroup%
\setbox0=\vbox{#1}%
\testimagemagick[box=0,
                filename={shd-\the\shdw}]%
\defineoverlay[backg]%
        [{\externalfigure[shd-\the\shdw.png]}]%
\framed[background=backg,
        frame=off,offset=4pt]{\box0}%
\global\advance\shdw by 1%
\egroup%
}
\starttext
\startTEXpage%
\startShadowtext%
\input tufte
\stopShadowtext%
\stopTEXpage
\stoptext
```

As we can see, there is an almost one-to-one mapping between Python code and Lua code, a good thing for a small script.
And here is the result:



We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

What about symbols collisions?

```
$> eu-readelf --all luatex &> luatex.dump
$> export LD_DEBUG=all;context test-imagemagick.tex &> test-
imagemagick.tex.LD_DEBUG; export LD_DEBUG=
```

If we search png_memcpy_check which is coded in `$HOMEDIR/source/libs/libpng/libpng-1.2.38/pngmem.c` of luatex, we will find that it’s bound to system libpng:

```
25749: symbol=png_memcpy_check;
  lookup in file=luatex [0]

25749: symbol=png_memcpy_check;
  lookup in file=/lib/tls/i686/cmov/libm.so.6 [0]

25749: symbol=png_memcpy_check;
  lookup in file=/lib/tls/i686/cmov/libdl.so.2 [0]
:
: (62 lines after)
:
25749: symbol=png_memcpy_check;
  lookup in file=/usr/lib/libpng12.so.0 [0]

25749: binding file /usr/lib/libpng12.so.0 [0]
to /usr/lib/libpng12.so.0 [0]:
normal symbol ‘png_memcpy_check’ [PNG12_0]
```

In fact if we search for png_memcpy_check in luatex.dump we see that it’s hidden now:

```
Symbol table [40] ’.symtab’ contains 10087 entries:
 9592 local symbols  String table: [41] ’.strtab’
:
Num:    Value   Size Type
4837: 082022b0   27 FUNC

Bind   Vis     Ndx Name
LOCAL  HIDDEN  13  png_memcpy_check
:
```

As a counterexample, suppose that we don’t use hidden flags, so now png_memcpy_check is visible:

```
Num:    Value   Size Type
2273: 08243050     27 FUNC

Bind   Vis     Ndx Name
GLOBAL DEFAULT 13  png_memcpy_check
```

Now we have a fatal error:

```
$> export LD_DEBUG=all;context test-imagemagick.tex &> test-
imagemagick.tex.LD_DEBUG; export LD_DEBUG=
:
MTXrun | fatal error, no return code, message: luatex: execu-
tion interrupted
:
```

and we see that png_memcpy_check is resolved in luatex:

```
24213: symbol=png_memcpy_check;
       lookup in file=luatex [0]
24213: binding file /usr/lib/libpng12.so.0 [0]
       to luatex [0]:
       normal symbol ‘png_memcpy_check’ [PNG12_0]
```

so we have symbols collisions. In this case it can be hard to track the guilty symbol; even in this case the fatal error can be given by another symbols collision, not necessarily png_memcpy_check. Also note that this code

```
\starttext
\externalfigure[out.png]
\stoptext
```

compiles right—of course, because there is no PythonImagickWand involved and so no symbols collisions. So this kind of error can become a nightmare.

Let’s continue with our gallery.

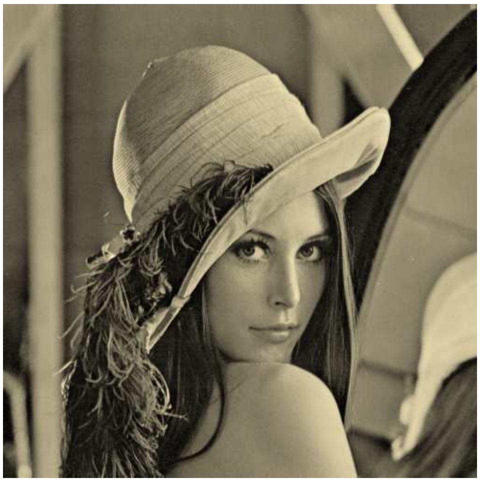
*PIL – PythonImageLibrary.* PIL (see [51]) is similar to ImageMagick, but at least for png doesn’t require libpng, so we are safe from symbol collisions.

```
\startluacode
function testPIL(imageorig,imagesepia)
  require(”python”)
  PIL_Image = python.import(”PIL.Image”)
  PIL_ImageOps = python.import(”PIL.ImageOps”)
  python.execute([[
def make_linear_ramp(white):
    ramp = []
    r, g, b = white
    for i in range(255):
        ramp.extend((r*i/255, g*i/255, b*i/255))
    return ramp
]])
    -- make sepia ramp
    -- (tweak color as necessary)
    sepia = python.eval
           (”make_linear_ramp((255, 240, 192))”)
    im = PIL_Image.open(imageorig)

    -- convert to grayscale
    if not(im.mode == ”L”)
     then
        im = im.convert(”L”)
    end
    --  optional: apply contrast
    -- enhancement here, e.g.
    im = PIL_ImageOps.autocontrast(im)
    --  apply sepia palette
    im.putpalette(sepia)
    --  convert back to RGB
    --  so we can save it as JPEG
    --  (alternatively, save it in PNG or similar)
    im = im.convert(”RGB”)
    im.save(imagesepia)
end
\stopluacode

\def\SepiaImage#1#2{%
\ctxlua{testPIL(”#1”,”#2”)}%
\startcombination[1*2]
{\externalfigure[#1][width=512pt]}{\ss Orig.}
{\externalfigure[#2][width=512pt]}{\ss Sepia}
\stopcombination
}
```

```
\starttext
\startTEXpage
%\SepiaImage{lena.jpg}{lena-sepia.jpg}
\SepiaImage{lena.png}{lena-sepia.png}
\stopTEXpage
\stoptext
```

Here is the result (sorry, Lena is too nice to show her only in black and white):

Orig.



Sepia

The code shows how to define a Python function inside a Lua function and how to call it. Note that we must respect the Python indentation rules, so we can use the multiline string of Lua `[[..]]`.

### Language adapter

Suppose we have a `C` library for a format of a file (i.e. TIFF, PostScript) that we want to manage in the same way as png, pdf, jpeg and jbig. One solution is to build a quick binding with ctypes of Python, and then import it in luatex-lunatic as a traditional Lua module. As an example, let's consider `ghostscript` [10], here in vers. 8.64. It can be compiled as a shared library, and building a `testgs.py` (see [35]`#Ghostscript`) binding is not difficult (see file `base/gslib.c` in source distribution). The key here is to build a binding that fits our needs, not a general one.

```
\startluacode
function testgs(epsin,pdfout)
  require("python")
  gsmodule = python.import("testgs")
  ghost = gsmodule.gs()
  ghost.appendargs('-q')
  ghost.appendargs('-dNOPAUSE')
  ghost.appendargs('-dEPSCrop')
  ghost.appendargs('-sDEVICE=pdfwrite')
  ghost.InFile = epsin
  ghost.OutFile = pdfout
  ghost.run()
end
\stopluacode
```

```
\def\epstopdf#1#2{\ctxlua{testgs("#1","#2")}}
\def\EPSfigure[#1]{%lazy way to load eps
\epstopdf{#1.eps}{#1.pdf}%
\externalfigure[#1.pdf]}
```

```
\starttext
\startTEXpage
{\EPSfigure[golfer]}
{\ss golfer.eps}
\stopTEXpage
\stoptext
```

Here is the result:

golfer.eps

We can also use PostScript libraries: for example `barcode.ps` [56], a PostScript barcode library:

```
\startluacode
function epstopdf(epsin,pdfout)
  require("python")
  gsmodule = python.import("testgs")
  ghost = gsmodule.gs()
  ghost.appendargs('-q')
  ghost.appendargs('-dNOPAUSE')
  ghost.appendargs('-dEPSCrop')
  ghost.appendargs('-sDEVICE=pdfwrite')
  ghost.InFile = epsin
  ghost.OutFile = pdfout
  ghost.run()
end

function barcode(text,type,options,savefile)
  require("python")
  gsmodule = python.import("testgs")
  barcode_string =
    string.format("%%!\n100 100 moveto (%s) (%s)
%s barcode showpage",
                  text,options,type)
  psfile = string.format("%s.ps",savefile)
  epsfile = string.format("%s.eps",savefile)
  pdffile = string.format("%s.pdf",savefile)
  temp = io.open(psfile,'w')
  print(psfile)
  temp:write(tostring(barcode_string),"\n")
  temp:flush()
  io.close(temp)
  ghost = gsmodule.gs()
  ghost.rawappendargs('-q')
  ghost.rawappendargs('-dNOPAUSE')
  ghost.rawappendargs('-sDEVICE=epswrite')
  ghost.rawappendargs(
       string.format('-sOutputFile=%s',epsfile))
  ghost.rawappendargs('barcode.ps')
  ghost.InFile= psfile
  ghost.run()
end
\stopluacode

\def\epstopdf#1#2{\ctxlua{epstopdf("#1","#2")}}
\def\EPSfigure[#1]{%lazy way to load eps
\epstopdf{#1.eps}{#1.pdf}%
\externalfigure[#1.pdf]%
}

\def\PutBarcode[#1]{%
\getparameters[bc][#1]%
\ctxlua{barcode("\csname bctext\endcsname",
             "\csname bctype\endcsname",
             "\csname bcoptions\endcsname",
             "\csname bcsavefile\endcsname" )}%
\expanded{\EPSfigure
                [\csname bcsavefile\endcsname]}%
}

\starttext
\startTEXpage
{\PutBarcode[text={CODE 39},type={code39},
             options={includecheck includetext},
             savefile={TEMP1}]}\\
{\ss code39}
\blank
{\PutBarcode[text={CONTEXT},type={code93},
            options={includecheck includetext},
            savefile={TEMP2}]}\\

{\ss code93}
\blank
{\PutBarcode[text={977147396801},type={ean13},
             options={includetext},
             savefile={TEMP3}]}\\
{\ss ean13}
\stopTEXpage
\stoptext
```

Of course one can implement a direct conversion into ps->pdf, instead of ps->eps->pdf.
Here is the result:

For a beautiful book on PostScript see [58] (and its site [42]) and also [2].

### Scientific & math extensions

*Sage.* "*Sage is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface. Mission: Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab.*" [53]

Given that Sage comes with its own Python interpreter, we must rebuild lunatic-python and adapt python.lua accordingly; also sage is a command line program, so we need a stub sagestub.py:

```
from sage.all_cmdline import *
```

Here is the ConTEXt-mkiv code:

```
\startluacode
function test_ode(graphout)
  require("python")
  pg = python.globals()
  SAGESTUB = python.import("sagestub")
  sage = SAGESTUB.sage
  python.execute([[
def f_1(t,y,params):
   return[y[1],
         -y[0]-params[0]*y[1]*(y[0]**2-1)]
]])
python.execute([[
def j_1(t,y,params):
    return [ [0,1.0],
             [-2.0*params[0]*y[0]*y[1]-1.0,
             -params[0]*(y[0]*y[0]-1.0)], [0,0]]
]])
```

```
  T=sage.gsl.ode.ode_solver()
  T.algorithm="rk8pd"
  f_1 = pg.f_1
  j_1 = pg.j_1
  pg.T=T
  python.execute("T.function=f_1")
  T.jacobian=j_1
  python.execute("T.ode_solve(y_0=[1,0],
                   t_span=[0,100],
                   params=[10],num_points=1000)")
  python.execute(string.format(
                 "T.plot_solution(filename='%s')",
                  graphout ))
end
\stopluacode
```

```
\def\TestODE#1{%
\ctxlua{test_ode("#1")}%
\startcombination[1*2]
{%
\vbox{\hsize=8cm
Consider solving the Van der Pol oscillator
$x''(t) +ux'(t)(x(t)^2-1)+x(t)=0 $
between $t=0$ and $t= 100$.
As a first order system it is
$x'=y$
$y'=-x+uy(1-x^2)$
Let us take $u=10$ and use
initial conditions $(x,y)=(1,0)$ and use the
\emphsl{\hbox{Runge-Kutta} \hbox{Prince-Dormand}}
algorithm.
}%
}{\ss \ }
{\externalfigure[#1][width=9cm]}{\ss Result
for 1000 points}}
```

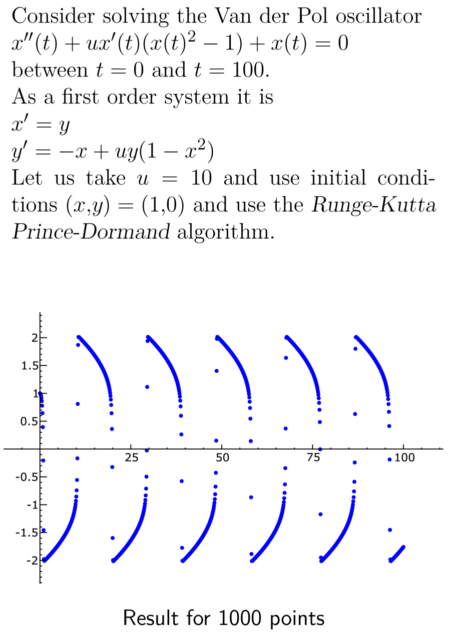
```
\starttext
\startTEXpage
\TestODE{ode1.pdf}
\stopTEXpage
\stoptext
```

As we can see, here we use the `python.globals()` Lua function to communicate between the Python interpreter and Lua, and this can generate a bit of useless redundancy.

*R.* R "*is a free software environment for statistical computing and graphics*" [52]. It has its own language, but there is also a Python binding, rpy2 [27], that we install in our `$HOMEDIR`.

It can be necessary to add these env. variabless

```
$>export R_HOME=/opt/luatex/luatex-lunatic/lib/R
$>export LD_PRELOAD=/opt/luatex/
          luatex-lunatic/lib/R/lib/libR.so
```



Consider solving the Van der Pol oscillator
$x''(t) + ux'(t)(x(t)^2 - 1) + x(t) = 0$
between $t = 0$ and $t = 100$.
As a first order system it is
$x' = y$
$y' = -x + uy(1 - x^2)$
Let us take $u = 10$ and use initial conditions $(x,y) = (1,0)$ and use the *Runge-Kutta Prince-Dormand* algorithm.

Result for 1000 points

Figure 1. Result of the Sage code, with sage-3.2.3-pentiumM-ubuntu32bit-i686-Linux

For R we split the Python side in Lua in a pure Python script `test-R.py`:

```
import rpy2.robjects as robjects
import rpy2.rinterface as rinterface
class density(object):
   def __init__(self,samples,outpdf,w,h,kernel):
     self.samples = samples
     self.outpdf= outpdf
     self.kernel = kernel
     self.width=w
     self.height=h
   def run(self):
       r = robjects.r
       data = [int(k.strip())
          for k in
             file(self.samples,'r').readlines()
             ]
       x = robjects.IntVector(data)
       r.pdf(file=self.outpdf,
             width=self.width,
             height=self.height)
       z = r.density(x,kernel=self.kernel)
       r.plot(z[0],z[1],xlab='',ylab='')
       r['dev.off']()
if __name__ == '__main__' :
   dens =
    density('u-random-int','test-001.pdf',10,7,'o')
   dens.run()
```

and import this into Lua:

```
\startluacode
function testR(samples,outpdf,w,h,kernel)
  require("python")
  pyR  = python.import("test-R")
  dens =
   pyR.density(samples,outpdf,w,h,kernel)
  dens.run()
end
\stopluacode
```

```
\def\plotdenstiy[#1]{%
\getparameters[R][#1]%
\expanded{\ctxlua{testR("\Rsamples",
                        "\Routpdf",
                        \Rwidth,
                        \Rheight,"\Rkernel")}}}
```

```
\setupbodyfont[sans,14pt]
\starttext
\startTEXpage
\plotdenstiy[samples={u-random-int},
                      outpdf={test-001.pdf},
                      width={10},height={7},
                      kernel={o}]
\setupcombinations[location=top]
\startcombination[1*2]
{\vbox{\hsize=400bp
This is a density plot of around {\tt 100 000}
random numbers between
$0$ and $2^{16}-1$ generated
from {\tt \hbox{/dev/urandom}}}}{}
{\externalfigure[test-001.pdf][width={400bp}]}{}
\stopcombination
\stopTEXpage
\stoptext
```

Note the conditional statement
`if __name__ == '__main__' :`
that permits to test the script with an ordinary Python interpreter.

It's worth noting that rpy2 is included in Sage too.

For more information about scientific computation with Python see [61] and [62] (also with site [31]) and [54].

The example of Sage shows that in this case we can think of luatex lunatic as an *extension* of Sage but also that luatex lunatic is *extended* with Sage. This is somehow similar to CWEB: code and description are tangled together, but now there's not a specific language like C in CWEB (in fact we can do the same with R). Eventually "untangled" is a matter of separation of Python code in a different file from the source tex file.

This is a density plot of around 100 000 random numbers between $0$ and $2^{16}-1$ generated from /dev/urandom

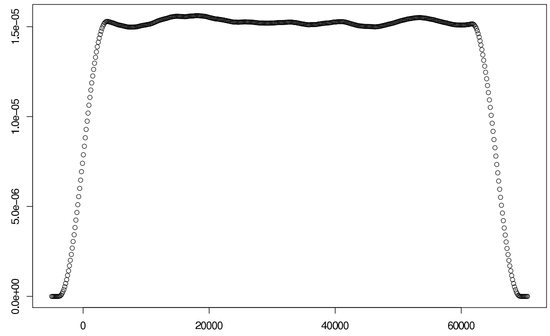


Figure 2. Result of the R code

By the way, it's important to underline that CWEB is more advanced than other (C) code documentation systems because it embeds source code inside descriptive text rather than the reverse (as is common practice in most programming languages). Documentation can be not "linear", a bit unusual for ordinary programmers, but it's a more efficient and effective description of complex systems. Here we are talking about "linear" documentation, much like this article: a linear sequence of text-and-code printed as they appear.

Of course some computations may require much more time to be completed than the time of generation of the respective document (and ConTeXt is also a multipass system), so this approach is pretty inefficient—we need a set of macros that take care of intermediate results, i.e. *caching macros* or *multipass macros*. For example, in ConTeXt-mkiv we can say

```
\doifmode{*first}{%
   % this code will be executed only at first pass
   \Mymacro
}
```

so `\Mymacro` will be executed only at the first pass; there is also a `Two Pass Data` module `core-two.mkiv` that can be used for this, but don't forget that we also have Lua and Python for our needs.

**Graphs**

In LuaTeX–ConTeXt-mkiv we already have a very powerful tool for technical drawing: MetaPost. Simple searching reveals for example METAGRAPH [32] or the more generic LuaGRAPH [19], a Lua binding to graphviz [38] with output also in MetaPost; both are capable of drawing (un)directed graphs and/or networks. The next two modules are more oriented to graph calculations.

MetaPost is an example of "embedding" an interpreter in LuaTEX at compilation time (see luaopen_mplib(L) in void luainterpreter(void) in $HOMEDIR/source/texk/web2c/luatexdir/lua/luastuff.c). So hosting Python is not a new idea: the difference is that the "embedding" is done at run time.

*igraph.* igraph "*is a free software package for creating and manipulating undirected and directed graphs. It includes implementations for classic graph theory problems like minimum spanning trees and network flow, and also implements algorithms for some recent network analysis methods, like community structure search. The efficient implementation of igraph allows it to handle graphs with millions of vertices and edges. The rule of thumb is that if your graph fits into the physical memory then igraph can handle it.* [11]

To install igraph we must first install pycairo, a Python binding to cairo [1], a well known 2D graphics library: so we gain another tool for generic drawing.

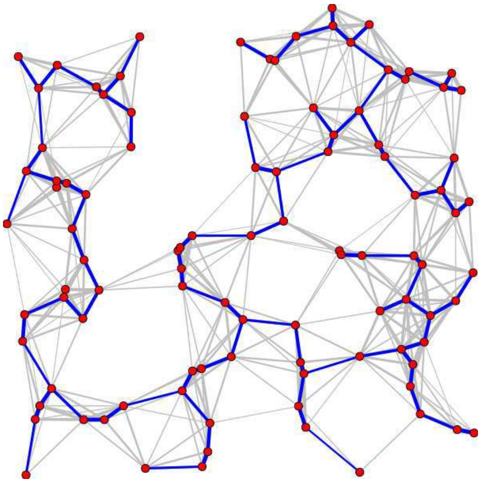


Figure 3. The result of the igraph code.

This time we coded the Python layer as a class:

```
import igraph
class spanningtree(object) :
 def __init__(self,ofn):
  self.ofn = ofn
 def  distance(self,p1, p2):
  return ((p1[0]-p2[0]) ** 2
       + (p1[1]-p2[1]) ** 2) ** 0.5
 def plotimage(self):
  res = igraph.Graph.GRG(100,
              0.2, return_coordinates=True)
  g = res[0]
  xs = res[1]
  ys = res[2]
  layout = igraph.Layout(zip(xs, ys))
  weights = [self.distance(layout[edge.source],
          layout[edge.target]) for edge in g.es]
  max_weight = max(weights)
  g.es["width"] = \
  [6 - 5*weight/max_weight for weight in weights]
  mst = g.spanning_tree(weights)

  fig = igraph.Plot(target=self.ofn)
  fig.add(g, layout=layout,
          opacity=0.25,
          vertex_label=None)
  fig.add(mst,
          layout=layout,
          edge_color="blue",
          vertex_label=None)
  fig.save()
if __name__ == '__main__':
    sp = spanningtree('test-igraph.png')
    sp.plotimage()
```

In this case we calculate a minimum spanning tree of a graph, and save the result in test-igraph.png. The Lua layer is so simple that it is encapsulated in a TEX macro:

```
\def\PlotSpanTree#1{%
\startluacode
require("python")
local spantree_module
local sp
spantree_module =
  python.import("test-igraph")
sp = spantree_module.spanningtree("#1")
sp.plotimage()
\stopluacode
\externalfigure[#1]}
\starttext
\startTEXpage
\PlotSpanTree{test-igraph.png}
\stopTEXpage
\stoptext
```

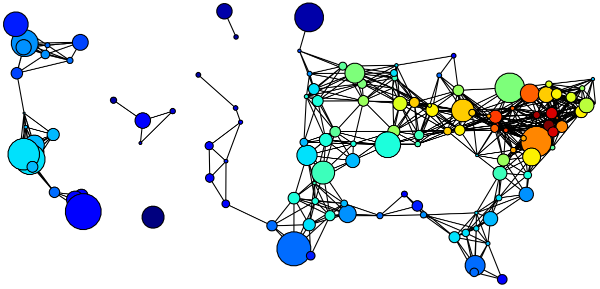
*NetworkX.* NetworkX *is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.* [22]

The code is simpler: we have only two layers: the Python layer, and the TEX layer. The Python layer is a trivial modification of knuth_miles.py (see [24], [36], [60]), and is left to the reader (hint: rename ..__init__.. in def run()).

```
\starttext
\startTEXpage
\ctxlua{require("python");
knuth=python.import("test-networkx");
knuth.run();}
```

```
\externalfigure[knuth_miles]
\stopTEXpage
\stoptext
```

Here is the result (with a bit of imagination, one can see the USA):



*ROOT.* ROOT *is an object-oriented program and library developed by* CERN. *It was originally designed for particle physics data analysis and contains several features specific to this field.* [7], [26]

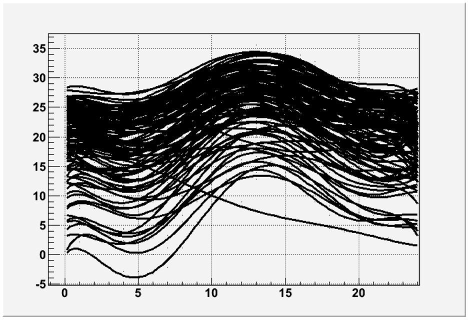
In this example we will draw 110 lines from file `data` (each line being 24 float values separated by spaces); each line will be a curve to fit with a polynomial of degree 6. We isolate all relevant parts in a Python script `test-ROOT1.py`:

```
from ROOT import TCanvas,
  TGraph,TGraphErrors,TMultiGraph
from ROOT import gROOT
from math import sin
from array import array
def run(filename):
  c1 = TCanvas("c1","multigraph",200,10,700,500)
  c1.SetGrid()
  mg = TMultiGraph()
  n = 24; x = array('d',range(24))
  data = file('data').readlines()
  for line in data:
    line = line.strip()
    y  = array('d',
      [float(d) for d in line.split()])
    gr =  TGraph(n,x,y)
    gr.Fit("pol6","q")
    mg.Add(gr)
  mg.Draw("ap")
  c1.Update(); c1.Print(filename)
```

This is the ConTEXt side:

```
\startluacode
function test_ROOT(filename)
  require("python")
  test = python.import('test-ROOT1')
  test.run(filename)
end
\stopluacode
\starttext \startTEXpage
\ctxlua{test_ROOT("data.pdf")}
\rotate[rotation=90]{\externalfigure[data.pdf]}
\stopTEXpage \stoptext
```

Here is the result:



### Database

*Oracle Berkeley DB XML.* Oracle Berkeley DB XML "*is an open source, embeddable xml database with XQuery-based access to documents stored in containers and indexed based on their content. Oracle Berkeley DB XML is built on top of Oracle Berkeley DB and inherits its rich features and attributes*" [45];

We take as our data source a `Wikiversity XML` dump [8], more specifically
`enwikiversity-20090627-pages-articles.xml`,
a ~95MByte uncompressed xml file (in some sense, we end were we started).

Building a database is not trivial, so one can see [35] under `Build_the_container` for details. The most important things are indexes; here we use

```
container.addIndex("","title",
     "edge-element-substring-string",uc)
container.addIndex("","username",
     "edge-element-substring-string",uc)
container.addIndex("","text",
     "edge-element-substring-string",uc)
```

These indexes will be used for substring queries, but not for regular expressions, for which it will be used the much slower standard way. Again it's better to isolate the Python code in a specific module, `wikidbxml_queryTxn.py` (see [35] under `Make pdf` for details). This module does the most important work: translate from a 'MediaWiki-format' to ConTEXt-`mkiv`. A 'MediaWiki-format' is basically made by <page> like this:

```
<page>
<title>Wikiversity:What is Wikiversity?</title>
```

```
<id>6</id>
<revision>
<id>445629</id>
<timestamp>2009-06-08T06:30:15Z</timestamp>
<contributor>
<username>Jr.duboc</username>
<id>138341</id>
</contributor>
<comment>/* Wikiversity for teaching */</comment>
<text xml:space="preserve">{{policy|[[WV:IS]]}}
{{about wikiversity}}
[[Image:Plato i sin akademi,
av Carl Johan Wahlbom
(ur Svenska Familj-Journalen).png
|thumb|left|300px|Collaboration between students
and teachers.]]
__TOC__
==Wikiversity is a learning community==
[[Wikiversity]] is a community effort to learn
and facilitate others'
learning. You can use Wikiversity to find
information or ask questions about a subject you
need to find out more about. You can also use it
to share your knowledge about a subject,
and to build learning
materials around that knowledge.
:
&lt;!-- That's all, folks! --&gt;
</text>
</revision>
</page>
```

So, a <page> is an xml document with non-xml markup in <text> node (which is an unfortunate tag name for an xml document); even if <page> is simple, parsing <text> content, or, more exactly, the text node of <text> node, is not trivial, and we can:

- □ implement a custom parser using the lpeg module of ConTEXt-mkiv (e.g. $HOMEDIR/minimals/tex /texmf-context/tex/context/base/lxml-tab.lua); this can be a good choice, because we can translate 'MediaWiki-format' directly into ConTEXt markup, but of course we must start from scratch;
- □ use an external tool, like the Python module mwlib: MediaWiki parser and utility library [25].

We choose mwlib (here in vers. 0.11.2) and implement the translation in two steps:

1. from 'MediaWiki-format' to XML-DocBook (more exactly DocBook RELAX NG grammar 4.4; see [44])
2. from XML-DocBook to ConTEXt-mkiv (this is done by the getConTeXt(title,res) function)

Actually, the wikidbxml_queryTxn.writeres() function writes the result of the query by calling wikidbxml_queryTxn.getArticleByTitle() which in turn calls wikidbxml_queryTxn.getConTeXt() function.

The ConTEXt-mkiv side is (for the moment forget about the functions listtitles(title) and simplereports(title)):

```
\usetypescriptfile[type-gentium]
\usetypescript[gentium]
\setupbodyfont[gentium,10pt]
\setuppapersize[A5][A5]
\setuplayout[height=middle,
topspace=1cm,header={2\lineheight},
footer=0pt,backspace=1cm,margin=1cm,
width=middle]
%%
%% DB XML
%%
\startluacode
function testdbxml(title,preamble,
                  postamble,filename)
  require("python")
  pg = python.globals()
  wikiversity =
    python.import("wikidbxml_queryTxn")
  wikiversity.writeres(title,preamble,
                       postamble,filename)
end
\stopluacode
%%
%% sqlite
%%
\startluacode
function listtitles(title)
  require("python")
  pg = python.globals()
  wikiversity =
    python.import("wikidbxml_queryTxn")
  r = wikiversity.querycategory(title)
  local j = 0
  local res = r[j] or {}
  while res do
   local d =
    string.format("\%s\\par",
       string.gsub(tostring(res),'_',' '))
   tex.sprint(tex.ctxcatcodes,d)
   j = j+1
   res = r[j]
  end
end
\stopluacode
%%
%% sqlite
```

```
%%
\startluacode
function simplereports(title)
  require("python")
  pg = python.globals()
  wikiversity =
       python.import("wikidbxml_queryTxn")
  r = wikiversity.simplereports(title)
  local j = tonumber(r)
  for v = 0,j-1 do
   local d =
       string.format("\\input reps\%04d ",v)
   tex.sprint(tex.ctxcatcodes,d)
  end
  print( j )
end
\stopluacode
%% ConTeXt
\def\testdbxml[#1]{%
\getparameters[dbxml][#1]%
\ctxlua{%
testdbxml("\csname dbxmltitle\endcsname",
          "\csname dbxmlpreamble\endcsname",
          "\csname dbxmlpostamble\endcsname",
          "\csname dbxmlfilename\endcsname")}%
\input \csname dbxmlfilename\endcsname %
}
\starttext
\testdbxml[title={Primary mathematics/Numbers},
           preamble={},
           postamble={},
           filename={testres.tex}]
\stoptext
```

Here we query for the exact title `Primary mathematics/Numbers`: for the result, see page 55.

*sqlite.* Python offers adapters for practically all well known databases like PostgreSQL, MySQL, ZODB, etc. ("*ZODB is a persistence system for Python objects*" written in Python, see [16]. ZODB is the heart of Plone [47], a popular content management system also written in Python), but here we conclude with `sqlite`, a "*software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. The source code for SQLite is in the public domain*" (see [29]).

`sqlite` is a module of the standard Python library, and we can use it to query a `category.db` for titles. (`category.db` is a db made from `enwikiversity-20090627-category.sql`, which is a MySQL dump. Conversion is not difficult and is not shown here.)

The code uses the two functions seen before, `listtitles` and `simplereports`:

```
\starttext
{\bfb Query for  'geometr':}
\ctxlua{listtitles("geometr")}%
\ctxlua{simplereports("geometr")}%
\stoptext
```

See p. 57 for a short result (actually the first page of second hit, `Geometry`. The complete document is 12 pages).

## MetaTEX

What is MetaTEX?
Quoting from `$HOMEDIR/tex/texmf-context/tex/context/base/metatex.tex`:

*This format is just a minimal layer on top of the LuaTEX engine and will not provide high level functionality. It can be used as basis for dedicated (specialized) macro packages.*

*A format is generated with the command:*
*`luatools --make --compile metatex`*

It should be clear from previous examples that a system with *all* these "bindings" becomes quickly unmanageable: one can spend almost all available time upgrading to latest releases. Just as an example: already at time of preprinting `ghostscript` was at rel. 8.70 (vs. rel. 8.64 of this article) `Sage` was at rel. 4.1 (vs. rel. 3.2.2), Python itself was at rel. 2.6.2 (vs. rel. 2.6.1) and there even exists rel. 3.1 . . . .

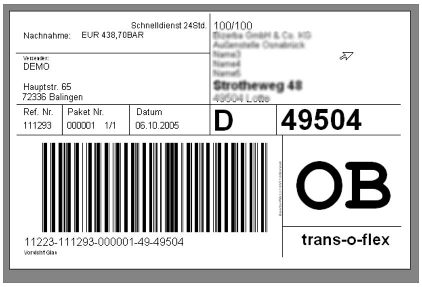
Also not all Python packages are "robust": for example, in the file `docbookwriter.py` of `mwlib` we can see

```
Generate DocBook from the DOM tree generated
by the parser.
Currently this is just a proof of concept
which is very incomplete
```

(and of course, mwlib was at rel. 0.12.2 (vs. rel. 0.11.2) so this message may have disappeared as well).

So, in the end, it's better to have more distinct "tools" than one big tool that does anything and badly. We can see now why MetaTEX fits well in this line: it permits to create the exact "tool" needed and luatex lunatic can be used to complete this tool. For example, consider the problem of typesetting labels like the one on top if the next page.

Basically, it's a table with barcode and two or three fonts (preferably monospaced fonts), most of the time black and white. ConTEXt-`mkiv` already comes with natural tables, or even a layer mechanism (see [70]); `luatex-lunatic` with `barcode.ps` provides the barcode. We don't need colors, interaction, indexes, sectioning.



Schnelldienst 24Std.
Nachnahme: EUR 438,70BAR

Versender:
DEMO

Hauptstr. 65
72336 Balingen

100/100
Bizerba GmbH & Co. KG
Außenstelle Osnabrück
Name3
Name4
Name5
Strotheweg 48
49504 Lotte

| Ref. Nr. | Paket Nr. | Datum | | |
|---|---|---|---|---|
| 111293 | 000001 1/1 | 06.10.2005 | D | 49504 |

11223-111293-000001-49-49504
Vorsicht Glas

OB

trans-o-flex

Financial reports are similar: here we can benefit from the decimal Python module that is included in the standard library (decimal is an implementation of Decimal fixed point and floating point arithmetic; see [28]).

MetaTEX can be used to produce very specific formats for educational purposes: think for example of a MetaTEXSage, or a MetaTEXR from the CWEB point of view, i.e. embedded source code inside descriptive text rather than the reverse.

Also, Python can be used as a query language for Plone (mentioned previously), a powerful CMS written in Python, so it can be possible to print specific content type without translating it into an intermediate form like xml (and maybe in the future the reverse too, i.e. push a content type made by a MetaTEXPlone).

## Conclusion

LuaTEX with ConTEXt-mkiv is a powerful tool for publishing content, and with an embedded Python interpreter we unquestionably gain more power, especially when MetaTEX becomes stabilized. If one wants, one can also experiment with JPype *"an effort to allow Python programs full access to Java class libraries. This is achieved not through re-implementing Python, as Jython/JPython has done, but rather through interfacing at the native level in both virtual machines"* [12] (currently unstable under Linux).

So it's better here to emphasize "the dark side of the moon".

First, it should be clear that currently we cannot assure **stability** and **portability** in the TEX sense.

Moreover, under Linux there is immediately a price to pay: symbol collisions. Even if the solution presented here should ensure that there are no symbol collisions between luatex and an external library, it doesn't resolve problems of collision between symbols of two *external* libraries; installing all packages under a folder /opt/luatex/luatex-lunatic can help to track this problem, but it's not a definitive solution. Of course, we avoid this problem if we use pure Python libraries, but these tend to be slower than C libraries.

ctypes looks fascinating, but a binding in ctypes is usually not easy to build; we must not forget that Lua offers its loadlib that can always be used as an alternative to ctypes or to any other Python alternative like SWIG [55] which can, anyway, build wrapper code for Lua too, at least from development release 1.3. In the end, an existing Python binding is a good choice if it is stable, rich, complete and mature with respect to an existing Lua binding, or if there is not a Lua binding.

For a small script, coding in Lua is not much different from coding in Python; but if we have complex objects, things can be more complicated: for example this Python code

```
z = x*np.exp(-x**2-y**2)
```

is translated in this not-so-beatiful Lua code

```
z=x.__mul__(np.exp((x.__pow__(2).
          __add__(y.__pow__(2))).__neg__()))
```

(see [35]#Scipy). It is better to separate the Python layer into an external file, so we can eventually end in a *py,*lua,*tex for the same job, adding complexity to manage.

In the end, note that a Python interpreter does not "complete" in any sense luatex, because Lua is a perfect choice: it's small, stable, and OS-aware. Conversely, Python is bigger, and today we are seeing Python versions 2.4, 2.5, 2.6.2, 2.7 alpha, 3.1 ... not exactly a stable language from a TEX user point of view.

## Acknowledgements

The author would like to thank Taco Hoekwater and Hans Hagen for their suggestions, help and encouragement during the development and the writing process of this article.

The author is also immensely grateful to Massimiliano "Max" Dominici for his strong support, help and encouragement.

## References

*All links were verified between 2009.08.17 and 2009.08.21.*

[1] http://cairographics.org
[2] http://cg.scs.carleton.ca/~luc/PSgeneral.html
[3] https://code.launchpad.net/~niemeyer/lunatic-python/trunk
[4] http://download.wikimedia.org
[5] http://en.wikipedia.org/wiki/Call_graphs
[6] http://en.wikipedia.org/wiki/Python_(programming_language)
[7] http://en.wikipedia.org/wiki/ROOT

[8] http://en.wikiversity.org/wiki/Getting_stats_out_of_Wikiversity_XML_dumps

[9] http://gcc.gnu.org/wiki/Visibility

[10] http://ghostscript.com/

[11] http://igraph.sourceforge.net

[12] http://jpype.sourceforge.net/

[13] http://kcachegrind.sourceforge.net/cgi-bin/show.cgi

[14] http://labix.org/lunatic-python

[15] http://labix.org/python-bz2

[16] https://launchpad.net/zodb

[17] http://linux.die.net/man/1/ld

[18] http://linux.die.net/man/3/dlopen

[19] http://luagraph.luaforge.net/graph.html

[20] http://luatex.bluwiki.com/go/User:Luigi.scarso

[21] http://meeting.contextgarden.net/2008

[22] http://networkx.lanl.gov

[23] http://minimals.contextgarden.net/

[24] http://networkx.lanl.gov/examples/drawing/knuth_miles.html

[25] http://pypi.python.org/pypi/mwlib

[26] http://root.cern.ch

[27] http://rpy.sourceforge.net/

[28] http://speleotrove.com/decimal

[29] http://sqlite.org/

[30] http://valgrind.org/

[31] http://vefur.simula.no/intro-programming/

[32] http://vigna.dsi.unimi.it/metagraph

[33] http://wiki.contextgarden.net/Future_ConTeXt_Users

[34] http://wiki.contextgarden.net/Image:Trick.zip

[35] http://wiki.contextgarden.net/User:Luigi.scarso/luatex_lunatic

[36] http://www-cs-faculty.stanford.edu/~knuth/sgb.html

[37] http://www.codeplex.com/IronPython

[38] http://www.graphviz.org

[39] http://www.gnu.org/software/libtool

[40] http://www.imagemagick.org/script/index.php

[41] http://www.jython.org

[42] http://www.math.ubc.ca/~cass/graphics/text/www/index.html

[43] http://www.luatex.org

[44] http://www.oasis-open.org/docbook

[45] http://www.oracle.com/database/berkeley-db/xml/index.html

[46] http://www.pathname.com/fhs/

[47] http://www.plone.org

[48] http://www.procoders.net/?p=39

[49] http://www.python.org

[50] http://www.python.org/doc/2.6.1/library/ctypes.html

[51] http://www.pythonware.com/products/pil/

[52] http://www.r-project.org/

[53] http://www.sagemath.org/

[54] http://www.scipy.org/

[55] http://www.swig.org

[56] http://www.terryburton.co.uk/barcodewriter/

[57] private email with Taco Hoekwater

[58] Bill Casselman, *Mathematical Illustrations: A Manual of Geometry and PostScript.* ISBN-10: 0521547881, ISBN-13: 9780521547888 Available at site http://www.math.ubc.ca/~cass/graphics/text/www/index.html

[59] Danny Brian, *The Definitive Guide to Berkeley DB XML.* Apress, 2006. ISBN-13: 978-1-59059-666-1

[60] Donald E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing.* ACM Press, New York, 1993. ISBN 978-0-470-75805-2

[61] Hans Petter Langtangen, *A Primer on Scientific Programming with Python.* Springer, 2009. ISBN: 978-3-642-02474-0

[62] Hans Petter Langtangen, *Python Scripting for Computational Science.* Springer, 2009. ISBN: 978-3-540-73915-9

[63] John Levine, *Linkers & Loaders.* Morgan Kaufmann Publisher, 2000. ISBN-13: 978-1-55860-496-4

[64] Mark Lutz, *Learning Python, Fourth Edition.* O'Reilly, September 2009 (est.) ISBN-10: 0-596-15806-8, ISBN 13: 978-0-596-15806-4

[65] Mark Lutz, *Programming Python, Third Edition.* O'Reilly, August 2006. ISBN-10: 0-596-00925-9, ISBN 13: 978-596-00925-0

[66] luatexref-t.pdf. Available in `manual` folder of `luatex-snapshot-0.42.0.tar.bz2`

[67] Priscilla Walmsley, *XQuery.* O'Reilly, April 2007. ISBN-13: 978-0-596-00634-1

[68] Roberto Ierusalimschy, *Programming in Lua (second edition).* Lua.org, March 2006. ISBN 85-903798-2-5

[69] Ulrich Drepper, *How to Write Shared Libraries.* http://people.redhat.com/drepper/dsohowto.pdf

[70] Willi Egger, ConTeXt: Positioning design elements at specific places on a page (tutorial). EuroTeX 2009 & 3rd ConTeXt Meeting

[71] Yosef Cohen and Jeremiah Cohen, *Statistic and Data with R.* Wiley 2008. ISBN 978-0-470-75805-2

> I currently use Ubuntu Linux, on a standalone laptop—it has no Internet connection. I occasionally carry flash memory drives between this machine and the Macs that I use for network surfing and graphics; but I trust my family jewels only to Linux.
> — Donald Knuth
> Interview with Donald Knuth
> By Donald E. Knuth and Andrew Binstock
> Apr. 25, 2008
> http://www.informit.com/articles/article.aspx?p=1193856

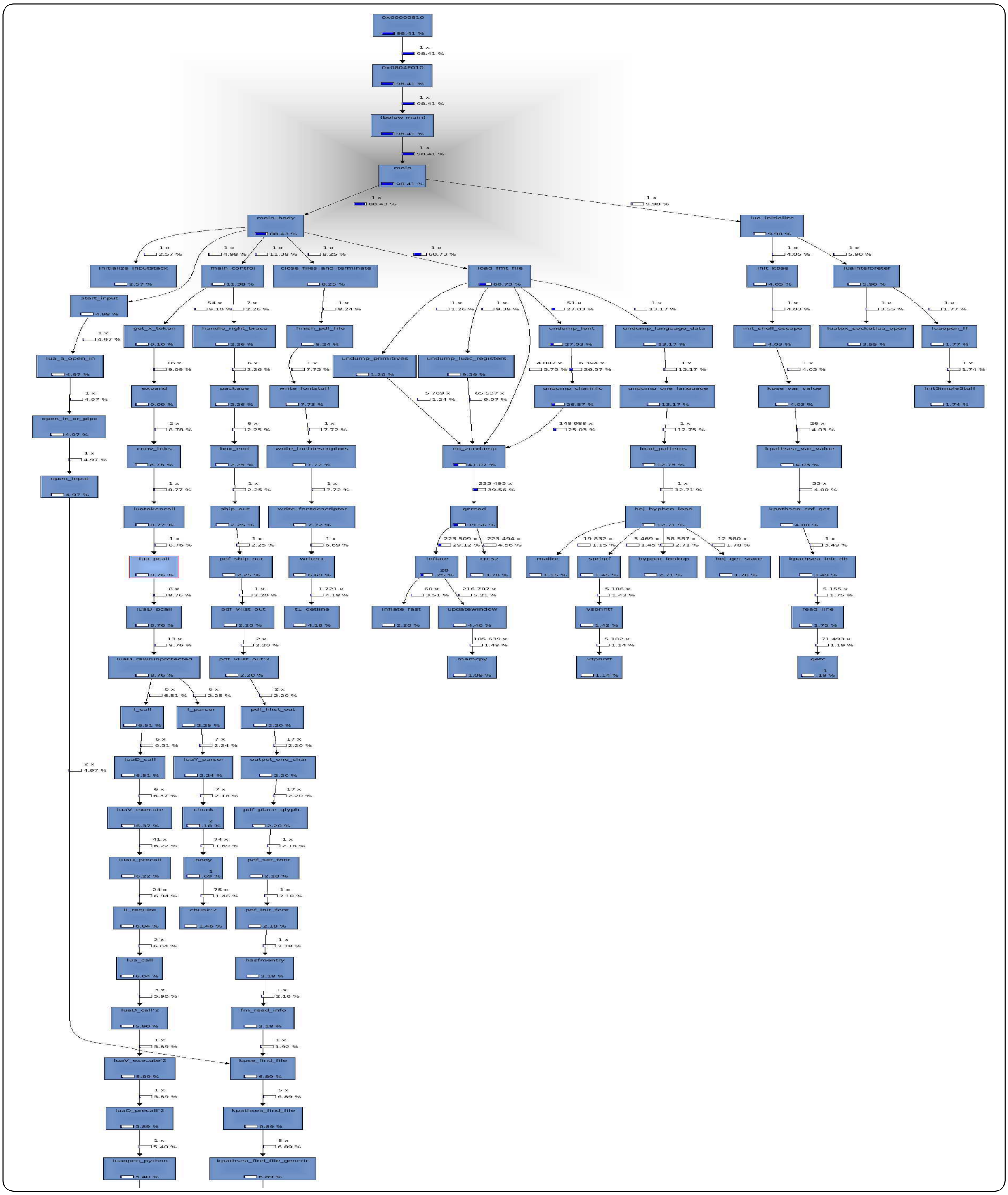
> The lunatic is on the grass
> The lunatic is on the grass
> Remembering games and daisy chains and laughs
> Got to keep the loonies on the path
> — Brain Damage,
> The Dark Side of the Moon,
> Pink Floyd 1970

> Mr. LuaTeX hosts a Python,
> and become a bit lunatic
> — Anonymous

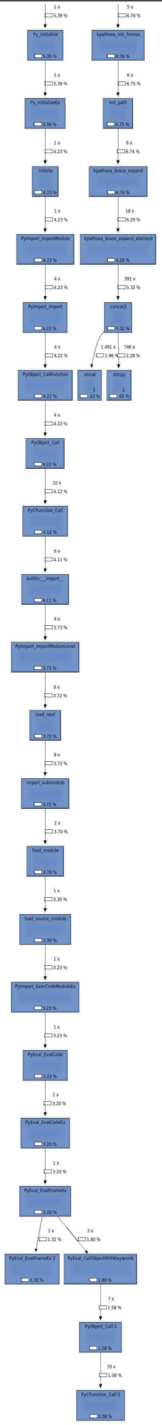
Luigi Scarso

# Appendix

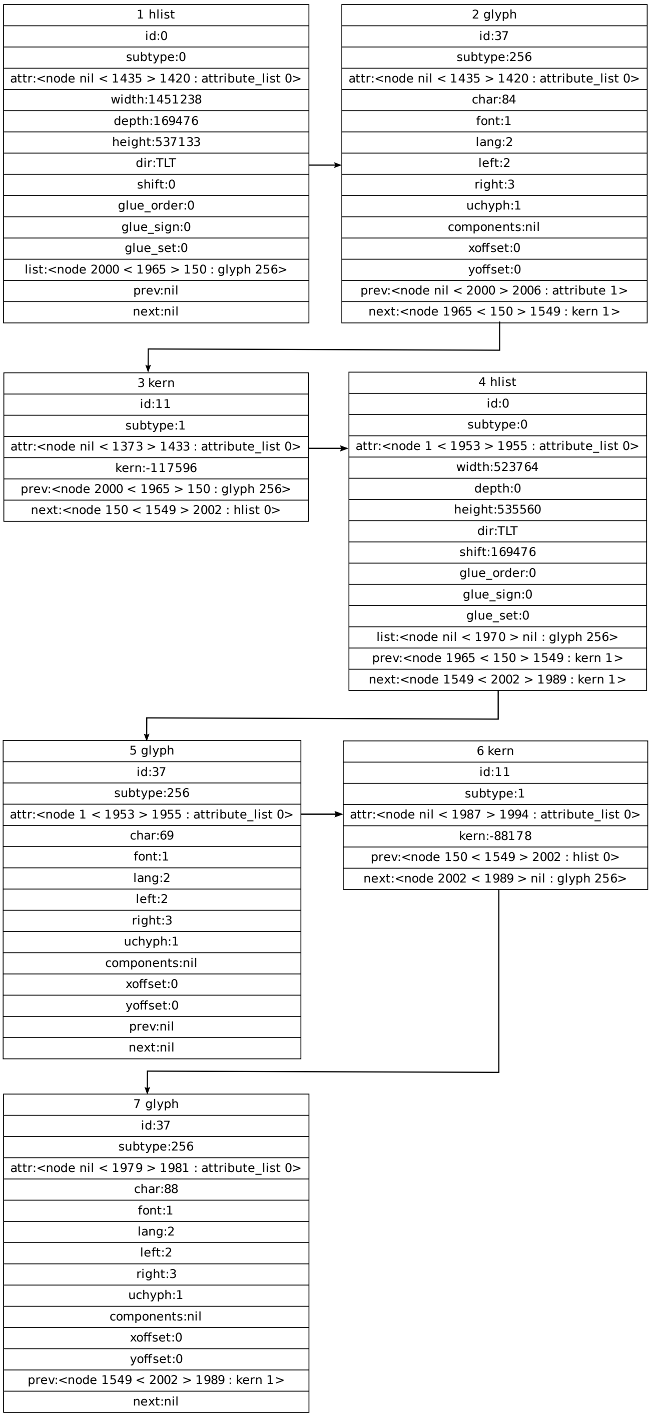
## Call graph of a simple run



## Call graph of a simple run, cont.



## TeX, forever



| 1 hlist |
|---|
| id:0 |
| subtype:0 |
| attr:<node nil < 1435 > 1420 : attribute_list 0> |
| width:1451238 |
| depth:169476 |
| height:537133 |
| dir:TLT |
| shift:0 |
| glue_order:0 |
| glue_sign:0 |
| glue_set:0 |
| list:<node 2000 < 1965 > 150 : glyph 256> |
| prev:nil |
| next:nil |

| 2 glyph |
|---|
| id:37 |
| subtype:256 |
| attr:<node nil < 1435 > 1420 : attribute_list 0> |
| char:84 |
| font:1 |
| lang:2 |
| left:2 |
| right:3 |
| uchyph:1 |
| components:nil |
| xoffset:0 |
| yoffset:0 |
| prev:<node nil < 2000 > 2006 : attribute 1> |
| next:<node 1965 < 150 > 1549 : kern 1> |

| 3 kern |
|---|
| id:11 |
| subtype:1 |
| attr:<node nil < 1373 > 1433 : attribute_list 0> |
| kern:-117596 |
| prev:<node 2000 < 1965 > 150 : glyph 256> |
| next:<node 150 < 1549 > 2002 : hlist 0> |

| 4 hlist |
|---|
| id:0 |
| subtype:0 |
| attr:<node 1 < 1953 > 1955 : attribute_list 0> |
| width:523764 |
| depth:0 |
| height:535560 |
| dir:TLT |
| shift:169476 |
| glue_order:0 |
| glue_sign:0 |
| glue_set:0 |
| list:<node nil < 1970 > nil : glyph 256> |
| prev:<node 1965 < 150 > 1549 : kern 1> |
| next:<node 1549 < 2002 > 1989 : kern 1> |

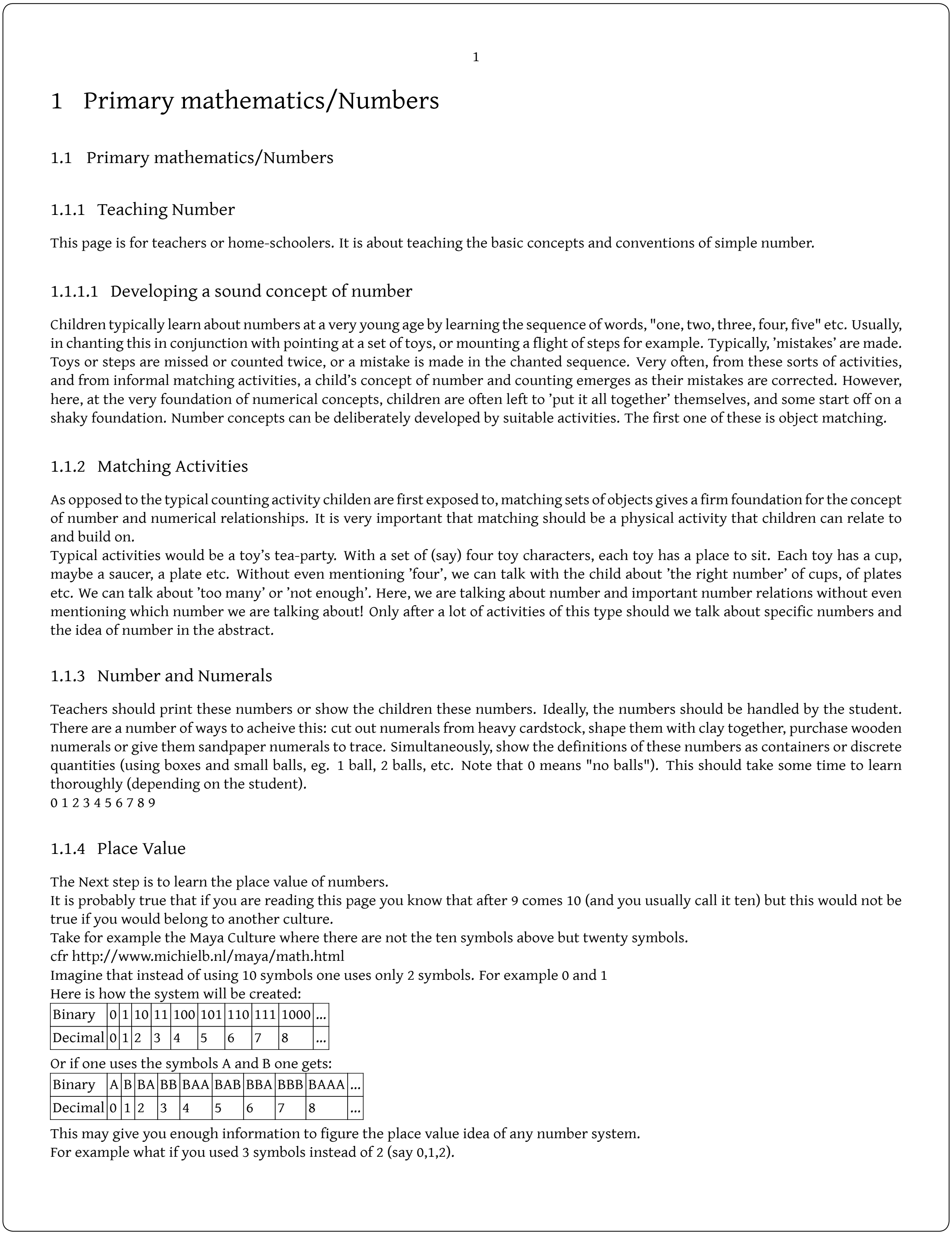
| 5 glyph |
|---|
| id:37 |
| subtype:256 |
| attr:<node 1 < 1953 > 1955 : attribute_list 0> |
| char:69 |
| font:1 |
| lang:2 |
| left:2 |
| right:3 |
| uchyph:1 |
| components:nil |
| xoffset:0 |
| yoffset:0 |
| prev:nil |
| next:nil |

| 6 kern |
|---|
| id:11 |
| subtype:1 |
| attr:<node nil < 1987 > 1994 : attribute_list 0> |
| kern:-88178 |
| prev:<node 150 < 1549 > 2002 : hlist 0> |
| next:<node 2002 < 1989 > nil : glyph 256> |

| 7 glyph |
|---|
| id:37 |
| subtype:256 |
| attr:<node nil < 1979 > 1981 : attribute_list 0> |
| char:88 |
| font:1 |
| lang:2 |
| left:2 |
| right:3 |
| uchyph:1 |
| components:nil |
| xoffset:0 |
| yoffset:0 |
| prev:<node 1549 < 2002 > 1989 : kern 1> |
| next:nil |

TeX nodelist made with lunatic binding for graphviz

## DB XML example



1

# 1 Primary mathematics/Numbers

## 1.1 Primary mathematics/Numbers

### 1.1.1 Teaching Number

This page is for teachers or home-schoolers. It is about teaching the basic concepts and conventions of simple number.

#### 1.1.1.1 Developing a sound concept of number

Children typically learn about numbers at a very young age by learning the sequence of words, "one, two, three, four, five" etc. Usually, in chanting this in conjunction with pointing at a set of toys, or mounting a flight of steps for example. Typically, 'mistakes' are made. Toys or steps are missed or counted twice, or a mistake is made in the chanted sequence. Very often, from these sorts of activities, and from informal matching activities, a child's concept of number and counting emerges as their mistakes are corrected. However, here, at the very foundation of numerical concepts, children are often left to 'put it all together' themselves, and some start off on a shaky foundation. Number concepts can be deliberately developed by suitable activities. The first one of these is object matching.

### 1.1.2 Matching Activities

As opposed to the typical counting activity childen are first exposed to, matching sets of objects gives a firm foundation for the concept of number and numerical relationships. It is very important that matching should be a physical activity that children can relate to and build on.
Typical activities would be a toy's tea-party. With a set of (say) four toy characters, each toy has a place to sit. Each toy has a cup, maybe a saucer, a plate etc. Without even mentioning 'four', we can talk with the child about 'the right number' of cups, of plates etc. We can talk about 'too many' or 'not enough'. Here, we are talking about number and important number relations without even mentioning which number we are talking about! Only after a lot of activities of this type should we talk about specific numbers and the idea of number in the abstract.

### 1.1.3 Number and Numerals

Teachers should print these numbers or show the children these numbers. Ideally, the numbers should be handled by the student. There are a number of ways to acheive this: cut out numerals from heavy cardstock, shape them with clay together, purchase wooden numerals or give them sandpaper numerals to trace. Simultaneously, show the definitions of these numbers as containers or discrete quantities (using boxes and small balls, eg. 1 ball, 2 balls, etc. Note that 0 means "no balls"). This should take some time to learn thoroughly (depending on the student).
0 1 2 3 4 5 6 7 8 9

### 1.1.4 Place Value

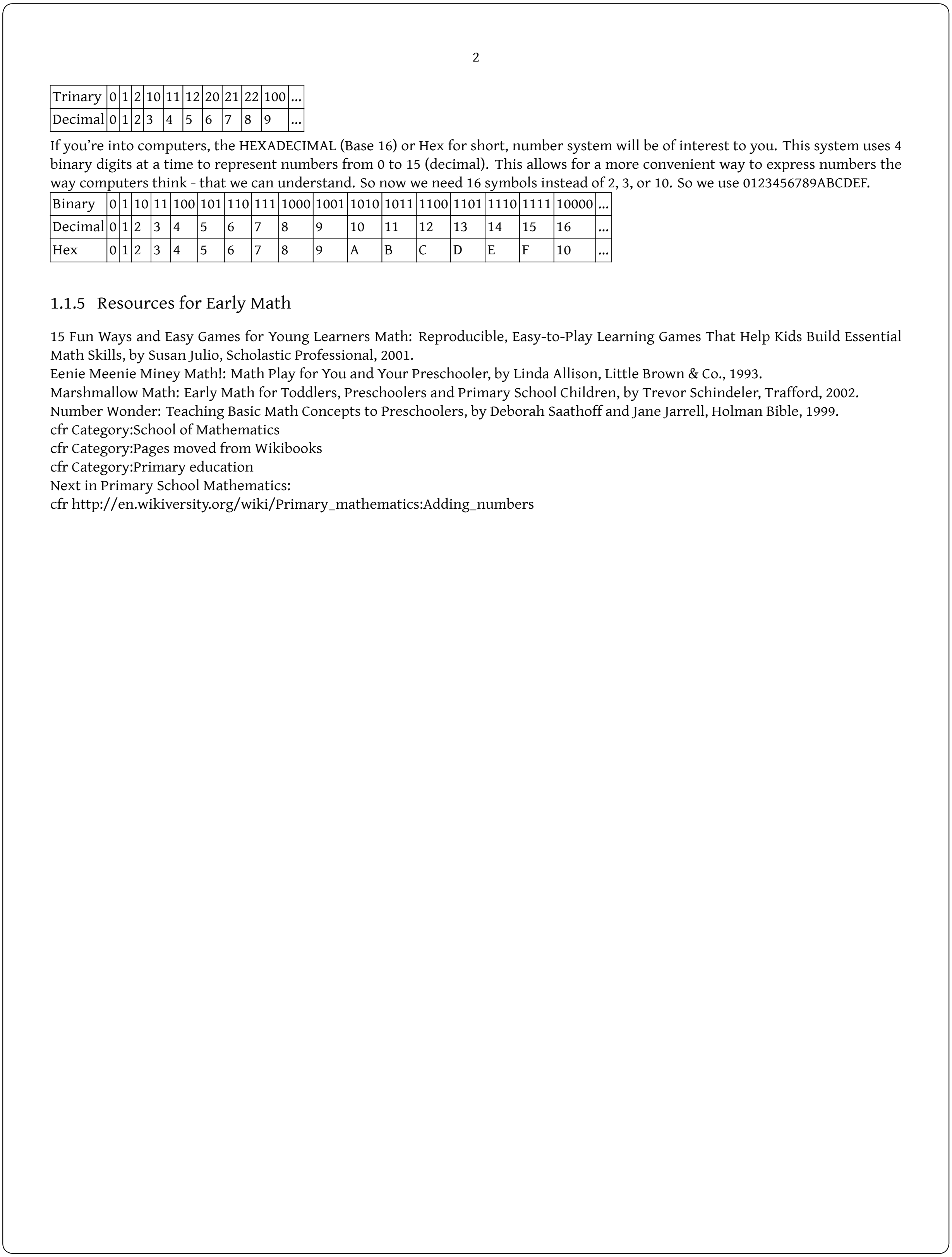
The Next step is to learn the place value of numbers.
It is probably true that if you are reading this page you know that after 9 comes 10 (and you usually call it ten) but this would not be true if you would belong to another culture.
Take for example the Maya Culture where there are not the ten symbols above but twenty symbols.
cfr http://www.michielb.nl/maya/math.html
Imagine that instead of using 10 symbols one uses only 2 symbols. For example 0 and 1
Here is how the system will be created:

| Binary | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |

Or if one uses the symbols A and B one gets:

| Binary | A | B | BA | BB | BAA | BAB | BBA | BBB | BAAA | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |

This may give you enough information to figure the place value idea of any number system.
For example what if you used 3 symbols instead of 2 (say 0,1,2).



| Trinary | 0 | 1 | 2 | 10 | 11 | 12 | 20 | 21 | 22 | 100 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... |

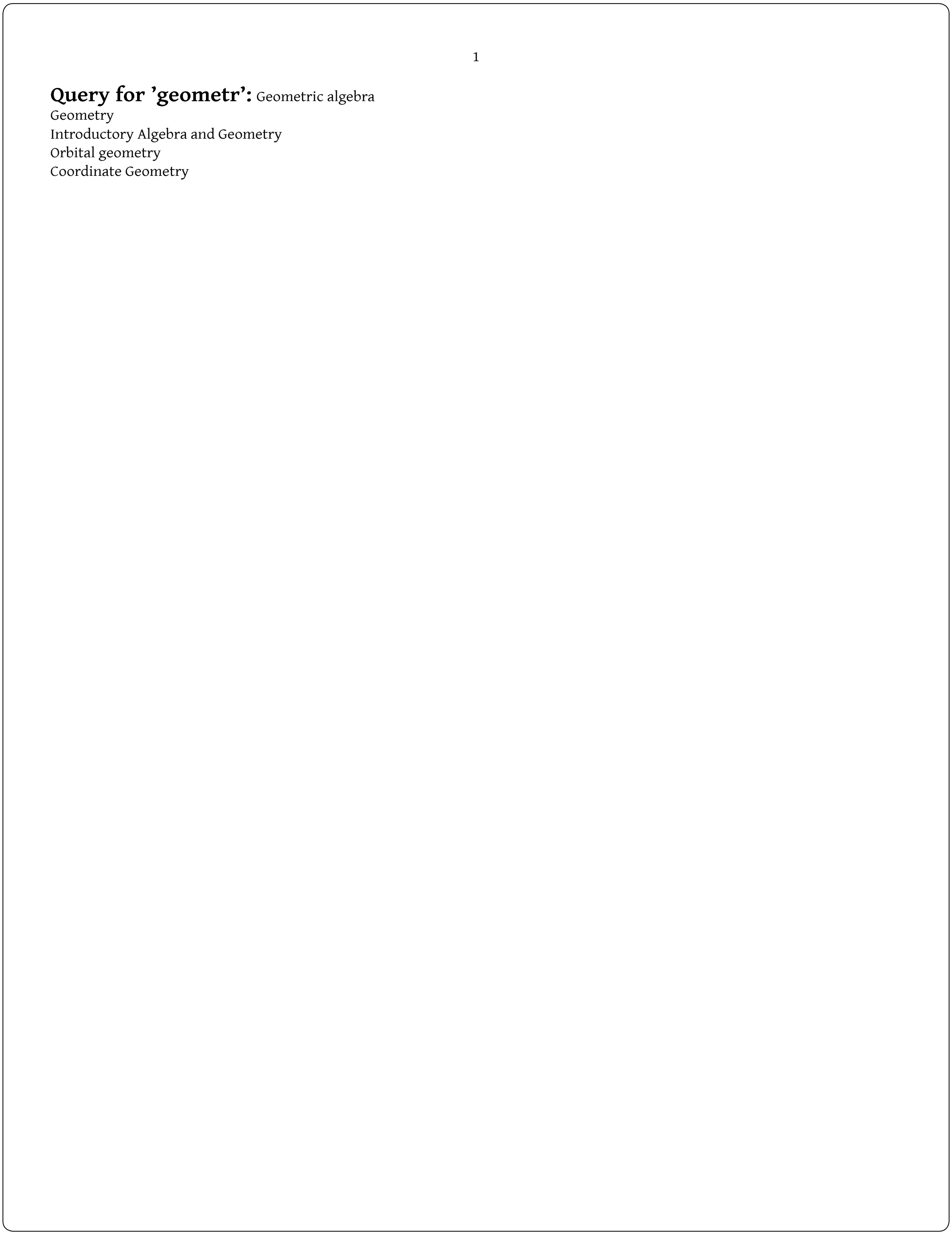
If you're into computers, the HEXADECIMAL (Base 16) or Hex for short, number system will be of interest to you. This system uses 4 binary digits at a time to represent numbers from 0 to 15 (decimal). This allows for a more convenient way to express numbers the way computers think - that we can understand. So now we need 16 symbols instead of 2, 3, or 10. So we use 0123456789ABCDEF.

| Binary | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 | 10000 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | ... |
| Hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | ... |

### 1.1.5 Resources for Early Math

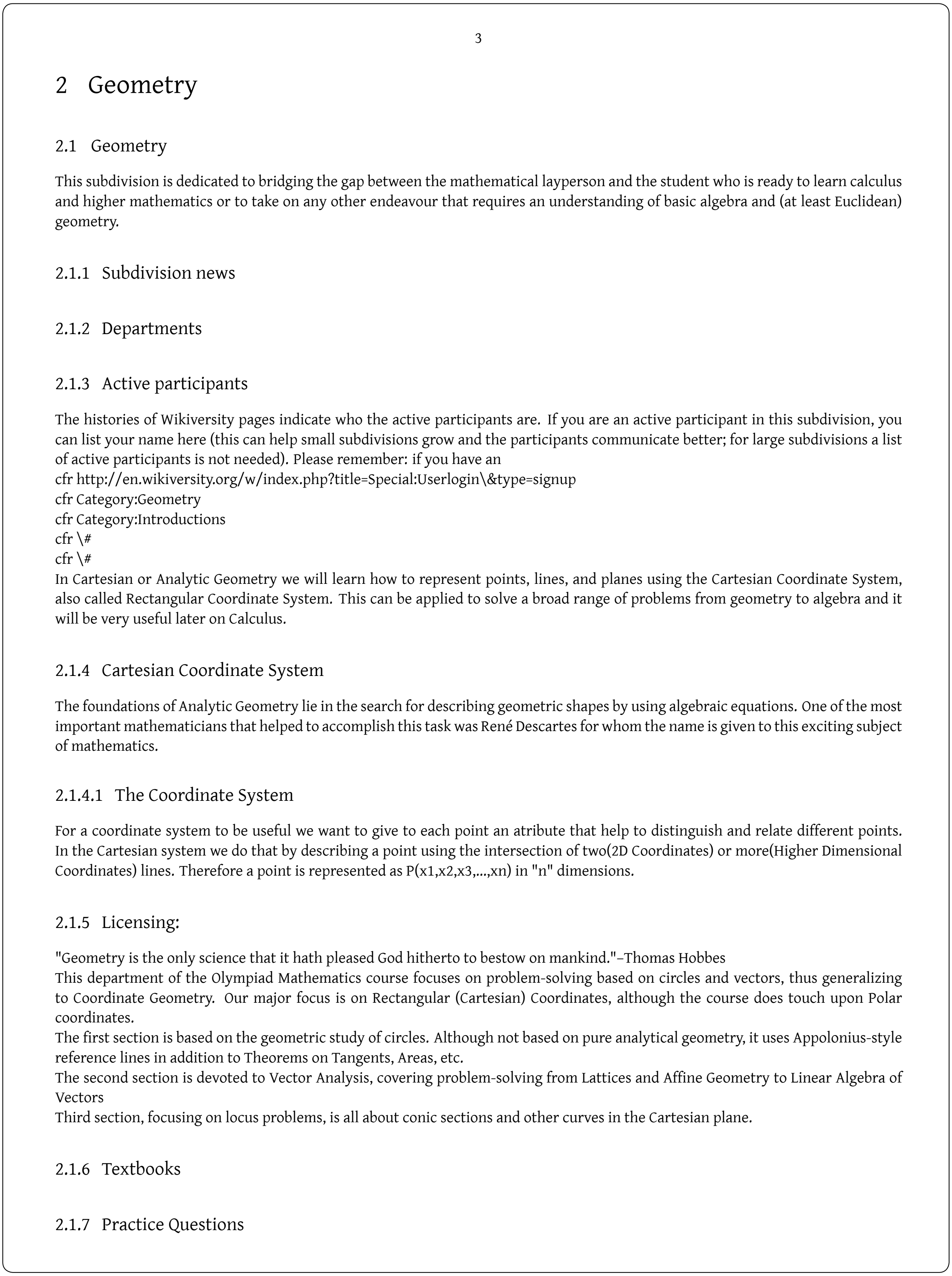
15 Fun Ways and Easy Games for Young Learners Math: Reproducible, Easy-to-Play Learning Games That Help Kids Build Essential Math Skills, by Susan Julio, Scholastic Professional, 2001.
Eenie Meenie Miney Math!: Math Play for You and Your Preschooler, by Linda Allison, Little Brown & Co., 1993.
Marshmallow Math: Early Math for Toddlers, Preschoolers and Primary School Children, by Trevor Schindeler, Trafford, 2002.
Number Wonder: Teaching Basic Math Concepts to Preschoolers, by Deborah Saathoff and Jane Jarrell, Holman Bible, 1999.
cfr Category:School of Mathematics
cfr Category:Pages moved from Wikibooks
cfr Category:Primary education
Next in Primary School Mathematics:
cfr http://en.wikiversity.org/wiki/Primary_mathematics:Adding_numbers

## sqlite example



1

**Query for 'geometr':** Geometric algebra
Geometry
Introductory Algebra and Geometry
Orbital geometry
Coordinate Geometry



# 2 Geometry

## 2.1 Geometry

This subdivision is dedicated to bridging the gap between the mathematical layperson and the student who is ready to learn calculus and higher mathematics or to take on any other endeavour that requires an understanding of basic algebra and (at least Euclidean) geometry.

### 2.1.1 Subdivision news

### 2.1.2 Departments

### 2.1.3 Active participants

The histories of Wikiversity pages indicate who the active participants are. If you are an active participant in this subdivision, you can list your name here (this can help small subdivisions grow and the participants communicate better; for large subdivisions a list of active participants is not needed). Please remember: if you have an
cfr http://en.wikiversity.org/w/index.php?title=Special:Userlogin\&type=signup
cfr Category:Geometry
cfr Category:Introductions
cfr \#
cfr \#
In Cartesian or Analytic Geometry we will learn how to represent points, lines, and planes using the Cartesian Coordinate System, also called Rectangular Coordinate System. This can be applied to solve a broad range of problems from geometry to algebra and it will be very useful later on Calculus.

### 2.1.4 Cartesian Coordinate System

The foundations of Analytic Geometry lie in the search for describing geometric shapes by using algebraic equations. One of the most important mathematicians that helped to accomplish this task was René Descartes for whom the name is given to this exciting subject of mathematics.

#### 2.1.4.1 The Coordinate System

For a coordinate system to be useful we want to give to each point an atribute that help to distinguish and relate different points. In the Cartesian system we do that by describing a point using the intersection of two(2D Coordinates) or more(Higher Dimensional Coordinates) lines. Therefore a point is represented as P(x1,x2,x3,...,xn) in "n" dimensions.

### 2.1.5 Licensing:

"Geometry is the only science that it hath pleased God hitherto to bestow on mankind."–Thomas Hobbes
This department of the Olympiad Mathematics course focuses on problem-solving based on circles and vectors, thus generalizing to Coordinate Geometry. Our major focus is on Rectangular (Cartesian) Coordinates, although the course does touch upon Polar coordinates.
The first section is based on the geometric study of circles. Although not based on pure analytical geometry, it uses Appolonius-style reference lines in addition to Theorems on Tangents, Areas, etc.
The second section is devoted to Vector Analysis, covering problem-solving from Lattices and Affine Geometry to Linear Algebra of Vectors
Third section, focusing on locus problems, is all about conic sections and other curves in the Cartesian plane.

### 2.1.6 Textbooks

### 2.1.7 Practice Questions

# Decorating CD-ROMs and DVDs (Tutorial)

**Abstract**
After having burned a disk you sometimes need to add a label and, if the disk is stored in a jewel case, a booklet and an inlay for the jewel case. The following article describes how to create a label for the disk on a commercial label-sheet and a booklet and an inlay for the jewel case. The following solutions are based on ConTEXt's built-in layer capabilities.

**Keywords**
ConTeXt, CD-ROM, DVD, label, booklet, inlay, layer.

## Label

The label's several elements can be switched on/off with the following three modes:

□ enabling *draft* mode will draw an outline of the label. Beware, simultaneously enabling the *withBackground* mode will obscure the outline.
□ enabling *withLogo* mode will show an image at the top of the label.
□ enabling *withBackground* mode will place the background image on the label.

```
\enablemode[draft]
\enablemode[withLogo]
\enablemode[withBackground]
```

We begin by specifying the main language.

```
\mainlanguage[en]
```

Next, we choose the label font. Since we are using ConTEXt MkIV, we will choose the Iwona-medium font in its otf variant.

```
\usetypescript[iwona-medium]
\setupbodyfont[iwona-medium, 10pt]
```

All texts in the different boxes on the various layers are of type `\framed`. As a visual aid, *draft* mode switches on all their frames.

```
\doifmodeelse{draft}
   {\def\Onoff{on}}
   {\def\Onoff{off}}
```

We will place two images in the background. As a convenience, we define two macros whose names are the names of their respective images. The *Labelbackground* image covers the complete label, while the *Logo* image will appear at the top of the label when the *withLogo* mode is enabled.

```
\def\Labelbackground{fish}
\def\Logo{eurotexlogo}
```

Texts can be placed at the top and bottom of the label, and to the left and right of the disk's center. The bottom area is divided into three sections, each wide enough to fit, depending on the position of the text, within the available space.

To keep things uncluttered, we place the texts in buffers here and use only the buffers' names in the later typesetting instructions. All the texts will be placed in the `\framed` environment.

```
\startbuffer[BottomtextA]
   \green Contains all files for decorating a CD or DVD:\crlf
   \em \tfx CD-label, CD-inlay-booklet, CD-inlay for the jewel case
\stopbuffer
```

```
\startbuffer[BottomtextB]
   \green \em Published in the Euro\TEX -proceedings
\stopbuffer
```

```
\startbuffer[BottomtextC]
   \green 2009
\stopbuffer
```

There is one text area at the top of the label.

```
\startbuffer[Toptext]
   {\tfc \red CD/DVD decoration}\blank \green{\tfb Tutorial}\vfill Over-
lays and layers
\stopbuffer
```

The text areas to the left and right of the disk's center are, like the area at the top of the label, based on a single block of text.

```
\startbuffer[Righttext]
   \green CD-ROM
   \blank
   \gray \currentdate
\stopbuffer
```

```
\startbuffer[Lefttext]
   \green \TEX -tutorial
   \blank
   Euro\TEX 2009
\stopbuffer
```

By separating the content elements above from the typesetting commands below, we can change the content without worrying about the code below this point. We will add a comment to emphasize this.

```
% -- Basically you do not need to edit the lines below
```

First, we tell ConTeXt how the label will look when typeset.

```
\setuppapersize[A4][A4]
\setuppagenumbering[state=stop]
\setupcolors[state=start]
```

```
\setuplayout
   [topspace=0pt,
   backspace=0pt,
   header=0pt,
   footer=0pt,
   margin=0pt,
   width=210mm,
   height=297mm,
   marking=on,
   location=middle]
```

As mentioned above, enabling *draft* mode will draw the label outline. The drawing itself consists of two concentric circles drawn with MetaPost.

```
\startreusableMPgraphic{CDShape}
   draw fullcircle scaled 117mm;
   draw fullcircle scaled 41mm;
\stopreusableMPgraphic
```

ConTEXt provides the `\doifmode[]{}` command, which we will use to set the label background to our predefined background image when the *withBackground* mode is enabled.

```
\doifmode{withBackground}
   {\defineoverlay
      [Lbackground]
      [{\externalfigure
         [\Labelbackground]
         [width=\overlaywidth,height=\overlayheight]}]}
```

We use the same mechanism to place the optional logo image.

```
\doifmode{withLogo}
   {\defineoverlay
      [Logo]
      [{\externalfigure[\Logo]
      [width=\overlaywidth,height=\overlayheight]}]}
```

Here we define a layer that will cover the entire page. To indicate that we do not want relative positioning of this layer, we set its `position` to `no` and its reference point to `bottom`. Lastly, we place the layer into the page area as a background.

```
\definelayer[PageLayer][position=no]

\setuplayer
   [PageLayer]
   [corner=bottom,location=c,height=\paperheight]

\setupbackgrounds[page][background=PageLayer]
```

We define a second layer to hold the label fields we have already defined. We set this layer's reference point to `top` and `left` and the location of the layer data to `bottom right`. We also define its width and height. Set `option=test` to see what ConTEXt does with these settings.

```
\definelayer
   [Label]
   [position=no,corner={top,left},location={bottom,right},
    width=117mm,height=117mm,option=test]
```

In the following lines we fill the Label layer with our predefined label fields, and then typeset it. To tell ConTEXt to flush the layer at the end of the page we enclose the filling commands and the typesetting command: `\placelayer[]` within a `\standardmakeup ... \stopstandardmakeup` block.

```
\starttext
\startstandardmakeup[page=no,doublesided=no]
\setlayerframed
   [Label]
   [x=\dimexpr(\textwidth-117mm)/2,y=21.43mm]
   [width=117mm,height=117mm,frame=\Onoff,background=Lbackground]
   {}
```

```
\setlayerframed
   [Label]
   [x=\dimexpr(\textwidth-13mm)/2,y=22.43mm]
   [width=13mm,height=13mm,frame=\Onoff,align={top,middle},
    background=Logo]
   {}
\setlayerframed
   [Label]
   [x=\dimexpr(\textwidth-78mm)/2,y=35.43mm]
   [width=78mm,height=24mm,frame=\Onoff,align={top,middle}]
   {\getbuffer[Toptext]}
\setlayerframed
   [Label]
   [x=126mm,y=60mm]
   [width=34mm,height=40mm,frame=\Onoff,align={middle,lohi}]
   {\getbuffer[Righttext]}
\setlayerframed
   [Label]
   [x=50mm,y=60mm]
   [width=34mm,height=40mm,frame=\Onoff,align={flushleft,lohi}]
   {\getbuffer[Lefttext]}
\setlayerframed
   [Label]
   [x=\dimexpr(\textwidth-98mm)/2,y=100.43mm]
   [width=98mm,height=\dimexpr(38mm/3),frame=\Onoff,align=middle]
   {\getbuffer[BottomtextA]}
\setlayerframed
   [Label]
   [x=\dimexpr(\textwidth-72mm)/2,y=\dimexpr(100.43mm+38mm/3)]
   [width=72mm,height=\dimexpr(38mm/3),frame=\Onoff,align=middle]
   {\getbuffer[BottomtextB]}
\setlayerframed
   [Label]
   [x=\dimexpr(\textwidth-18mm)/2,y=\dimexpr(100.43mm+38mm/3*2)]
   [width=18mm,height=\dimexpr(38mm/3),frame=\Onoff,align=middle]
   {\getbuffer[BottomtextC]}

\doifmode{draft}
   {\setlayer[PageLayer][x=.5\paperwidth,y=216.93mm]
       {\useMPgraphic{CDShape}}
   \setlayer[PageLayer][x=.5\paperwidth,y=79.93mm]
       {\useMPgraphic{CDShape}}}
\placelayer[Label]
\stopstandardmakeup

\stoptext
```

As you can see in the code above, we move a piece of information to its correct position by adjusting its vertical and horizontal offsets.

Near the end of the code there is another conditional action that controls the placement of the label shape.

The preceding code yields the following result:

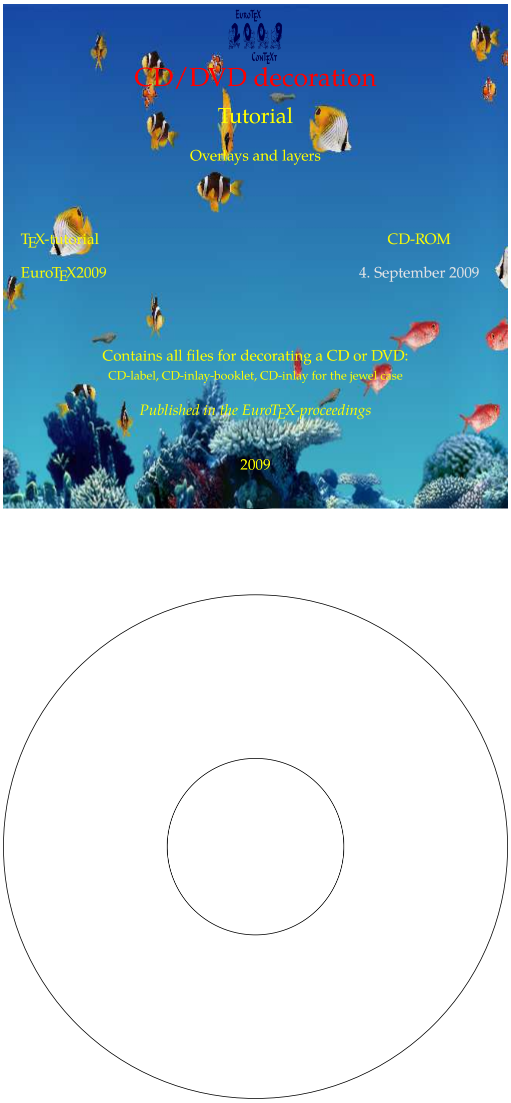

CD/DVD decoration
Tutorial
Overlays and layers
TEX-tutorial
CD-ROM
EuroTEX2009
4. September 2009
Contains all files for decorating a CD or DVD:
CD-label, CD-inlay-booklet, CD-inlay for the jewel case
Published in the EuroTEX-proceedings
2009

Figure 1. Example CD-label

## Booklet

The CD-booklet is composed of a single section which we arrange with `\setuparranging[2UP]`. As before, we start with setting the main language and choosing the font for the booklet.

```
\mainlanguage[en]
```

```
\usetypescript[palatino]
\setupbodyfont[palatino,8pt]
```

If you want to place a background image on the front page, you can define a macro with a symbolic name which will be used later when you setup the page layer.

```
\def\Pageimage{fish}
```

We can add a title typeset along the spine of the cover page. The title is placed in a buffer:

```
\startbuffer[Sidetitle]
  {\tfc \yellow CD/DVD decoration}
\stopbuffer
```

In the next buffer we place the contents of the cover page.

```
\startbuffer[Covertext]
   \strut
   \blank[line]
   \startalignment[middle]
      \startlines
         {\tfc \red CD/DVD decoration}
         \blank \green{\tfb Tutorial}
         \blank Overlays and layers
      \stoplines
   \stopalignment
   \blank
\stopbuffer
```

The booklet contents is put into its own buffer.

```
\startbuffer[Bookletcontent]
   \input knuth\par
\stopbuffer
```

Now that all the content elements for the booklet have been defined, the positioning code that follows, once written, need not be changed.

```
% -- Basically you do not need to edit the lines below
```

The CD-booklet is typeset on a custom size of paper. We define this paper size and put it on landscape A4 sheets. We choose our layout parameters and, because we are using the standard-makeup-environment, we turn vertical centering off.

```
\definepapersize[CDbooklet][width=120mm,height=118mm]
\setuppapersize[CDbooklet][A4,landscape]
\setuppagenumbering[location={bottom,right},alternative=doublesided]
\setupcolors[state=start]
\setupnarrower[left=.5em]
\setuplayout
    [topspace=2mm,
    backspace=9mm,
    header=0pt,
    footer=5mm,
    margin=8.5mm,
    margindistance=.5mm,
    width=100mm,
    height=115mm,
    marking=on,
    location=middle]
\setupmakeup[standard][top=,bottom=]
```

The background image on the cover should be a bleeding image, i.e. it should be larger than the crop marks indicate. However, as soon as we use imposition, the image is cropped to the paper size and the bleed is gone. We define the bleed as follows:

```
\definemeasure[bleed][\dimexpr1mm\relax]
```

Now we define a layer which is filled with the background image.

```
\definelayer
   [Background]
   [position=no]

\setlayer
   [Background]
   {\externalfigure
      [\Pageimage]
      [height=\dimexpr118mm+2\measure{bleed},
       width=\dimexpr120mm+2\measure{bleed}]}
```

The filled layer is placed into the page background. Because we bled the image, we have to add a background offset equal to the bleed.

```
\setupbackgrounds[page][background=Background,
                        backgroundoffset=\measure{bleed}]
```

As mentioned earlier, the booklet is typeset with imposition.

```
\setuparranging[2UP]
```

Now that everything is in place, we can produce the booklet.

```
\starttext

\startstandardmakeup[doublesided=no,page=yes]
   \inleft{\rotate[rotation=90]{%
      \framed[frame=off,align={lohi,middle},width=\textheight]
         {\bfd \getbuffer[Sidetitle]}}}
   \getbuffer[Covertext]
\stopstandardmakeup

\setupbackgrounds[page][background=]
\getbuffer[Bookletcontent]
\stoptext
```

These parameters will produce the following (only the cover page is shown):

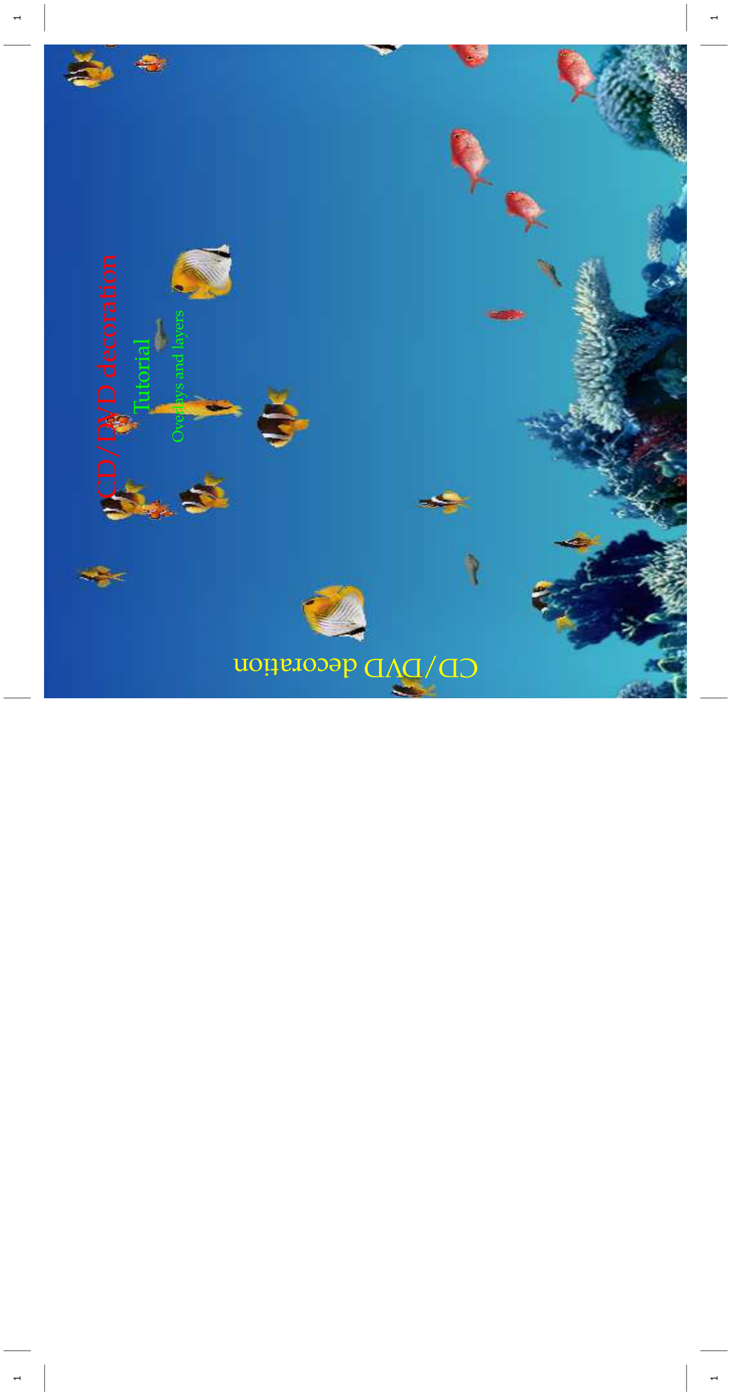


Figure 2. The cover of the booklet

## Jewel case inlay

To complete the CD project, we want to prepare an inlay for the jewel case.

As in the previous sections, we start with setting the main language:

```
\mainlanguage[en]
```

We tell ConTEXt the font we want to use for the inlay

```
\usetypescript[palatino]
\setupbodyfont[palatino,10pt]
```

The inlay will be defined such that you can have as many as three images on the inlay. There is a page image, a text image and an image for the small strips to the left and right of the inlay. Again, we define macros in order to replace the actual filename in the code with a symbolic name.

```
\def\Pageimage{fish}
\def\Textimage{eurotexlogo}
\def\SideTitleimage{}
```

We have three text areas. At the left and right sides of the inlay there are small strips along the spine for the title information. Both strips are automatically filled with the same information. In between these is the main text area.

```
\startbuffer[Sidetitle]
   \tfa \yellow CD/DVD decoration\hfill2009
\stopbuffer

\startbuffer[Maintext]
   \startalignment[middle]
      \startlines
         {\tfc \red CD/DVD decoration}
         \blank \yellow{\tfb Tutorial}
         \blank Overlays and layers
      \stoplines
   \stopalignment
\stopbuffer
```

The following comment reminds us that the code below this point is usually left untouched.

```
% -- Basically you do not need to edit the lines below.
```

We must define the inlay paper size ourselves. Its dimensions are 150 × 118 mm and typeset on an A4.

```
\definepapersize[CDinlaycase][width=150mm,height=118mm]
\setuppapersize[CDinlaycase][A4]
```

We define a bleed measure to insure that the page image will cover the entire page.

```
\definemeasure[bleed][\dimexpr1mm\relax]
```

We specify the various layout settings. The width of the main text area is 14 mm smaller than the paper width. We use a backspace of 7 mm, which is filled with the margin + margin distance.

```
\setuplayout
   [topspace=0mm,
   backspace=7mm,
   margin=6.5mm,
   header=0pt,
   footer=0pt,
   margindistance=.5mm,
   width=136mm,
   height=118mm,
   location=middle,
   marking=on]
```

```
\setupcolors[state=start]
```

```
\setupmakeup[standard][bottom=,top=]
```

To fit the inlay into the jewel case we have to make two folds. It is important to make these folds accurately. To help, we add a layer with fold marks that extend out into the cut space.

```
\definelayer
   [Foldmarks]
   [position=no,
   height=\dimexpr(\paperheight+10mm),
   width=\dimexpr(\paperwidth+10mm),
   x=0mm,
   y=-8mm]
```

Next we define two layers, one each for the page and text images. We do not want the layers to be positioned relative to the text so we set each `position` to `no`.

```
\definelayer
   [Pagebackground]
   [position=no]
```

```
\definelayer
   [Textbackground]
   [position=no]
```

As mentioned earlier, we can use an overlay to add an image or a background along the spine. The image could be a previously defined external image, and the background could be a transparent color generated with MetaPost.

```
\defineoverlay
   [SideTitlebackground]
   [{\externalfigure[\SideTitleimage][width=\overlaywidth,
                                      height=\overlayheight]}]
```

or

```
\defineoverlay
   [SideTitlebackground]
   [\useMPgraphic{TransparentBackground}]
```

We define two additional layers intended for the side titles. Again, there is no need for relative positioning.

```
\definelayer
   [SideTitleL]
   [position=no]
```

```
\definelayer
   [SideTitleR]
   [position=no]
```

We use MetaPost for the fold marks.

```
\startuniqueMPgraphic{Marks}
   path p;
   pair pt[];
   p := unitsquare xscaled 136mm yscaled 135mm;
   pt[1] := llcorner p;
   pt[2] := point 3.95 of p;
   pt[3] := point 3.05 of p;
   pt[4] := ulcorner p;
   pt[5] := lrcorner p;
   pt[6] := point 1.05 of p;
   pt[7] := point 1.95 of p;
   pt[8] := urcorner p;

   for i= 1 step 2 until 7 :
      draw pt[i]--pt[i+1];
   endfor;
\stopuniqueMPgraphic
```

It is easy to prepare a transparent colored background for an overlay with MetaPost.

```
\startreusableMPgraphic{TransparentBackground}
   path p;
   p:= unitsquare xscaled \overlaywidth yscaled \overlayheight;
   fill p withcolor transparent(1,0.3,yellow);
\stopreusableMPgraphic
```

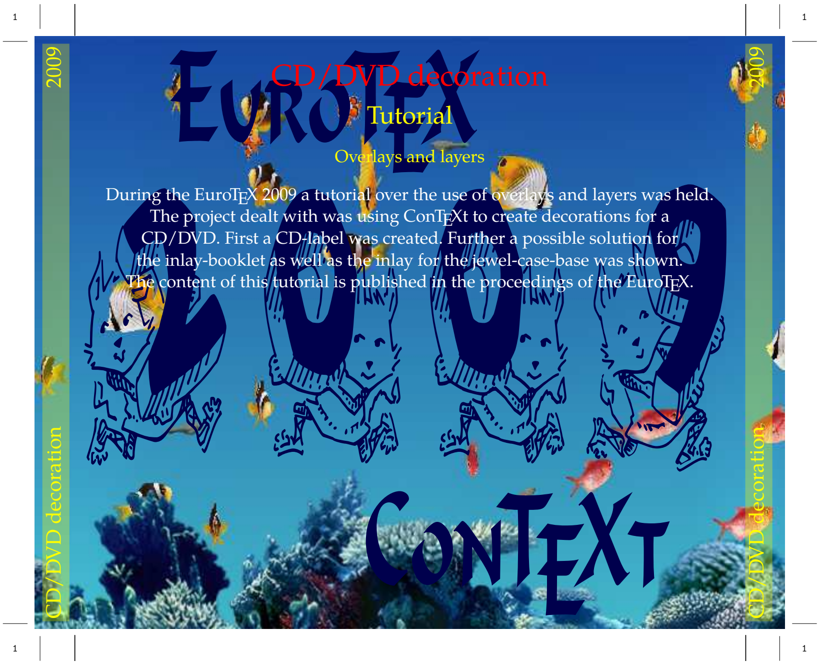
Now we are ready to fill the layers with their respective content, and assign the layers as page or text backgrounds.

```
\setlayer
   [Foldmarks]{\useMPgraphic{Marks}}
\setlayer
   [Pagebackground]
   {\externalfigure
      [\Pageimage]
      [height=\dimexpr118mm+2\measure{bleed},
       width=\dimexpr150mm+2\measure{bleed}]}
\setlayer
   [Textbackground]
   {\externalfigure[\Textimage][height=\textheight,width=\textwidth]}
\setlayer
   [SideTitleL]
   [x=.5mm,y=.5mm]
   {\rotate[rotation=90]{%
      \framed
      [frame=off,align={right,lohi},
      width=\dimexpr\textheight-1mm,
      background=SideTitlebackground]
      {\bf \getbuffer[Sidetitle]}}}
\setlayer
   [SideTitleR]
   [x=-.5mm,y=.5mm]
   {\rotate[rotation=90]{%
      \framed
      [frame=off,align={right,lohi},
      width=\dimexpr\textheight-1mm,
      background=SideTitlebackground]
      {\bf \getbuffer[Sidetitle]}}}

\setupbackgrounds
   [page]
   [background=Pagebackground,backgroundoffset=\measure{bleed}]
\setupbackgrounds[text][background={Foldmarks,Textbackground}]
\setupbackgrounds[text][leftmargin][background=SideTitleL]
\setupbackgrounds[text][rightmargin][background=SideTitleR]
```

What remains to be done is to start a document and add the information for the main text area.

```
\starttext

\strut
   \framedtext
      [frame=off,
      rulethickness=3pt,
      offset=10pt,
      width=\textwidth,
      height=\textheight,
      align=middle]
      {\getbuffer[Maintext]}

\stoptext
```

The result is as follows:



CD/DVD decoration

Tutorial

Overlays and layers

During the EuroTEX 2009 a tutorial over the use of overlays and layers was held. The project dealt with was using ConTEXt to create decorations for a CD/DVD. First a CD-label was created. Further a possible solution for the inlay-booklet as well as the inlay for the jewel-case-base was shown. The content of this tutorial is published in the proceedings of the EuroTEX.

Figure 3. The inlay for the jewel case

## Further reading

□ ConTEXt the manual. Hans Hagen. November 2001, available from http://www.pragma-ade.com/
□ Metafun. Hans Hagen. January 2002, available from http://www.pragma-ade.com/
□ It's in the details. Hans Hagen. Spring 2002, available from http://www.pragma-ade.com/
□ http://wiki.contextgarden.net/Layers.

## Conclusion

This small project demonstrates ConTEXt's capability to place information and graphics at specific locations using layers. Once you become familiar with how they work, you will find many more situations where layers can be used in modern design. It is worthwhile having a look at the literature references given above since there is more to be said about layers than can be presented in this project.

## Acknowledgements

I would like to thank Michael Guravage for proofreading this article and for all the improvements he has made to it.

Willi Egger
w.egger (at) boede.nl

# The language mix

**Abstract**

During the third ConTeXt conference that ran in parallel to EuroTeX 2009 in The Hague we had several sessions where mkiv was discussed and a few upcoming features were demonstrated. The next sections summarize some of that. It's hard to predict the future, especially because new possibilities show up once LuaTeX is opened up more, so remarks about the future are not definitive.

## TeX

From now on, if I refer to TeX in the perspective of LuaTeX I mean "Good Old TeX", the language as well as the functionality. Although LuaTeX provides a couple of extensions it remains pretty close to compatible to its ancestor, certainly from the perspective of the end user.

As most ConTeXt users code their documents in the TeX language, this will remain the focus of MKIV. After all, there is no real reason to abandon it. However, although ConTeXt already stimulates users to use structure where possible and not to use low level TeX commands in the document source, we will add a few more structural variants. For instance, we already introduced `\startchapter` and `\startitem` in addition to `\chapter` and `\item`.

We even go further, by using key/value pairs for defining section titles, bookmarks, running headers, references, bookmarks and list entries at the start of a chapter. And, as we carry around much more information in the (for TeX so typical) auxiliary data files, we provide extensive control over rendering the numbers of these elements when they are recalled (like in tables of contents). So, if you really want to use different texts for all references to a chapter header, it can be done:

```
\startchapter
  [label=emcsquare,
   title={About $e=mc^2$},
   bookmark={einstein},
   list={About $e=mc^2$ (Einstein)},
   reference={$e=mc^2$}]

  ... content ...

\stopchapter
```

Under the hood, the MKIV code base is becoming quite a mix and once we have a more clear picture of where we're heading, it might become even more of a hybrid. Already for some time most of the font handling is done by Lua, and a bit more logic and management might move to Lua as well. However, as we want to be downward compatible we cannot go as far as we want (yet). This might change as soon as more of the primitives have associated Lua functions. Even then it will be a trade off: calling Lua takes some time and it might not pay off at all.

Some of the more tricky components, like vertical spacing, grid snapping, balancing columns, etc. are already in the process of being Luafied and their hybrid form might turn into complete Lua driven solutions eventually. Again, the compatibility issue forces us to follow a stepwise approach, but at the cost of (quite some) extra development time. But whatever happens, the TeX input language as well as machinery will be there.

## MetaPost

I never regret integrating MetaPost support in ConTeXt and a dream came true when MPLIB became part of LuaTeX. Apart from a few minor changes in the way text integrates into MetaPost graphics the user interface in MKIV is the same as in MKII. Insofar as Lua is involved, this is hidden from the user. We use Lua for managing runs and conversion of the result to PDF. Currently generating MetaPost code by Lua is limited to assisting in the typesetting of chemical structure formulas which is now part of the core.

When defining graphics we use the MetaPost language and not some TeX-like variant of it. Information can be passed to MetaPost using special macros (like `\MPcolor`), but most relevant status information is passed automatically anyway.

You should not be surprised if at some point we can request information from TeX directly, because after all this information is accessible. Think of something `w := texdimen(0) ;` being expanded at the MetaPost end instead of `w := \the\dimen0 ;` being passed to MetaPost from the TeX end.

## Lua

What will the user see of Lua? First of all he or she can use this scripting language to generate content. But when making a format or by looking at the statistics printed at the end of a run, it will be clear that Lua is used all over the place.

So how about Lua as a replacement for the TEX input language? Actually, it is already possible to make such “ConTEXt Lua Documents” using MKIV’s built in functions. Each ConTEXt command is also available as a Lua function.

```
\startluacode
  context.bTABLE {
      framecolor = "blue",
      align= "middle",
      style = "type",
      offset=".5ex",
    }
    for i=1,10 do
      context.bTR()
      for i=1,20 do
        local r= math.random(99)
        if r < 50 then
          context.bTD {
            background = "color",
            backgroundcolor = "blue"
          }
          context(context.white("%#2i",r))
        else
          context.bTD()
          context("%#2i",r)
        end
        context.eTD()
      end
      context.eTR()
    end
  context.eTABLE()
\stopluacode
```

Of course it helps if you know ConTEXt a bit. For instance we can as well say:

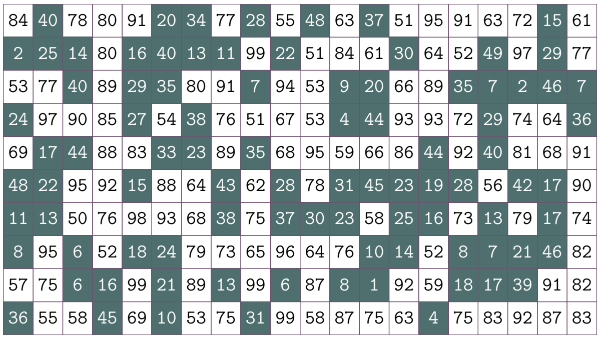
```
if r < 50 then
  context.bTD {
    background = "color",
    backgroundcolor = "blue",
    foregroundcolor = "white",
  }
else
  context.bTD()
end
context("%#2i",r)
context.eTD()
```

And, knowing Lua helps as well, since the following is more efficient:

```
\startluacode
  local colored = {
    background = "color",
    backgroundcolor = "bluegreen",
    foregroundcolor = "white",
  }
  local basespec = {
    framecolor = "bluered",
    align= "middle",
    style = "type",
    offset=".5ex",
  }
  local bTR, eTR = context.bTR, context.eTR
  local bTD, eTD = context.bTD, context.eTD
  context.bTABLE(basespec)
    for i=1,10 do
      bTR()
      for i=1,20 do
        local r= math.random(99)
        bTD((r < 50 and colored) or nil)
        context("%#2i",r)
        eTD()
      end
      eTR()
    end
  context.eTABLE()
\stopluacode
```

Since in practice the speedup is negligible and the memory footprint is about the same, such optimizations seldom make sense.

At some point this interface will be extended, for instance when we can use TEX’s main (scanning, parsing and processing) loop as a so-called coroutine and when we have opened up more of TEX’s internals. Of course, instead of putting this in your TEX source, you can as well keep the code at the Lua end.



| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 84 | 40 | 78 | 80 | 91 | 20 | 34 | 77 | 28 | 55 | 48 | 63 | 37 | 51 | 95 | 91 | 63 | 72 | 15 | 61 |
| 2 | 25 | 14 | 80 | 16 | 40 | 13 | 11 | 99 | 22 | 51 | 84 | 61 | 30 | 64 | 52 | 49 | 97 | 29 | 77 |
| 53 | 77 | 40 | 89 | 29 | 35 | 80 | 91 | 7 | 94 | 53 | 9 | 20 | 66 | 89 | 35 | 7 | 2 | 46 | 7 |
| 24 | 97 | 90 | 85 | 27 | 54 | 38 | 76 | 51 | 67 | 53 | 4 | 44 | 93 | 93 | 72 | 29 | 74 | 64 | 36 |
| 69 | 17 | 44 | 88 | 83 | 33 | 23 | 89 | 35 | 68 | 95 | 59 | 66 | 86 | 44 | 92 | 40 | 81 | 68 | 91 |
| 48 | 22 | 95 | 92 | 15 | 88 | 64 | 43 | 62 | 28 | 78 | 31 | 45 | 23 | 19 | 28 | 56 | 42 | 17 | 90 |
| 11 | 13 | 50 | 76 | 98 | 93 | 68 | 38 | 75 | 37 | 30 | 23 | 58 | 25 | 16 | 73 | 13 | 79 | 17 | 74 |
| 8 | 95 | 6 | 52 | 18 | 24 | 79 | 73 | 65 | 96 | 64 | 76 | 10 | 14 | 52 | 8 | 7 | 21 | 46 | 82 |
| 57 | 75 | 6 | 16 | 99 | 21 | 89 | 13 | 99 | 6 | 87 | 8 | 1 | 92 | 59 | 18 | 17 | 39 | 91 | 82 |
| 36 | 55 | 58 | 45 | 69 | 10 | 53 | 75 | 31 | 99 | 58 | 87 | 75 | 63 | 4 | 75 | 83 | 92 | 87 | 83 |

Figure 1. The result of the displayed Lua code.

The script that manages a ConTEXt run (also called context) will process files with that consist of such commands directly if they have a cld suffix or when you provide the flag --forcecld.[1]

```
context yourfile.cld
```

But will this replace TEX as an input language? This is quite unlikely because coding documents in TEX is so convenient and there is not much to gain here. Of course in a pure Lua based workflow (for instance publishing information from databases) it would be nice to code in Lua, but even then it's mostly syntactic sugar, as TEX has to do the job anyway. However, eventually we will have a quite mature Lua counterpart.

## XML

This is not so much a programming language but more a method of tagging your document content (or data). As structure is rather dominant in XML, it is quite handy for situations where we need different output formats and multiple tools need to process the same data. It's also a standard, although this does not mean that all documents you see are properly structured. This in turn means that we need some manipulative power in ConTEXt, and that happens to be easier to do in MKIV than in MKII.

In ConTEXt we have been supporting XML for a long time, and in MKIV we made the switch from stream based to tree based processing. The current implementation is mostly driven by what has been possible so far but as LuaTEX becomes more mature, bits and pieces will be reimplemented (or at least cleaned up and brought up to date with developments in LuaTEX).

One could argue that it makes more sense to use XSLT for converting XML into something TEX, but in most of the cases that I have to deal with much effort goes into mapping structure onto a given layout specification. Adding a bit of XML to TEX mapping to that directly is quite convenient. The total amount of code is probably smaller and it saves a processing step.

We're mostly dealing with education-related documents and these tend to have a more complex structure than the final typeset result shows. Also, readability of code is not served with such a split as most mappings look messy anyway (or evolve that way) due to the way the content is organized or elements get abused.

There is a dedicated manual for dealing with XML in MKIV, so we only show a simple example here. The documents to be processed are loaded in memory and serialized using setups that are associated to elements. We keep track of documents and nodes in a way that permits multipass data handling (rather usual in TEX). Say that we have a document that contains questions. The following definitions will flush the (root element) `questions`:

```
\startxmlsetups xml:mysetups
    \xmlsetsetup{#1}{questions}{xml:questions}
\stopxmlsetups

\xmlregistersetup{xml:mysetups}

\startxmlsetups xml:questions
    \xmlflush{#1}
\stopxmlsetups

\xmlprocessfile{main}{somefile.xml}{}
```

Here the `#1` represents the current XML element. Of course we need more associations in order to get something meaningful. If we just serialize then we have mappings like:

```
\xmlsetsetup{#1}{question|answer}{xml:*}
```

So, questions and answers are mapped onto their own setup which flushes them, probably with some numbering done at the spot.

In this mechanism Lua is sort of invisible but quite busy as it is responsible for loading, filtering, accessing and serializing the tree. In this case TEX and Lua hand over control in rapid succession.

You can hook in your own functions, like:

```
\xmlfilter{#1}
  {(wording|feedback|choice)/function(cleanup)}
```

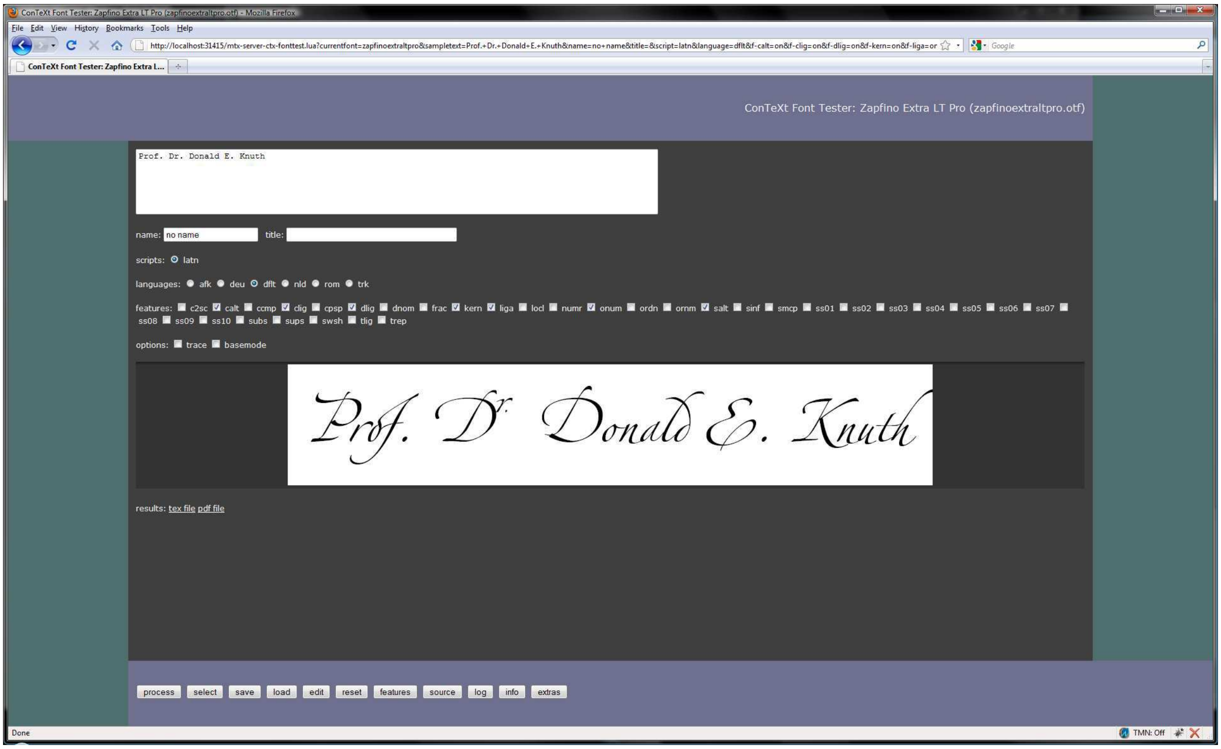
In this case the function `cleanup` is applied to elements with names that match one of the three given.[2]

Of course, once you start mixing in Lua in this way, you need to know how we deal with XML at the Lua end. The following function show how we calculate scores:

```
\startluacode
function xml.functions.totalscore(root)
  local n = 0
  for e in xml.collected(root,"/outcome") do
    if xml.filter(e,"action[text()='add']") then
      local m = xml.filter
                    (e,"xml:///score/text()")
      n = n + (tonumber(m or 0) or 0)
    end
  end
  tex.write(n)
end
\stopluacode
```

You can either use such a function in a filter or just use it as a TEX macro:

```
\startxmlsetups xml:question
  \blank
  \xmlfirst{#1}{wording}
  \startitemize
    \xmlfilter{#1}
      {/answer/choice/command(xml:answer:choice)}
  \stopitemize
```


ConTeXt Font Tester: Zapfino Extra LT Pro (zapfinoextraltpro.otf) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
ConTeXt Font Tester: Zapfino Extra L...
process
select
save
load
edit
reset
features
source
log
info
extras
Done
TMN: Off

http://localhost:31415/mtx-server-ctx-fonttest.lua?currentfont=zapfinoextraltpro&sampletext=Prof.+Dr.+Donald+E.+Knuth&name=no+name&title=&script=latn&language=dflt&f-calt=on&f-clig=on&f-dlig=on&f-kern=on&f-liga=or

ConTeXt Font Tester: Zapfino Extra LT Pro (zapfinoextraltpro.otf)

```
Prof. Dr. Donald E. Knuth
```

name: no name title:

scripts: ◉ latn

languages: ○ afk ○ deu ◉ dflt ○ nld ○ rom ○ trk

features: ☐ c2sc ☑ calt ☐ ccmp ☑ clig ☐ cpsp ☑ dlig ☐ dnom ☐ frac ☑ kern ☑ liga ☐ locl ☐ numr ☑ onum ☐ ordn ☐ ornm ☑ salt ☐ sinf ☐ smcp ☐ ss01 ☐ ss02 ☐ ss03 ☐ ss04 ☐ ss05 ☐ ss06 ☐ ss07 ☐ ss08 ☐ ss09 ☐ ss10 ☐ subs ☐ sups ☐ swsh ☐ tlig ☐ trep

options: ☐ trace ☐ basemode

Prof. Dr. Donald E. Knuth

results: tex file pdf file

Figure 2. An example of using the font tester.

```
  \endgraf
  score: \xmlfunction{#1}{totalscore}
  \blank
\stopxmlsetups

\startxmlsetups xml:answer:choice
    \startitem
        \xmlflush{#1}
    \stopitem
\stopxmlsetups
```

The filter variant is like this:

```
\xmlfilter{#1}{./function('totalscore')}
```

So you can take your choice and make your source look more XML-ish, Lua-like or TEX-wise. A careful reader might have noticed the peculiar `xml://` in the function code. When used inside MKIV, the serializer defaults to TEX so results are piped back into TEX. This prefix forced the regular serializer which keeps the result at the Lua end.

Currently some of the XML related modules, like MATHML and handling of tables, are really a mix of TEX code and Lua calls, but it makes sense to move them completely to Lua. One reason is that their input (formulas and table content) is restricted to non-TEX anyway. On the other hand, in order to be able to share the implementation with TEX input, it also makes sense to stick to some hybrid approach. In any case, more of the calculations and logic will move to Lua, while TEX will deal with the content.

A somewhat strange animal here is XSL-FO. We do support it, but the MKII implementation was always somewhat limited and the code was quite complex. So, this needs a proper rewrite in MKIV, which will happen indeed. It's mostly a nice exercise of hybrid technology but until now I never really needed it. Other bits and pieces of the current XML goodies might also get an upgrade.

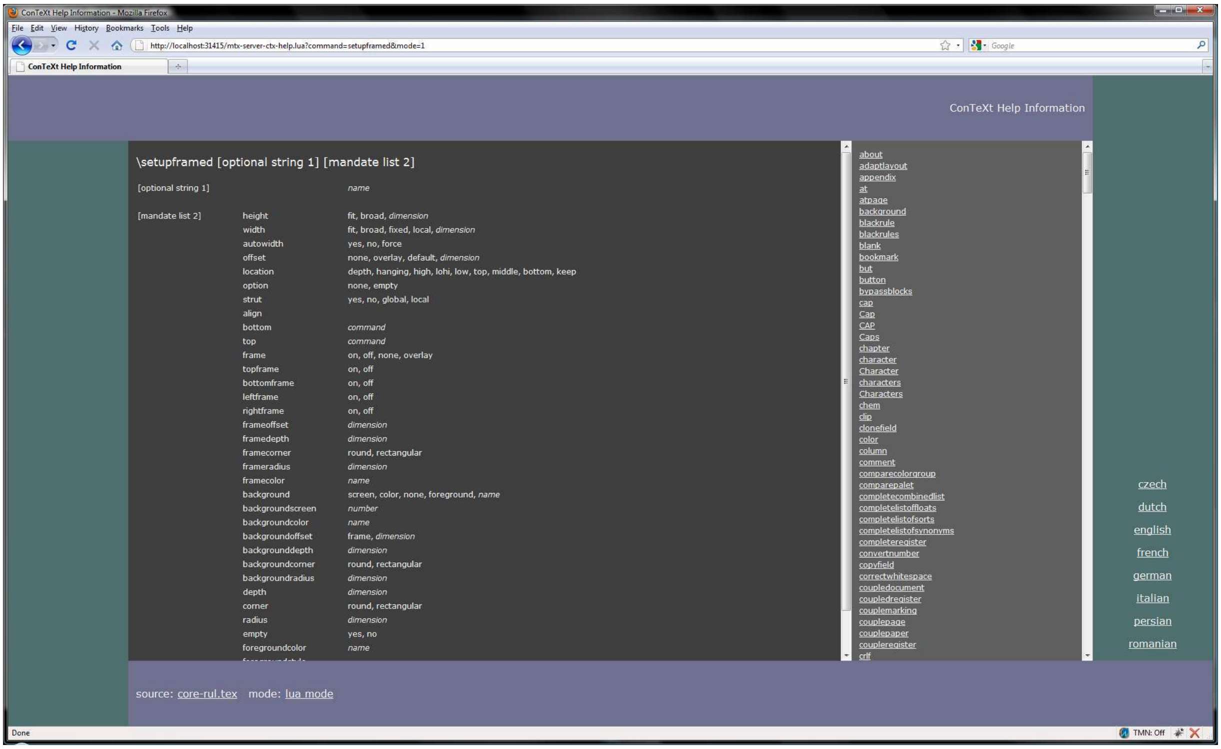
There is already a bunch of functions and macros to filter and manipulate XML content and currently the code involved is being cleaned up. What direction we go also depends on users' demands. So, with respect to XML you can expect more support, a better integration and an upgrade of some supported XML related standards.

## Tools

Some of the tools that ship with ConTEXt are also examples of hybrid usage.

Take this:

```
mtxrun --script server --auto
```



\setupframed [optional string 1] [mandate list 2]

| | | |
|---|---|---|
| [optional string 1] | | *name* |
| [mandate list 2] | height | fit, broad, *dimension* |
| | width | fit, broad, fixed, local, *dimension* |
| | autowidth | yes, no, force |
| | offset | none, overlay, default, *dimension* |
| | location | depth, hanging, high, lohi, low, top, middle, bottom, keep |
| | option | none, empty |
| | strut | yes, no, global, local |
| | align | |
| | bottom | *command* |
| | top | *command* |
| | frame | on, off, none, overlay |
| | topframe | on, off |
| | bottomframe | on, off |
| | leftframe | on, off |
| | rightframe | on, off |
| | frameoffset | *dimension* |
| | framedepth | *dimension* |
| | framecorner | round, rectangular |
| | frameradius | *dimension* |
| | framecolor | *name* |
| | background | screen, color, none, foreground, *name* |
| | backgroundscreen | *number* |
| | backgroundcolor | *name* |
| | backgroundoffset | frame, *dimension* |
| | backgrounddepth | *dimension* |
| | backgroundcorner | round, rectangular |
| | backgroundradius | *dimension* |
| | depth | *dimension* |
| | corner | round, rectangular |
| | radius | *dimension* |
| | empty | yes, no |
| | foregroundcolor | *name* |

source: core-rul.tex mode: lua mode

Figure 3. An example of a help screen for a command.

On my machine this reports:

```
MTXrun | running at port: 31415
MTXrun | document root: c:/data/develop/context/
                                              lua
MTXrun | main index file: unknown
MTXrun | scripts subpath: c:/data/develop/context
                                             /lua
MTXrun | context services: http://localhost:31415
                          /mtx-server-ctx-startup.lua
```

The mtxrun script is a Lua script that acts as a controller for other scripts, in this case mtx-server.lua that is part of the regular distribution. As we use LuaTEX as a Lua interpreter and since LuaTEX has a socket library built in, it can act as a web server, limited but quite right for our purpose.[3]

The web page that pops up when you enter the given address lets you currently choose between the ConTEXt help system and a font testing tool. In figure 2 you seen an example of what the font testing tool does.

Here we have LuaTEX running a simple web server but it's not aware of having TEX on board. When you click on one of the buttons at the bottom of the screen, the server will load and execute a script related to the request and in this case that script will create a TEX file and call LuaTEX with ConTEXt to process that file. The result is piped back to the browser.

You can use this tool to investigate fonts (their bad and good habits) as well as to test the currently available OpenType functionality in MKIV (bugs as well as goodies).

So again we have a hybrid usage although in this case the user is not confronted with Lua and/or TEX at all. The same is true for the other goodie, shown in figure 3. Actually, such a goodie has always been part of the ConTEXt distribution but it has been rewritten in Lua.

The ConTEXt user interface is defined in an XML file, and this file is used for several purposes: initializing the user interfaces at format generation time, typesetting the formal command references (for all relevant interface languages), for the wiki, and for the mentioned help goodie.

Using the mix of languages permits us to provide convenient processing of documents that otherwise would demand more from the user than it does now. For instance, imagine that we want to process a series of documents in the so-called EPUB format. Such a document is a zipped file that has a description and resources. As the content of this archive is prescribed it's quite easy to process it:

```
context --ctx=x-epub.ctx yourfile.epub
```

This is equivalent to:

```
texlua mtxrun.lua --script context --ctx=x-epub.
                                ctx yourfile.epub
```

So, here we have LuaTEX running a script that itself (locates and) runs a script `context`. That script loads a ConTEXt job description file (with suffix `ctx`). This file tells what styles to load and might have additional directives but none of that has to bother the end user. In the automatically loaded style we take care of reading the XML files from the zipped file and eventually map the embedded HTML like files onto style elements and produce a PDF file. So, we have Lua managing a run and MKIV managing with help of Lua reading from zip files and converting XML into something that TEX is happy with. As there is no standard with respect to the content itself, i.e. the rendering is driven by whatever kind of structure is used and whatever the CSS file is able to map it onto, in practice we need an additional style for this class of documents. But anyway it's a good example of integration.

## The future

Apart from these language related issues, what more is on the agenda? To mention a few integration related thoughts:

□ At some point I want to explore the possibility to limit processing to just one run, for instance by doing trial runs without outputting anything but still collecting multipass information. This might save some runtime in demanding workflows especially when we keep extensive font loading and image handling in mind.

□ Related to this is the ability to run MKIV as a service but that demands that we can reset the state of LuaTEX and actually it might not be worth the trouble at all given faster processors and disks. Also, it might not save much runtime on larger jobs.

□ More interesting can be to continue experimenting with isolating parts of ConTEXt in such a way that one can construct a specialized subset of functionality. Of course the main body of code will always be loaded as one needs basic typesetting anyway.

Of course we keep improving existing mechanisms and improve solutions using a mix of TEX and Lua, using each language (and system) for what it can do best.

## Notes

1. Similar methods exist for processing XML files.
2. This example is inspired by one of our projects where the cleanup involves sanitizing (highly invalid) HTML data that is embedded as a `CDATA` stream, a trick to prevent the XML file to be invalid.
3. This application is not intentional but just a side effect.

Hans Hagen
Pragma ADE, Hasselt
pragma (at) wxs dot nl

# E16 & DEtool

## typesetting language data using ConTeXt

**Abstract**
This article describes two recent projects in which ConTeXt was used to typeset language data. The goal of project E16 was to typeset the 16th edition of the Ethnologue, an encyclopaedia of the languages of the world. The complexity of the data and the size of the project made this an interesting test case for the use of TeX and ConTeXt. The Dictionary Express tool (DEtool) is developed to typeset linguistic data in a dictionary layout. DEtool (which is part of a suite of linguistic software) uses ConTeXt for the actual typesetting.

## Introduction

Some background: SIL is an NGO dedicated to serve the world's minority language communities in a variety of language-related ways. Collecting all sorts of language data is the basis of much of the work. This could be things like the number of speakers of a particular language, relations between different languages, literacy rates and bi- and multilingualism. Much of this data ends up in a huge database, which in turn is used as the source for publications like the Ethnologue.[1] which is an encyclopaedia of languages. It consists of four parts, starting with an introductory chapter explaining the scope of the publication and 25 pages of 'Statistical summaries'. Part 1 has 600 pages with language descriptions, describing all the 6909 languages of the world. Part 2 consists of 200 pages with language maps and Part 3 has of 400 pages of indexes, for Language names, Language Codes and Country names.

## Typesetting the Ethnologue

Data flow and directory structure: All the data is stored in an Oracle database running on a secure web server. The XML output is manipulated using XSLT to serve different 'views'. One output path leads to html (for the website http://www.ethnologue.com) and another output path gives TeX-output of with the codes are defined in ConTeXt. Once the data is downloaded from the server, it is stored locally in the 'data' directory of the typesetting system. There is also a 'content' directory containing small files that \input the data files (and do some tricky things with catcodes.) All the content-files are loaded using a 'project' file in the root directory. This (slightly complicated) process allows for easy updating of the data and convenient testing of all the different parts, both separately and together. The macro definitions are all stored in a module.

### Module

In good ConTeXt style all the code for this project is placed in a module. A ConTeXt module starts with a header like this:

```
%D \module
%D   [       file=p-ethnologue,
%D        version=2009.01.14
%D          title=\CONTEXT\ User Module,
%D       subtitle=Typesetting Ethnologue 16,
%D         author=Jelle Huisman, SIL International,
%D           date=\currentdate,
%D      copyright=SIL International]
%C Copyright SIL International
```

```
\writestatus{loading}{Context User Module  Typesetting Ethnologue 16}
\unprotect
\startmodule[ethnologue]
```

```
All the macro definitions go here... and the module is closed with:
```

```
\stopmodule
\protect \endinput
```

With the command `texexec --modu p-ethnologue.tex` it is easy to make a pdf with the module code, comments and even an index.

## E16 code examples

A couple of code examples are presented here to give an impression of the project. This is part of the standard page setup for the paper size and the setup of two basic layouts.

```
\definepapersize [ethnologue][width=179mm, height=255mm]
```

```
\startmode[book] % basic page layout for the book
\setuppapersize [ethnologue][letter]% paper size for book mode
\setuplayout[backspace=18mm, width=148mm, topspace=7mm, top=0mm,
             header=6mm, footer=7mm, height=232mm]
\stopmode
```

```
\startmode[proofreading] % special layout for proofreading mode
\setuppapersize [letter][letter]% paper size for proofreading mode
\setuplayout[backspace=18mm, width=160mm, topspace=7mm, top=0mm,
             header=16mm, footer=6mm, height=250mm]
\stopmode
```

### Use of modes: proofreading vs. final output

To facilitate the proofreading a special proofreading ‘mode’ was defined with wider margins, as shown in the code example in the previous section and with a single column layout (not in this code example). The ‘modes’ mechanism is used to switch between different setups. This code:

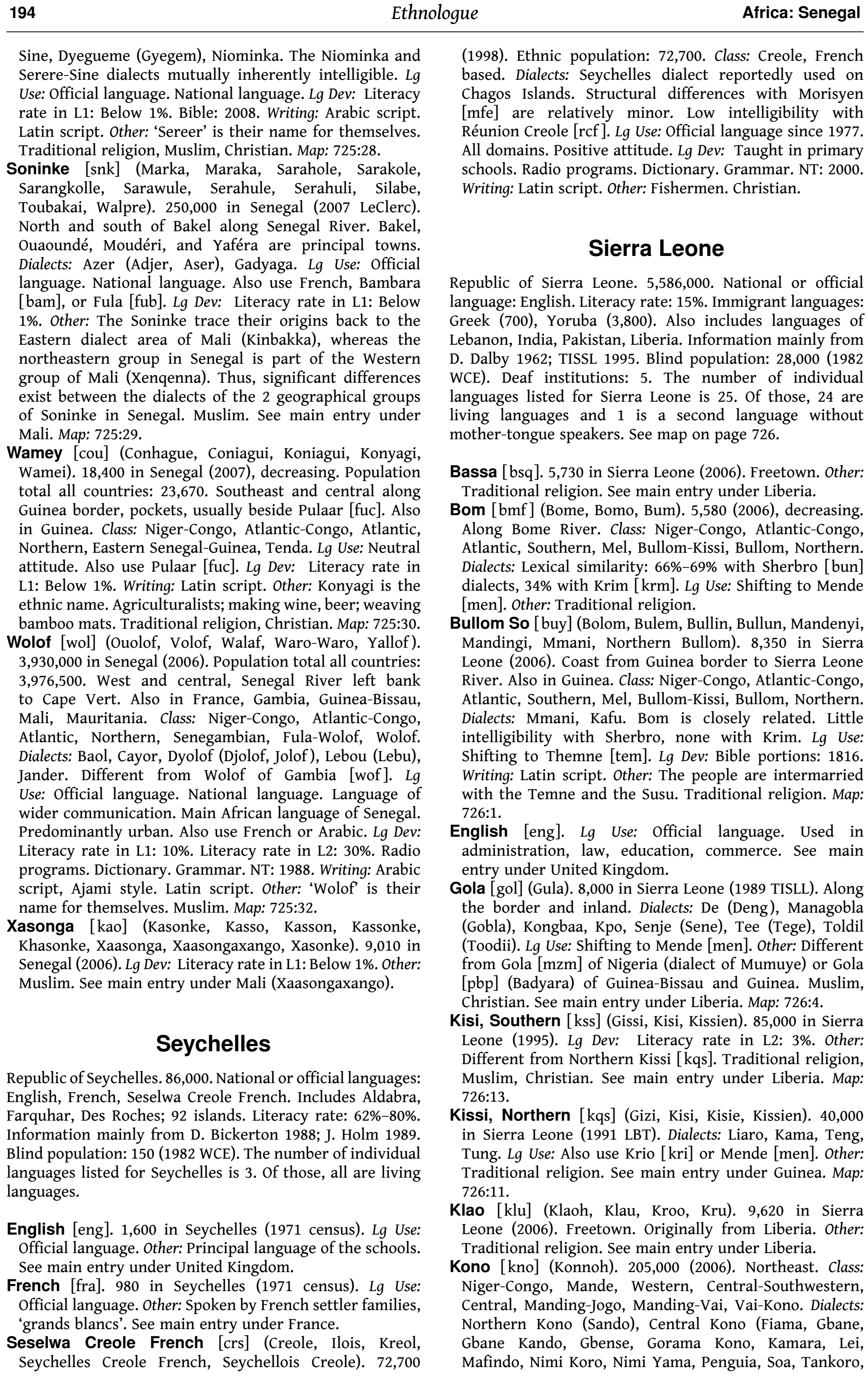
```
%\enablemode[book]
\enablemode[proofreading]
```

is used in a ‘project setup’ file to switch between the proofreading mode (single column, bigger type) and the book mode showing the layout of the final publication. One other application of modes is the possible publication of separate extracts with e.g. the language descriptions of only one country. This could be published using a Printing on Demand process.

### Language description

The biggest part of the publication is the section with the language descriptions. Each language description consists of: a page reference (not printed), the language name, the language code, a short language description and a couple of special ‘items’ like: language class, dialects, use and writing system. This is an example of the raw data for Belarusian:

```
\startLaDes{ % start of Language Description
\pagereference[bel-BY] % used for index
\startLN{Belarusan }\stopLN % LN: Language name
[bel] % ISO 639-3 code for this language
(Belarusian, Belorussian, Bielorussian, Byelorussian, White Russian,
White Ruthenian). 6,720,000 in Belarus (Johnstone and Mandryk 2001).
Population total all countries: 8,620,000.  Ethnic population:
9,051,080. Also in Azerbaijan, Canada, Estonia, Kazakhstan,
Kyrgyzstan, Latvia, Lithuania, Moldova, Poland, Russian Federation
```



**194** *Ethnologue* **Africa: Senegal**

Sine, Dyegueme (Gyegem), Niominka. The Niominka and Serere-Sine dialects mutually inherently intelligible. *Lg Use:* Official language. National language. *Lg Dev:* Literacy rate in L1: Below 1%. Bible: 2008. *Writing:* Arabic script. Latin script. *Other:* 'Sereer' is their name for themselves. Traditional religion, Muslim, Christian. *Map:* 725:28.

**Soninke** [snk] (Marka, Maraka, Sarahole, Sarakole, Sarangkolle, Sarawule, Serahule, Serahuli, Silabe, Toubakai, Walpre). 250,000 in Senegal (2007 LeClerc). North and south of Bakel along Senegal River. Bakel, Ouaoundé, Moudéri, and Yaféra are principal towns. *Dialects:* Azer (Adjer, Aser), Gadyaga. *Lg Use:* Official language. National language. Also use French, Bambara [bam], or Fula [fub]. *Lg Dev:* Literacy rate in L1: Below 1%. *Other:* The Soninke trace their origins back to the Eastern dialect area of Mali (Kinbakka), whereas the northeastern group in Senegal is part of the Western group of Mali (Xenqenna). Thus, significant differences exist between the dialects of the 2 geographical groups of Soninke in Senegal. Muslim. See main entry under Mali. *Map:* 725:29.

**Wamey** [cou] (Conhague, Coniagui, Koniagui, Konyagi, Wamei). 18,400 in Senegal (2007), decreasing. Population total all countries: 23,670. Southeast and central along Guinea border, pockets, usually beside Pulaar [fuc]. Also in Guinea. *Class:* Niger-Congo, Atlantic-Congo, Atlantic, Northern, Eastern Senegal-Guinea, Tenda. *Lg Use:* Neutral attitude. Also use Pulaar [fuc]. *Lg Dev:* Literacy rate in L1: Below 1%. *Writing:* Latin script. *Other:* Konyagi is the ethnic name. Agriculturalists; making wine, beer; weaving bamboo mats. Traditional religion, Christian. *Map:* 725:30.

**Wolof** [wol] (Ouolof, Volof, Walaf, Waro-Waro, Yallof). 3,930,000 in Senegal (2006). Population total all countries: 3,976,500. West and central, Senegal River left bank to Cape Vert. Also in France, Gambia, Guinea-Bissau, Mali, Mauritania. *Class:* Niger-Congo, Atlantic-Congo, Atlantic, Northern, Senegambian, Fula-Wolof, Wolof. *Dialects:* Baol, Cayor, Dyolof (Djolof, Jolof), Lebou (Lebu), Jander. Different from Wolof of Gambia [wof]. *Lg Use:* Official language. National language. Language of wider communication. Main African language of Senegal. Predominantly urban. Also use French or Arabic. *Lg Dev:* Literacy rate in L1: 10%. Literacy rate in L2: 30%. Radio programs. Dictionary. Grammar. NT: 1988. *Writing:* Arabic script, Ajami style. Latin script. *Other:* 'Wolof' is their name for themselves. Muslim. *Map:* 725:32.

**Xasonga** [kao] (Kasonke, Kasso, Kasson, Kassonke, Khasonke, Xaasonga, Xaasongaxango, Xasonke). 9,010 in Senegal (2006). *Lg Dev:* Literacy rate in L1: Below 1%. *Other:* Muslim. See main entry under Mali (Xaasongaxango).

### Seychelles

Republic of Seychelles. 86,000. National or official languages: English, French, Seselwa Creole French. Includes Aldabra, Farquhar, Des Roches; 92 islands. Literacy rate: 62%–80%. Information mainly from D. Bickerton 1988; J. Holm 1989. Blind population: 150 (1982 WCE). The number of individual languages listed for Seychelles is 3. Of those, all are living languages.

**English** [eng]. 1,600 in Seychelles (1971 census). *Lg Use:* Official language. *Other:* Principal language of the schools. See main entry under United Kingdom.

**French** [fra]. 980 in Seychelles (1971 census). *Lg Use:* Official language. *Other:* Spoken by French settler families, 'grands blancs'. See main entry under France.

**Seselwa Creole French** [crs] (Creole, Ilois, Kreol, Seychelles Creole French, Seychellois Creole). 72,700 (1998). Ethnic population: 72,700. *Class:* Creole, French based. *Dialects:* Seychelles dialect reportedly used on Chagos Islands. Structural differences with Morisyen [mfe] are relatively minor. Low intelligibility with Réunion Creole [rcf]. *Lg Use:* Official language since 1977. All domains. Positive attitude. *Lg Dev:* Taught in primary schools. Radio programs. Dictionary. Grammar. NT: 2000. *Writing:* Latin script. *Other:* Fishermen. Christian.

### Sierra Leone

Republic of Sierra Leone. 5,586,000. National or official language: English. Literacy rate: 15%. Immigrant languages: Greek (700), Yoruba (3,800). Also includes languages of Lebanon, India, Pakistan, Liberia. Information mainly from D. Dalby 1962; TISSL 1995. Blind population: 28,000 (1982 WCE). Deaf institutions: 5. The number of individual languages listed for Sierra Leone is 25. Of those, 24 are living languages and 1 is a second language without mother-tongue speakers. See map on page 726.

**Bassa** [bsq]. 5,730 in Sierra Leone (2006). Freetown. *Other:* Traditional religion. See main entry under Liberia.

**Bom** [bmf] (Bome, Bomo, Bum). 5,580 (2006), decreasing. Along Bome River. *Class:* Niger-Congo, Atlantic-Congo, Atlantic, Southern, Mel, Bullom-Kissi, Bullom, Northern. *Dialects:* Lexical similarity: 66%–69% with Sherbro [bun] dialects, 34% with Krim [krm]. *Lg Use:* Shifting to Mende [men]. *Other:* Traditional religion.

**Bullom So** [buy] (Bolom, Bulem, Bullin, Bullun, Mandenyi, Mandingi, Mmani, Northern Bullom). 8,350 in Sierra Leone (2006). Coast from Guinea border to Sierra Leone River. Also in Guinea. *Class:* Niger-Congo, Atlantic-Congo, Atlantic, Southern, Mel, Bullom-Kissi, Bullom, Northern. *Dialects:* Mmani, Kafu. Bom is closely related. Little intelligibility with Sherbro, none with Krim. *Lg Use:* Shifting to Themne [tem]. *Lg Dev:* Bible portions: 1816. *Writing:* Latin script. *Other:* The people are intermarried with the Temne and the Susu. Traditional religion. *Map:* 726:1.

**English** [eng]. *Lg Use:* Official language. Used in administration, law, education, commerce. See main entry under United Kingdom.

**Gola** [gol] (Gula). 8,000 in Sierra Leone (1989 TISLL). Along the border and inland. *Dialects:* De (Deng), Managobla (Gobla), Kongbaa, Kpo, Senje (Sene), Tee (Tege), Toldil (Toodii). *Lg Use:* Shifting to Mende [men]. *Other:* Different from Gola [mzm] of Nigeria (dialect of Mumuye) or Gola [pbp] (Badyara) of Guinea-Bissau and Guinea. Muslim, Christian. See main entry under Liberia. *Map:* 726:4.

**Kisi, Southern** [kss] (Gissi, Kisi, Kissien). 85,000 in Sierra Leone (1995). *Lg Dev:* Literacy rate in L2: 3%. *Other:* Different from Northern Kissi [kqs]. Traditional religion, Muslim, Christian. See main entry under Liberia. *Map:* 726:13.

**Kissi, Northern** [kqs] (Gizi, Kisi, Kisie, Kissien). 40,000 in Sierra Leone (1991 LBT). *Dialects:* Liaro, Kama, Teng, Tung. *Lg Use:* Also use Krio [kri] or Mende [men]. *Other:* Traditional religion. See main entry under Guinea. *Map:* 726:11.

**Klao** [klu] (Klaoh, Klau, Kroo, Kru). 9,620 in Sierra Leone (2006). Freetown. Originally from Liberia. *Other:* Traditional religion. See main entry under Liberia.

**Kono** [kno] (Konnoh). 205,000 (2006). Northeast. *Class:* Niger-Congo, Mande, Western, Central-Southwestern, Central, Manding-Jogo, Manding-Vai, Vai-Kono. *Dialects:* Northern Kono (Sando), Central Kono (Fiama, Gbane, Gbane Kando, Gbense, Gorama Kono, Kamara, Lei, Mafindo, Nimi Koro, Nimi Yama, Penguia, Soa, Tankoro,

Figure 1. Example of page with language descriptions

```
(Europe), Tajikistan, Turkmenistan, Ukraine, United States, Uzbekistan.
\startLDitem{Class: }\stopLDitem % LDitem: Language description item
Indo-European, Slavic, East.
\startLDitem{Dialects: }\stopLDitem Northeast Belarusan (Polots,
Viteb-Mogilev), Southwest Belarusan (Grodnen-Baranovich,
Slutsko-Mozyr, Slutska-Mazyrski), Central Belarusan. Linguistically
between Russian and Ukrainian [ukr], with transitional dialects to both.
\startLDitem{Lg Use: }\stopLDitem National language.
\startLDitem{Lg Dev: }\stopLDitem Fully developed. Bible: 1973.
\startLDitem{Writing: }\stopLDitem Cyrillic script.
\startLDitem{Other: }\stopLDitem Christian, Muslim (Tatar).  }
\stopLaDes % end of Language Description
```

The styles for the different elements are defined using start-stop setups. One example is the style for the LDitem (Language Definition item) which was initially coded in this way:

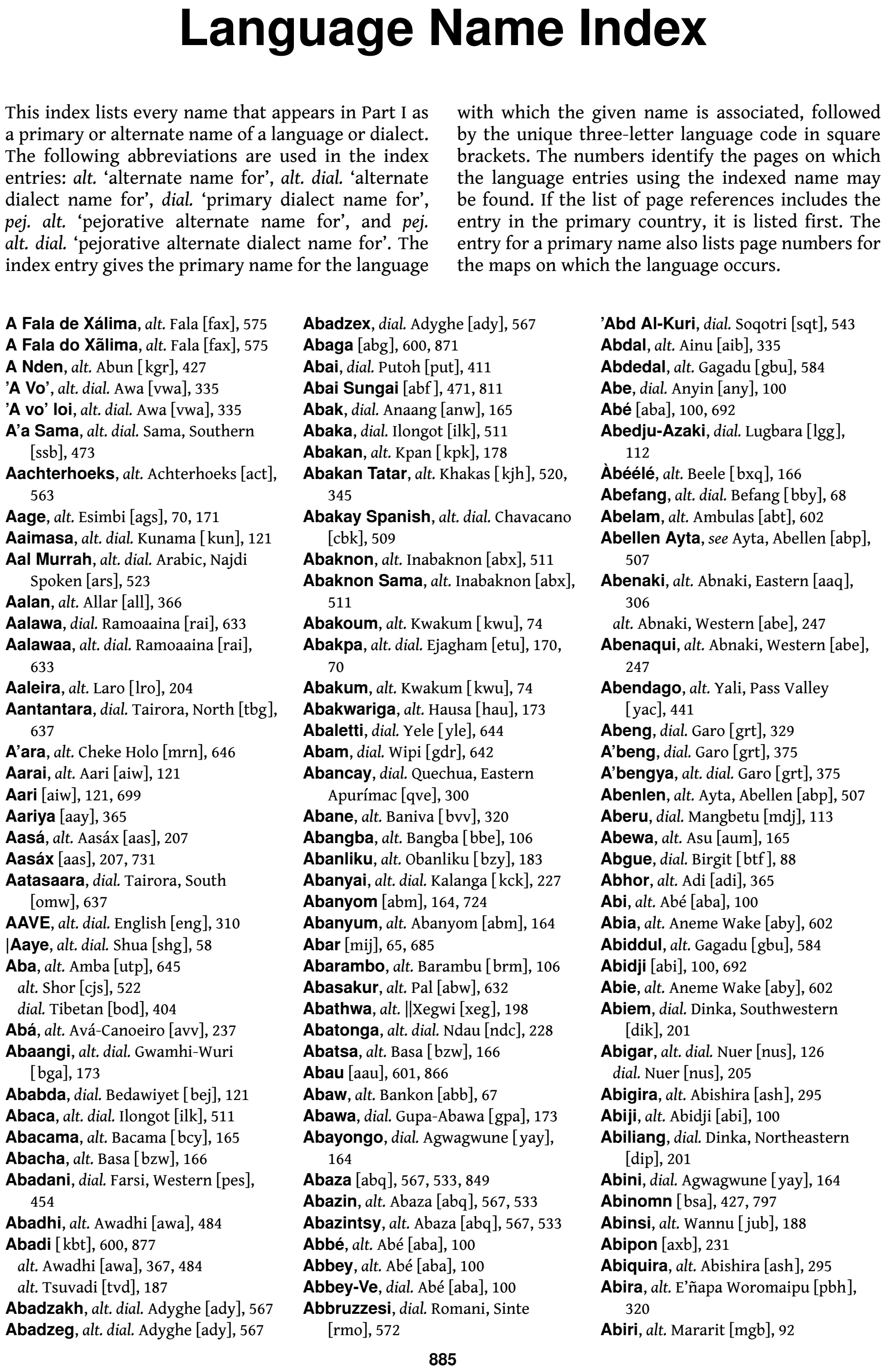
```
\definestartstop % Language Description Item Part 1 % deprecated code!
  [LDitem]
  [before={\switchtobodyfont[GentiumBookIt,\LDitemfontsize]},
   after={\switchtobodyfont[Gentium,\bodyfontpartone]}]
```

Eventually bodyfont switches were replaced by proper ConTEXt-style typescripts, but the idea remains the same: `\definestartstop[something][code here]` makes it possible to use the pair `\startsomething` and `\stopsomething`.

### Dynamic running header

As the example of the page with language descriptions (figure 1) shows the Country name is inserted in the header of the page, using the first country on a left page and the last country on the right page. The code used to do this is based on an example in `page-set.tex` in the ConTEXt distribution.

```
\definemarking[headercountryname]
\setupheadertexts[\setups{show-headercountryname-marks}]
 \startsetups show-headercountryname-first
 \getmarking[headercountryname][1][first] % get first marking
 \stopsetups
 \startsetups show-headercountryname-last
 \getmarking[headercountryname][2][last] % get last marking
 \stopsetups
\setupheadertexts[]
\setupheadertexts
  [\setups{text a}][]
  [][\setups{text b}] % setup header text (left and right pages)
\startsetups[text a] % setup contents page a
 \rlap{Ethnologue}
 \hfill
 {\pagenumber}
 \hfill
 \llap{\setups{show-headercountryname-last}}
\stopsetups
\startsetups[text b] % setup contents page b
  \rlap{\setups{show-headercountryname-first}}
  \hfill
  \pagenumber
  \hfill
  \llap{Ethnologue}
\stopsetups
```



# Language Name Index

This index lists every name that appears in Part I as a primary or alternate name of a language or dialect. The following abbreviations are used in the index entries: *alt.* 'alternate name for', *alt. dial.* 'alternate dialect name for', *dial.* 'primary dialect name for', *pej. alt.* 'pejorative alternate name for', and *pej. alt. dial.* 'pejorative alternate dialect name for'. The index entry gives the primary name for the language with which the given name is associated, followed by the unique three-letter language code in square brackets. The numbers identify the pages on which the language entries using the indexed name may be found. If the list of page references includes the entry in the primary country, it is listed first. The entry for a primary name also lists page numbers for the maps on which the language occurs.

**A Fala de Xálima**, *alt.* Fala [fax], 575
**A Fala do Xãlima**, *alt.* Fala [fax], 575
**A Nden**, *alt.* Abun [kgr], 427
**'A Vo'**, *alt. dial.* Awa [vwa], 335
**'A vo' loi**, *alt. dial.* Awa [vwa], 335
**A'a Sama**, *alt. dial.* Sama, Southern [ssb], 473
**Aachterhoeks**, *alt.* Achterhoeks [act], 563
**Aage**, *alt.* Esimbi [ags], 70, 171
**Aaimasa**, *alt. dial.* Kunama [kun], 121
**Aal Murrah**, *alt. dial.* Arabic, Najdi Spoken [ars], 523
**Aalan**, *alt.* Allar [all], 366
**Aalawa**, *dial.* Ramoaaina [rai], 633
**Aalawaa**, *alt. dial.* Ramoaaina [rai], 633
**Aaleira**, *alt.* Laro [lro], 204
**Aantantara**, *dial.* Tairora, North [tbg], 637
**A'ara**, *alt.* Cheke Holo [mrn], 646
**Aarai**, *alt.* Aari [aiw], 121
**Aari** [aiw], 121, 699
**Aariya** [aay], 365
**Aasá**, *alt.* Aasáx [aas], 207
**Aasáx** [aas], 207, 731
**Aatasaara**, *dial.* Tairora, South [omw], 637
**AAVE**, *alt. dial.* English [eng], 310
**|Aaye**, *alt. dial.* Shua [shg], 58
**Aba**, *alt.* Amba [utp], 645
*alt.* Shor [cjs], 522
*dial.* Tibetan [bod], 404
**Abá**, *alt.* Avá-Canoeiro [avv], 237
**Abaangi**, *alt. dial.* Gwamhi-Wuri [bga], 173
**Ababda**, *dial.* Bedawiyet [bej], 121
**Abaca**, *alt. dial.* Ilongot [ilk], 511
**Abacama**, *alt.* Bacama [bcy], 165
**Abacha**, *alt.* Basa [bzw], 166
**Abadani**, *dial.* Farsi, Western [pes], 454
**Abadhi**, *alt.* Awadhi [awa], 484
**Abadi** [kbt], 600, 877
*alt.* Awadhi [awa], 367, 484
*alt.* Tsuvadi [tvd], 187
**Abadzakh**, *alt. dial.* Adyghe [ady], 567
**Abadzeg**, *alt. dial.* Adyghe [ady], 567
**Abadzex**, *dial.* Adyghe [ady], 567
**Abaga** [abg], 600, 871
**Abai**, *dial.* Putoh [put], 411
**Abai Sungai** [abf], 471, 811
**Abak**, *dial.* Anaang [anw], 165
**Abaka**, *dial.* Ilongot [ilk], 511
**Abakan**, *alt.* Kpan [kpk], 178
**Abakan Tatar**, *alt.* Khakas [kjh], 520, 345
**Abakay Spanish**, *alt. dial.* Chavacano [cbk], 509
**Abaknon**, *alt.* Inabaknon [abx], 511
**Abaknon Sama**, *alt.* Inabaknon [abx], 511
**Abakoum**, *alt.* Kwakum [kwu], 74
**Abakpa**, *alt. dial.* Ejagham [etu], 170, 70
**Abakum**, *alt.* Kwakum [kwu], 74
**Abakwariga**, *alt.* Hausa [hau], 173
**Abaletti**, *dial.* Yele [yle], 644
**Abam**, *dial.* Wipi [gdr], 642
**Abancay**, *dial.* Quechua, Eastern Apurímac [qve], 300
**Abane**, *alt.* Baniva [bvv], 320
**Abangba**, *alt.* Bangba [bbe], 106
**Abanliku**, *alt.* Obanliku [bzy], 183
**Abanyai**, *alt. dial.* Kalanga [kck], 227
**Abanyom** [abm], 164, 724
**Abanyum**, *alt.* Abanyom [abm], 164
**Abar** [mij], 65, 685
**Abarambo**, *alt.* Barambu [brm], 106
**Abasakur**, *alt.* Pal [abw], 632
**Abathwa**, *alt.* ||Xegwi [xeg], 198
**Abatonga**, *alt. dial.* Ndau [ndc], 228
**Abatsa**, *alt.* Basa [bzw], 166
**Abau** [aau], 601, 866
**Abaw**, *alt.* Bankon [abb], 67
**Abawa**, *dial.* Gupa-Abawa [gpa], 173
**Abayongo**, *dial.* Agwagwune [yay], 164
**Abaza** [abq], 567, 533, 849
**Abazin**, *alt.* Abaza [abq], 567, 533
**Abazintsy**, *alt.* Abaza [abq], 567, 533
**Abbé**, *alt.* Abé [aba], 100
**Abbey**, *alt.* Abé [aba], 100
**Abbey-Ve**, *dial.* Abé [aba], 100
**Abbruzzesi**, *dial.* Romani, Sinte [rmo], 572
**'Abd Al-Kuri**, *dial.* Soqotri [sqt], 543
**Abdal**, *alt.* Ainu [aib], 335
**Abdedal**, *alt.* Gagadu [gbu], 584
**Abe**, *dial.* Anyin [any], 100
**Abé** [aba], 100, 692
**Abedju-Azaki**, *dial.* Lugbara [lgg], 112
**Àbéélé**, *alt.* Beele [bxq], 166
**Abefang**, *alt. dial.* Befang [bby], 68
**Abelam**, *alt.* Ambulas [abt], 602
**Abellen Ayta**, *see* Ayta, Abellen [abp], 507
**Abenaki**, *alt.* Abnaki, Eastern [aaq], 306
*alt.* Abnaki, Western [abe], 247
**Abenaqui**, *alt.* Abnaki, Western [abe], 247
**Abendago**, *alt.* Yali, Pass Valley [yac], 441
**Abeng**, *dial.* Garo [grt], 329
**A'beng**, *dial.* Garo [grt], 375
**A'bengya**, *alt. dial.* Garo [grt], 375
**Abenlen**, *alt.* Ayta, Abellen [abp], 507
**Aberu**, *dial.* Mangbetu [mdj], 113
**Abewa**, *alt.* Asu [aum], 165
**Abgue**, *dial.* Birgit [btf], 88
**Abhor**, *alt.* Adi [adi], 365
**Abi**, *alt.* Abé [aba], 100
**Abia**, *alt.* Aneme Wake [aby], 602
**Abiddul**, *alt.* Gagadu [gbu], 584
**Abidji** [abi], 100, 692
**Abie**, *alt.* Aneme Wake [aby], 602
**Abiem**, *dial.* Dinka, Southwestern [dik], 201
**Abigar**, *alt. dial.* Nuer [nus], 126
*dial.* Nuer [nus], 205
**Abigira**, *alt.* Abishira [ash], 295
**Abiji**, *alt.* Abidji [abi], 100
**Abiliang**, *dial.* Dinka, Northeastern [dip], 201
**Abini**, *dial.* Agwagwune [yay], 164
**Abinomn** [bsa], 427, 797
**Abinsi**, *alt.* Wannu [jub], 188
**Abipon** [axb], 231
**Abiquira**, *alt.* Abishira [ash], 295
**Abira**, *alt.* E'ñapa Woromaipu [pbh], 320
**Abiri**, *alt.* Mararit [mgb], 92

885

Figure 2. Start of the language name index

### Index

Since all the data for this publication comes from a database it was easy to compile a list of index items from that data. Page numbers were resolved using ConTEXt's internal referencing system. The data contains references using three letter ISO code for language and a two letter country code like this:

```
\pagereference[bel-BY] % ISO code - country code
```

In the file with the index data this reference is linked to an index item:

```
Belarusan [bel], \at[bel-BY]
```

The code `[bel-BY]` is automatically replaced by the right page number(s) producing the correct entry in the index:

Belarusan [bel], 32, 224

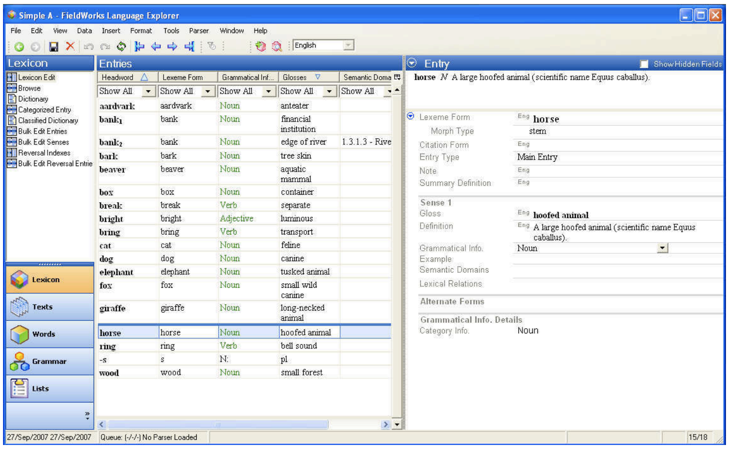
Since the language name index (the biggest index) contains more than 100.000 references it can be imagined that typesetting this publication in one run was pushing the limits of TEX. This is the first time that ConTEXt is used to typesetting this publication. The previous version was produced using Ventura but when that program was replaced by InDesign there were some questions about the way in which InDesign works with the automatically generated data. TEX seemed to be the right tool to use for this project and it sparked renewed interest in the use of TEX for other data-intensive publications like dictionaries.

## Exploring language

Counting languages is not the only way to collect language data: many linguists move into a language group and take a closer look at the different parts of the actual languages. Some linguists focus on the sounds of a language, others analyse the sentence structure or the way in which language is used in specific communication processes. The collected data is stored in a special database program called FieldWorks. FieldWorks runs on Windows only (though a Linux port is work in progress) and it is a free download from the SIL website[2]. FieldWorks is actually a suite of programs consisting of Data Notebook, Language Explorer and WorldPad. FieldWorks Data Notebook is used for anthropological observations. FieldWorks WorldPad is a 'world ready' text editor with some special script support (including Graphite[3]). FieldWorks Language Explorer (FLEx) is used to store all sorts of language related data. It is basically a complex database program with a couple of linguistics related tools. FLEx contains a lexicon for storing data related to words, meaning(s), grammatical information about words and translations in other languages. Another part of FLEx is the interlinear tool which makes it possible to take a text in one language and to give a 'word for word translation' in another language, for example as a way to discover grammatical structures. FLEx comes with a grammar tool to facilitate the analysis and description of the grammar of a language. Since all language data is stored in the same database there are some interesting possibilities to integrate the language data and analysis tools.

## Dictionaries

Once a field linguist has collected a certain amount of data he can start to think about the production of a word list or a real dictionary. To facilitate this a team of programmers has made tool called 'Dictionary Express'. This tool allows for the easy production of dictionaries based on data available in the FLEx database. The user of FLEx gets a menu option 'Print dictionary' and is presented with small window to enter some layout options. Behind the scenes one of two output paths is used: one is based on the use of an OpenOffice document template and another one uses XeTEX and ConTEXt to typeset the dictionary. XeTEX was chosen because of the requirement to facilitate the



| Headword | Lexeme Form | Grammatical Inf... | Glosses | Semantic Doma |
|---|---|---|---|---|
| Show All | Show All | Show All | Show All | Show All |
| aardvark | aardvark | Noun | anteater | |
| bank₁ | bank | Noun | financial institution | |
| bank₂ | bank | Noun | edge of river | 1.3.1.3 - Rive |
| bark | bark | Noun | tree skin | |
| beaver | beaver | Noun | aquatic mammal | |
| box | box | Noun | container | |
| break | break | Verb | separate | |
| bright | bright | Adjective | luminous | |
| bring | bring | Verb | transport | |
| cat | cat | Noun | feline | |
| dog | dog | Noun | canine | |
| elephant | elephant | Noun | tusked animal | |
| fox | fox | Noun | small wild canine | |
| giraffe | giraffe | Noun | long-necked animal | |
| horse | horse | Noun | hoofed animal | |
| ring | ring | Verb | bell sound | |
| -s | s | N: | pl | |
| wood | wood | Noun | small forest | |

horse *N* A large hoofed animal (scientific name Equus caballus).

| | | |
|---|---|---|
| Lexeme Form | Eng | **horse** |
| Morph Type | | stem |
| Citation Form | Eng | |
| Entry Type | | Main Entry |
| Note | Eng | |
| Summary Definition | Eng | |
| **Sense 1** | | |
| Gloss | Eng | **hoofed animal** |
| Definition | Eng | A large hoofed animal (scientific name Equus caballus). |
| Grammatical Info. | | Noun |
| Example | | |
| Semantic Domains | | |
| Lexical Relations | | |
| **Alternate Forms** | | |
| **Grammatical Info. Details** | | |
| Category Info. | | Noun |

27/Sep/2007 27/Sep/2007 Queue: (-/-/-) No Parser Loaded 15/18

Figure 3. FieldWorks Language Explorer main window

use of the Graphite smart font technology used for the correct rendering of complex non-roman script fonts in use in some parts of the world (see footnote 2). The use of XeTeX does of course mean that we use ConTeXt MkII.

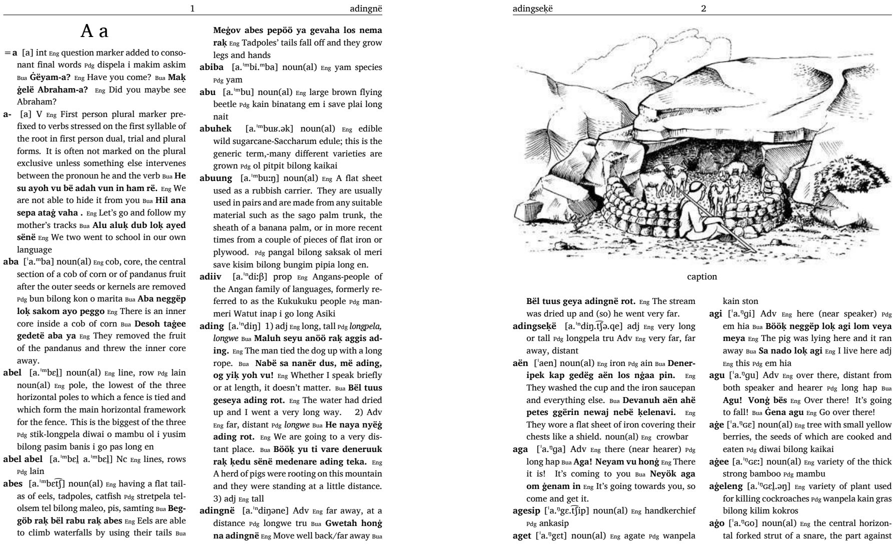
All data is available in an XML format and converted (using a purpose built converter) to a simple TeX-tagged format. A typical dictionary entry looks like this:

```
\Bentry
\Bhw{abel}\Ehw
\marking[guidewords]{abel}
\Bpr{a.ˈᵐbɛl̪}\Epr
\Bps{noun(al)}\Eps
\Blt{Eng}\Elt
\Bde{line, row}\Ede
\Blt{Pdg}\Elt
\Bde{lain}\Ede
\Bps{noun(al)}\Eps
\Blt{Eng}\Elt
\Bde{pole, the lowest of the three horizontal poles to which a fence is
tied and which form the main horizontal framework for the fence. This
is the biggest of the three}\Ede
\Eentry
```

The tags used in this data file include:

- □ headword (hw): this is the word that this particular entry is about,
- □ pronunciation (pr): the proper pronunciation of the word written using the International Phonetic Alphabet (IPA),
- □ part of speech (ps): the grammatical function of the word,
- □ language tag (lt): the language of the definition or example,
- □ definition (de): meaning of the headword,
- □ example (ex): example of the word used in a sentence.
- □ `\marking[guidewords]{}`: is used to put the correct guideword at the top of each page. (The code used here is inspired by the code used to put country name in the headers in the Ethnologue project.)

Currently most of the required features are implemented. This includes: font selection (including the use of Graphite fonts), basic dictionary layout and picture support. Some of these features are strait-forward and easy to implement. Other features such as picture support required more work e.g. page wide pictures keep floating to the next



1 adingnë

A a

**=a** [a] int Eng question marker added to consonant final words Pdg dispela i makim askim Bua **Ġëyam-a?** Eng Have you come? Bua **Maķ ġelë Abraham-a?** Eng Did you maybe see Abraham?

**a-** [a] V Eng First person plural marker prefixed to verbs stressed on the first syllable of the root in first person dual, trial and plural forms. It is often not marked on the plural exclusive unless something else intervenes between the pronoun he and the verb Bua **He su ayoh vu bë adah vun in ham rë.** Eng We are not able to hide it from you Bua **Hil ana sepa ataġ vaha .** Eng Let's go and follow my mother's tracks Bua **Alu aluķ dub loķ ayed sënë** Eng We two went to school in our own language

**aba** [ˈa.ᵐba] noun(al) Eng cob, core, the central section of a cob of corn or of pandanus fruit after the outer seeds or kernels are removed Pdg bun bilong kon o marita Bua **Aba neggëp loķ sakom ayo peggo** Eng There is an inner core inside a cob of corn Bua **Desoh taġee gedetë aba ya** Eng They removed the fruit of the pandanus and threw the inner core away.

**abel** [a.ˈᵐbɛl̪] noun(al) Eng line, row Pdg lain noun(al) Eng pole, the lowest of the three horizontal poles to which a fence is tied and which form the main horizontal framework for the fence. This is the biggest of the three Pdg stik-longpela diwai o mambu ol i yusim bilong pasim banis i go pas long en

**abel abel** [a.ˈᵐbɛl̪ a.ˈᵐbɛl̪] Nc Eng lines, rows Pdg lain

**abes** [a.ˈᵐbɛt͡ʃ] noun(al) Eng having a flat tail- as of eels, tadpoles, catfish Pdg stretpela telolsem tel bilong maleo, pis, samting Bua **Beggöb raķ bël rabu raķ abes** Eng Eels are able to climb waterfalls by using their tails Bua **Meġov abes pepöö ya gevaha los nema raķ** Eng Tadpoles' tails fall off and they grow legs and hands

**abiba** [a.ˈᵐbi.ᵐba] noun(al) Eng yam species Pdg yam

**abu** [a.ˈᵐbu] noun(al) Eng large brown flying beetle Pdg kain binatang em i save plai long nait

**abuhek** [a.ˈᵐbuʁ.ək] noun(al) Eng edible wild sugarcane-Saccharum edule; this is the generic term,-many different varieties are grown Pdg ol pitpit bilong kaikai

**abuung** [a.ˈᵐbu:ŋ] noun(al) Eng A flat sheet used as a rubbish carrier. They are usually used in pairs and are made from any suitable material such as the sago palm trunk, the sheath of a banana palm, or in more recent times from a couple of pieces of flat iron or plywood. Pdg pangal bilong saksak ol meri save kisim bilong bungim pipia long en.

**adiiv** [a.ˈⁿdi:β] prop Eng Angans-people of the Angan family of languages, formerly referred to as the Kukukuku people Pdg manmeri Watut inap i go long Asiki

**ading** [a.ˈⁿdiŋ] 1) adj Eng long, tall Pdg *longpela, longwe* Bua **Maluh seyu anöö raķ aggis ading.** Eng The man tied the dog up with a long rope. Bua **Nabë sa nanër dus, më ading, og yiķ yoh vu!** Eng Whether I speak briefly or at length, it doesn't matter. Bua **Bël tuus geseya ading rot.** Eng The water had dried up and I went a very long way. 2) Adv Eng far, distant Pdg *longwe* Bua **He naya nyëġ ading rot.** Eng We are going to a very distant place. Bua **Bööķ yu ti vare deneruuk raķ ķedu sënë medenare ading teka.** Eng A herd of pigs were rooting on this mountain and they were standing at a little distance. 3) adj Eng tall

**adingnë** [a.ˈⁿdiŋəne] Adv Eng far away, at a distance Pdg longwe tru Bua **Gwetah hoṅġ na adingnë** Eng Move well back/far away Bua

adingseķë 2

caption

**Bël tuus geya adingnë rot.** Eng The stream was dried up and (so) he went very far.

**adingseķë** [a.ˈⁿdiŋ.t͡ʃə.qe] adj Eng very long or tall Pdg longpela tru Adv Eng very far, far away, distant

**aën** [ˈaen] noun(al) Eng iron Pdg ain Bua **Deneripek kap gedëg aën los nġaa pin.** Eng They washed the cup and the iron saucepan and everything else. Bua **Devanuh aën ahë petes ggërin newaj nebë ķelenavi.** Eng They wore a flat sheet of iron covering their chests like a shield. noun(al) Eng crowbar

**aga** [ˈa.ᵑga] Adv Eng there (near hearer) Pdg long hap Bua **Aga! Neyam vu hoṅġ** Eng There it is! It's coming to you Bua **Neyök aga om ġenam in** Eng It's going towards you, so come and get it.

**agesip** [ˈa.ᵑgɛ.t͡ʃip] noun(al) Eng handkerchief Pdg ankasip

**aget** [ˈa.ᵑgɛt] noun(al) Eng agate Pdg wanpela kain ston

**agi** [ˈa.ᵑgi] Adv Eng here (near speaker) Pdg em hia Bua **Bööķ neggëp loķ agi lom veya meya** Eng The pig was lying here and it ran away Bua **Sa nado loķ agi** Eng I live here adj Eng this Pdg em hia

**agu** [ˈa.ᵑgu] Adv Eng over there, distant from both speaker and hearer Pdg long hap Bua **Agu! Voṅġ bës** Eng Over there! It's going to fall! Bua **Ġena agu** Eng Go over there!

**aġe** [ˈa.ᵑɢɛ] noun(al) Eng tree with small yellow berries, the seeds of which are cooked and eaten Pdg diwai bilong kaikai

**aġee** [a.ˈᵑɢɛ:] noun(al) Eng variety of the thick strong bamboo Pdg mambu

**aġeleng** [a.ˈᵑɢɛl̪.əŋ] Eng variety of plant used for killing cockroaches Pdg wanpela kain gras bilong kilim kokros

**aġo** [ˈa.ᵑɢo] noun(al) Eng the central horizontal forked strut of a snare, the part against

Figure 4. Sample double column dictionary layout

page. Since it is usually a good idea to separate form and content most of the layout related settings are not stored in the data file itself but in a separate settings file which is loaded at the start of the typesetting process. Examples of settings in this file include the fonts and the use of a double column layout. Default settings are used unless the user has specified different settings using the small layout options window at the start of the process.

Currently the test version of this ConTEXt-based system works with a stand alone ConTEXt-installation, using the 'minimals' distribution. One of the remaining challenges is to make a light weight, easy to install version of ConTEXt which can be included with the FieldWorks software. Since the main script used by ConTEXt Mark II is a Ruby script this requires dealing with (removing) the Ruby dependency. It is hoped that stripping the TEX-tree of all unused fonts and code will help too to reduce the space used by this tool. This is currently work in progress.

## Footnotes

1. Lewis, M. Paul (ed.), Ethnologue: Languages of the World, Sixteenth edition. Dallas, Tex.: SIL International (2009)
2. http://www.sil.org/computing/fieldworks/
3. http://scripts.sil.org/RenderingGraphite

Jelle Huisman
SIL International
Horsleys Green
High Wycombe
United Kingdom
HP14 3XL
jelle_huisman (at) sil (dot) org

# A network TEX Live installation
## at the University of Groningen

**Abstract**
This article describes a network TEX Live installation for Windows users and the context in which it operates.

**Keywords**
TEX Live, MiKTEX, installers, editors, roaming profiles, Windows Vista

Our university has a LAN-based TEX installation for Windows users. The current edition is based on TEX Live. After some historical notes, I discuss the computing environment at the university, the new TEX Live-based installation and issues with Windows Vista.

This article can be considered an update of a previous article[1] about the MiKTEX-based installation that I maintained for the university's economics department.

### Prehistory: 4TEX

Our department has had a network TEX installation for DOS/Windows users since the early nineties. It started out as a simple 4DOS menu for running TEX and associated programs. Later, it evolved into 4TEX, and was also made available to members of the Dutch-speaking TEX users group (the NTG) and others; see figure 1.

The final version was a Windows program, based on the same core as TEX Live. It included a vast array of utilities, some of them third-party or shareware.

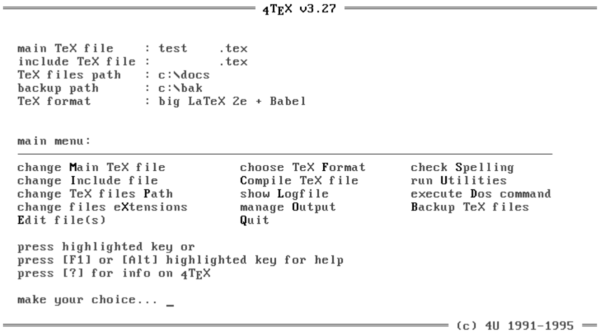
### A MiKTEX-based installation

When I took over in 2003, 4TEX was no longer being developed. We chose to make a fresh start, based on what was then available, and to limit ourselves to free software.

Modern LATEX editors such as TEXnicCenter can take care of running BIBTEX and MakeIndex, include spell checkers, and offer help in entering LATEX macros and in debugging. For many users, the editor is all they see from the TEX installation.

The good integration of MiKTEX as the TEX implementation and TEXnicCenter as editor and frontend was hard to argue with, so that was what we used.

For graphics support, there was a script to install WMF2EPS and its PostScript printer driver which did the real work in the background. For anything else, users had to look elsewhere.



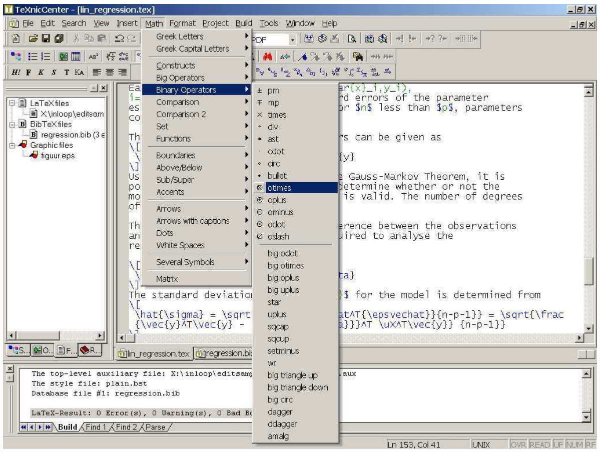
```
──────────────────────────── 4TEX v3.27 ──────────

main TeX file      : test     .tex
include TeX file   :          .tex
TeX files path     : c:\docs
backup path        : c:\bak
TeX format         : big LaTeX 2e + Babel


main menu:

change Main TeX file       choose TeX Format      check Spelling
change Include file        Compile TeX file       run Utilities
change TeX files Path      show Logfile           execute Dos command
change files eXtensions    manage Output          Backup TeX files
Edit file(s)               Quit

press highlighted key or
press [F1] or [Alt] highlighted key for help
press [?] for info on 4TEX

make your choice... _
──────────────────────────────── (c) 4U 1991-1995 ──
```

Figure 1. 4TEX main menu

### Evolution

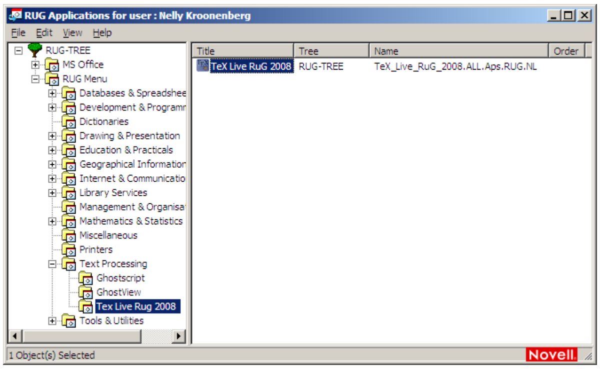
*Installer.* The original installer was a combined batch-file/Perl script, which also used some registry patches. This was replaced with a GUI installer based on NSIS, an open source installation system. Where possible, files and registry settings were generated during installation rather than copied from a prototype installation.

*Updated versions.* The editor and MiKTEX have been updated several times. This included a major change



```
The top-level auxiliary file: X:\inloop\editsam          .aux
The style file: plain.bst
Database file #1: regression.bib

LaTeX-Result: 0 Error(s), 0 Warning(s), 0 Bad Bo
```

Figure 2. TeXnicCenter as frontend to MiKTEX



| Title | Tree | Name | Order |
|---|---|---|---|
| TeX Live RuG 2008 | RUG-TREE | TeX_Live_RuG_2008.ALL.Aps.RUG.NL | |

Figure 3. The NAL application menu

in configuration strategy in MiKTEX from version 2.4 to version 2.5. I understand that there are again major changes with MiKTEX 2.8.

*CD edition.* A companion CD was created. At first this contained a set of downloaded installers with directions for use. As a result, I was asked time and again to install TEX on people's laptops. Therefore, a later version got a modified version of the network installer. Nowadays, the CD is offered as an ISO image in the root of the installation.

*Expanded user base.* The user base for our MiKTEX was expanded first with students, then with other departments.

*Standardization.* Around the time of the second MiKTEX edition, we also moved to standardized desktops and roaming profiles; see the next section.

*Graphics support.* WMF2EPS was dropped, in part because it was shareware and I didn't want to deal with licensing issues for the expanded user base, in part for technical reasons. In its place, I created EPSPDF to take care of converting and cropping arbitrary PostScript (print)files.

## The university network

For some time, we have had a centrally managed university-wide Novell network. Software and licenses are also centrally managed. There is a standardized Windows XP workstation. Standard software and additional software is installed from the network and where possible also run from the network. NAL (Novell Application Launcher) is the network component which takes care of this.

Figure 3 displays the NAL menu as seen from my computer at the university (the TEX Live entry merely points to the installation script).

Staff members can and do install software of their own on their local system if they want to. Students do not have this luxury. Some staff members download and install their own TEX.

The standard workstation is configured with *roaming profiles*, *i.e.* user configuration is mostly copied to the network on logout and copied back to the local system on login. Users see the same desktop on any client computer on the net, of course excepting software which staff members may have installed locally.

Roaming profile configuration should involve nothing local, unless it is copied to and from the network as part of the user profile. It should not require admin rights either. This is especially important for classroom computers and for students.

## TEX Live

In 2008 I got involved in the TEX Live project. I mostly worked on Windows support, keeping an eye on the needs of our university installation.

I have done only a little work on the 2009 edition. However, other team members now also have Windows virtual machines for testing, and we have been joined by a real Windows user, Tomasz Trzeciak. He proved to us that DOS batchfiles aren't quite as lame as we thought they were.

Compared to MiKTEX 2.5, TEX Live is a lot simpler to turn into a network installation:[2] in good Unix tradition, TEX Live uses environment variables and plain text files for configuration.

*Relocatable.* An important function of configuration is telling programs where they can find the files they need. Normally, TEX Live puts only relative paths in the configuration files. Programs can combine these relative paths with their own location to determine absolute paths. With this strategy, configuration files can stay the same if the installation as a whole is transferred to another place.

*Batteries included.* TEX Live contains copies of Perl and Ghostscript for Windows. This puts Windows on a more equal footing with Unix/Linux with regard to all the scripted utilities that are part of a typical TEX installation.

Both the included Ghostscript and the included Perl are hidden, *i.e.* TEX Live knows that they are there, but the rest of the system doesn't. They are not on the search path, and there are no environment variables or registry settings created for them. Therefore, they shouldn't interfere with pre-installed copies. The only disadvantage is the disk space they take up. But this is hardly significant with today's hard disk sizes.

## Creating the installation

I emulate the university networking setup by setting up a Samba server on my Linux machine. Its clients are virtual machines.

Samba has been set up for roaming profiles. There is a share for the profiles, an X:-share for home directories and a Z:-share with applications, in exactly the same layout as the university.

I install TEX Live into the right position on the Z:-share by running the installer on my own Linux system. I select binaries for both Linux and Windows.

I switch between this and my regular installation simply by changing environment variables, for which I have a small shell function. This lets me do much testing and all maintenance from Linux. I explained already that configuration files don't depend on the location of the installation. So it doesn't matter that, seen from Linux, the installation is in a totally different place than it is as seen from Windows.

I populate the texmf-local directory tree with the university house style and some legacy packages. It was almost a straight copy of the corresponding local tree from the previous MiKTEX-based installation. For students, there is no need for a local tree.

## The 2009 installer

The network installer doesn't have to do much: there are hardly any options, it doesn't have to download and install packages, it just has to add TEX Live to the search path, create some shortcuts, register an uninstaller and optionally create some file associations.

For most of this, it can use the library functions of the installer and the TEX Live Manager, both of which are written in Perl.

The following code adds TEX Live to the search path and creates some menu shortcuts:

```
#!/usr/bin/env perl
BEGIN {
  require "tlmgr.pl";
}

# Only make user-level changes even if admin
$opts{'w32mode'} = 'user';

# Note. The action_... functions read
# their arguments from @ARGV.

# Add TeX Live to path
unshift @ARGV, 'add';
action_path();

# create some shortcuts
unshift @ARGV, 'install', 'shortcut',
 'dviout.win32', 'texworks', 'texlive-en',
 'tlpsv.win32';
action_postaction();
```

File associations can be done similarly. A corresponding uninstaller script:

```
BEGIN {
  require "tlmgr.pl";
}
$opts{'w32mode'} = 'user';

# remove shortcuts
unshift @ARGV, 'remove', 'shortcut',
 'dviout.win32', 'texworks', 'texlive-en',
 'tlpsv.win32';
action_postaction();

# Remove TeX Live from path
unshift @ARGV, 'remove';
action_path();
```

*Registering and unregistering the uninstaller.* However, it is a bit more complicated to register one's custom uninstaller. The TEX Live modules in tlpkg/TeXLive, and the modules they load, contain everything needed, but the interface is comparatively low-level. Here is the code, for what it is worth:

```
# don't need to re-require modules but
# do need to re-import names
Win32::TieRegistry->import(qw($Registry));
$Registry->Delimiter('/');
$Registry->ArrayValues(0);
$Registry->FixSzNulls(1);

# register uninstaller. Failure not fatal.
my $Master_bsl = $Master;
$Master_bsl =~ s,/,\\,g;

my $rootkey = $Registry -> Open("CUser",
 {Access =>
   Win32::TieRegistry::KEY_ALL_ACCESS()});
my $k;
if ($rootkey) {
 $k = $rootkey->CreateKey(
  "software/microsoft/windows/" .
  "currentversion/uninstall/OurTeXLive/");
 if ($k) {
  $k->{"/DisplayName"} = "OurTeXLive 2009";
  $k->{"/UninstallString"} =
    "\"$Master_bsl\\w32unclient.bat\"";
  $k->{'/DisplayVersion'} = "2009";
  $k->{'/URLInfoAbout'} =
    "http://ourwebsite.edu/ourtexlive";
 }
}
```

```
warn "Failed to register uninstaller\n"
 unless $k;
```

and for unregistering the uninstaller:

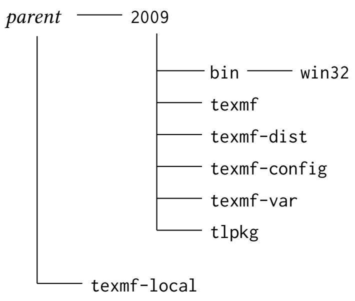
```
my $rootkey = $Registry -> Open("CUser",
 {Access =>
   Win32::TieRegistry::KEY_ALL_ACCESS()});
if ($rootkey) { # otherwise fail silently
 my $k = $rootkey->Open(
  "software/microsoft/windows/" .
  "currentversion/uninstall/");
 TeXLive::TLWinGoo::reg_delete_recurse($k,
  'OurTexLive/') if $k;
}
```

Prototype installer scripts are available at `http://tug.org/texlive/w32client.html`.

*ZENWorks.* Novell has a tool ZENworks for repackaging applications for NAL. It tracks changes in the registry and the filesystem during installation. The ZEN-generated installer of the repackaged application duplicates those changes. However, for me it was more practical to use a Perl script, and I am grateful to the ICT people that they let me.

## Directory layout

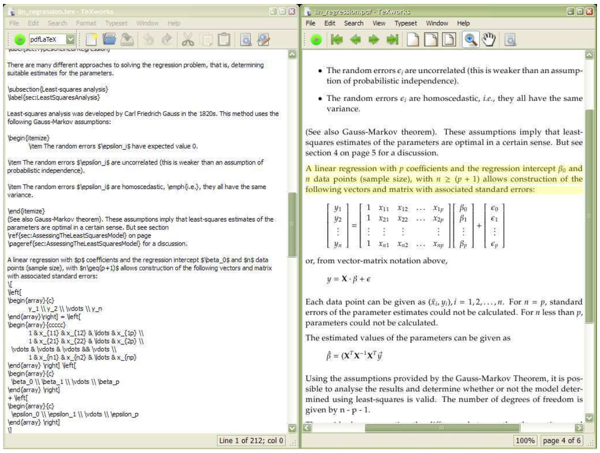
We assume the standard TeX Live directory layout, with texmf-local one level higher than the other trees:



It is possible to choose another layout, *e.g.* without the year level, but then the components of TeX Live need a bit more help to find their support files.

## Batch wrappers

We also need a wrapper batchfile to make sure that the Perl from TeX Live is used rather than a locally installed Perl, and to take care that `tlmgr.pl`, the TeX Live Manager Perl script, is found. This file is used as a library by our custom installer. Below is a bare-bones wrapper batchfile; in the standard TeX Live we use much more involved wrappers, with various safeguards.[3]


lin_regression.tex - TeXworks
File Edit Search Format Typeset Window Help
pdfLaTeX
Line 1 of 212; col 0
lin_regression.pdf - TeXworks
File Edit Search View Typeset Window Help
100%
page 4 of 6

```
\label{sec:TypesOfLinearRegression}

There are many different approaches to solving the regression problem, that is, determining
suitable estimates for the parameters.

\subsection{Least-squares analysis}
\label{sec:LeastSquaresAnalysis}

Least-squares analysis was developed by Carl Friedrich Gauss in the 1820s. This method uses the
following Gauss-Markov assumptions:

\begin{itemize}
	\item The random errors $\epsilon_i$ have expected value 0.

\item The random errors $\epsilon_i$ are uncorrelated (this is weaker than an assumption of
probabilistic independence).

\item The random errors $\epsilon_i$ are homoscedastic, \emph{i.e.}, they all have the same
variance.

\end{itemize}
(See also Gauss-Markov theorem). These assumptions imply that least-squares estimates of the
parameters are optimal in a certain sense. But see section
\ref{sec:AssessingTheLeastSquaresModel} on page
\pageref{sec:AssessingTheLeastSquaresModel} for a discussion.

A linear regression with $p$ coefficients and the regression intercept $\beta_0$ and $n$ data
points (sample size), with $n\geq(p+1)$ allows construction of the following vectors and matrix
with associated standard errors:
\[
\left[
\begin{array}{c}
	y_1 \\ y_2 \\ \vdots \\ y_n
\end{array}\right] = \left[
\begin{array}{ccccc}
	1 & x_{11} & x_{12} & \ldots & x_{1p} \\
	1 & x_{21} & x_{22} & \ldots & x_{2p} \\
 \vdots & \vdots & \vdots && \vdots \\
	1 & x_{n1} & x_{n2} & \ldots & x_{np}
\end{array} \right] \left[
\begin{array}{c}
 \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p
\end{array} \right]
+ \left[
\begin{array}{c}
 \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_p
\end{array} \right]
\]
```

- The random errors $\epsilon_i$ are uncorrelated (this is weaker than an assumption of probabilistic independence).
- The random errors $\epsilon_i$ are homoscedastic, *i.e.*, they all have the same variance.

(See also Gauss-Markov theorem). These assumptions imply that least-squares estimates of the parameters are optimal in a certain sense. But see section 4 on page 5 for a discussion.

A linear regression with $p$ coefficients and the regression intercept $\beta_0$ and $n$ data points (sample size), with $n \geq (p + 1)$ allows construction of the following vectors and matrix with associated standard errors:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_p \end{bmatrix}$$

or, from vector-matrix notation above,

$$y = \mathbf{X} \cdot \beta + \epsilon$$

Each data point can be given as $(\vec{x}_i, y_i), i = 1, 2, \ldots, n$. For $n = p$, standard errors of the parameter estimates could not be calculated. For $n$ less than $p$, parameters could not be calculated.

The estimated values of the parameters can be given as

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X}^{-1}\mathbf{X}^T\vec{y}$$

Using the assumptions provided by the Gauss-Markov Theorem, it is possible to analyse the results and determine whether or not the model determined using least-squares is valid. The number of degrees of freedom is given by n - p - 1.

Figure 4. TeXworks for Windows is included in TeX Live

```
set this=%~dp0
rem Use TL Perl
path %this%tlpkg\tlperl\bin;%this%bin\win32;
    %path%
rem (one line)
set PERL5LIB=%this%tlpkg\tlperl\lib;
    %this%tlpkg;%this%texmf\scripts\texlive
rem (one line)

rem Start Perl script of the same name
perl "%~dpn0" %*
rem Give user opportunity to scan output msgs
pause
```

Note the first line, where the batchfile finds its own directory: `set this=%~dp0`. This syntax has been available since Windows NT.

The `w32client` and `w32unclient` scripts assume that they are in the root of the installation, *i.e.* in <*parent*>`/2009`. However, this is easy to change.

## Getting TeX Live on the network

The first step was asking the ICT department for a directory to install into, preferably with write access for me, so that I can do maintenance without having to involve ICT every time.

The next step was copying everything to that directory. For transport, I nowadays make a giant zip of the installation and put it on a USB stick.

Finally, the installer is integrated into the Novell NAL menu; see figure 3 on page 87. This is done by ICT staff.

The installer places the TeX Live menu itself under Start / Programs, just as the native installer does.

For maintenance, I used to do small changes manually and do larger changes by wholesale replacement. For the future, rsync looks like a better way.

## Additional software

TEX Live by itself is pretty complete. For Windows, it includes the TEXworks cross-platform editor (figure 4), and the PS_View PostScript viewer, which can also read pdf. As mentioned earlier, it also contains private copies of Unix mainstays such as Perl and Ghostscript that many TEX Live components depend upon.

Nevertheless, some additions would be desirable: another editor besides TEXworks, more graphics support, maybe a bibliography editor.

But there are requirements for such add-ons:

□ Free (as in beer)
□ Per-user configuration
□ Usable for non-geeks

Several programs looked interesting, but didn't meet these requirements or had other problems. They include alternative LATEX editors TEXmaker and WinShell, bibliography editors JabRef (Java-based) and BibEdt, and a couple of draw programs, IPE and TpX. So I followed the example of previous TEX Live DVDs and put installers for them in a `support` subdirectory. Frankly, I don't know whether anybody has made use of these installers.

*Editor.* For 2008, I decided to stick with TEXnicCenter as editor. I wrote some fairly elaborate code to configure it for TEX Live, since the automatic configuration didn't work as nicely for TEX Live as it did for MiKTEX. I also looked at TEXmaker. It would have been much easier to configure, but at that time it still lacked some important features.

For the 2009 release I'll keep TEXnicCenter, if only because many users dislike change, but I'll also include TEXworks, which is already part of standard TEX Live.

*Documentation.* The TEX Live menu contains various shortcuts to manuals, such as the UK FAQ and the 'not so short introduction'. There are also links to the CTAN catalogue and to my own web page for this installation, `http://tex.aanhet.net/miktex/` (!).

## Vista and Windows 7

Strictly speaking, this topic doesn't belong here: the network installation only targets XP machines. However, for the standard TEX Live we had to make sure it would work properly with Vista and Windows 7. Testing this, we ran into some interesting problems.

*UAC.* Vista introduced User Account Control, or UAC in short. This means, among other things, that only administrators are allowed to install software under Program Files, and only administrators are allowed to change the more important system settings in the registry.

A new slightly annoying twist is that even administrators don't have these privileges automatically. To start a program with admin privileges, you can right-click the shortcut, which usually gives you an option 'Run as administrator'. An administrator has to confirm his intentions, a non-administrator has to provide administrator credentials at this point.

*Virtualization.* But there is another, more insidious twist: Vista/Win7 may guess that a program needs administrative privileges, *e.g.* because it has 'install' or 'setup' in its name. If such a program wasn't started with administrative privileges, Vista may fake them. In particular, attempts to write to Program Files might result in writings to *user*`\appdata\local\virtualstore`. For registry access, similar virtualization might be applied.[4]

Installing TEX Live with real admin privileges and adding packages with faked admin privileges is not healthy for a TEX Live installation.

This compatibility mode can be avoided by the addition of a *manifest* which is a bit of XML that explicitly tells Windows under which privileges the program needs to be run. The options are (explanations literally taken from `msdn.microsoft.com`):

**asInvoker** The application runs with the same access token as the parent process.

**highestAvailable** The application runs with the highest privileges the current user can obtain.

**requireAdministrator** The application runs only for administrators and requires that the application be launched with the full access token of an administrator.

This XML can be embedded into the binary or added as a separate file with the same name as the program, but with `.manifest` appended.

This is our manifest file for the Windows Perl executable:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
    manifestVersion="1.0">
  <assemblyIdentity
    version="1.0.0.0"
    processorArchitecture="*"
    name="perl.exe"
    type="win32"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level="asInvoker"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

We believe that, with the addition of a couple of manifest files and some tests on admin privileges, TEX Live 2009 has become Vista-safe.

## Conclusion

So you see that maintaining a TEX installation has little to do with TEX, slightly more with programming, but is mostly a matter of tying disparate pieces together.

## Notes

1. MAPS 33 pp. 59–64 and TUGboat 27:1 pp. 22–27.
2. MiKTEX 2.8 may be easier to deal with, but I didn't check this out.
3. TEX Live even uses a second, binary wrapper around the batch wrapper because some programs handle batchfiles badly.
4. This only happens on 32-bit Vista/Win7.

Siep Kroonenberg
n.s.kroonenberg@rug.nl

# Using TEX's language within a course about functional programming

**Abstract**
We are in charge of a teaching unit, entitled *Advanced Functional Programming*, for 4th-year university students in Computer Science. This unit is optional within the curriculum, so students attending it are especially interested in programming. The main language studied in this unit is Scheme, but an important part is devoted to general features, e.g., lexical vs. dynamic scoping, limited vs. unlimited extent, call by value vs. call by name or need, etc. As an alternative to other programming languages, TEX allows us to show a language where dynamic and lexical scoping—\def vs. \edef—coexist. In addition, we can show how dynamic scoping allows users to customise TEX's behaviour. Other commands related to strategies are shown, too, e.g., \expandafter, \noexpand. More generally, TEX commands are related to macros in more classical programming languages, and we can both emphasise difficulty related to macros and show non-artificial examples. So TEX is not our unit's main focus, but provides significant help to illustrate some difficult notions.

**Keywords**
Functional programming, TEX programming, lexical vs. dynamic scope, macros, evaluation strategies.

## Introduction

If we consider *programming* in TEX [17], we have to admit that this language is old-fashioned, and programs are often viewed as rebuses, as shown in the *Pearls of TEX programming* demonstrated at BachoTEX conferences.[1] Some interesting applications exemplifying this language can be found in [19, 30], but as noticed in [5], 'some of these programming tricks are ingenious and even elegant. However [...] it is time for a change'.

So, at first glance, it may be strange to use some examples of TEX programming within a present-day course devoted to *Functional programming*. Let us recall that this programming paradigm treats computation as the evaluation of mathematical functions, avoiding state and mutable data as much as possible. Functional programming emphasises functions' application, whereas *imperative programming*—the paradigm implemented within more 'traditional' languages, such as C [16]—emphasises changes in state. Many universities include courses about functional programming, examples being reported in [35]. Besides, such languages are sometimes taught as *first* programming languages, according to an approach comparable to [1, 8, 32] in the case of Scheme.

Let us remark that some tools developed as part of TEX's galaxy have already met functional programming: cl-bibtex [18], an extension of BibTEX—the bibliography processor [26] usually associated with the LATEX word processor [20]—is written using ANSI[2] Common Lisp [7]; xindy, a multilingual index processor for documents written using LATEX [24, § 11.3] is based on Common Lisp, too; BibTEX2HTML [4], a converter from .bib format—used by the bibliography database files of BibTEX—to HTML,[3] is written in CAML[4] [21]; MlBibTEX,[5], a re-implementation of BibTEX focusing on multilingual features [11] is written in Scheme [15]; as another example, Haskell[6] [28] has been used in [38]; last but not at least, there were proposals for developing NTS[7]—a re-implementation of TEX—using CLOS,[8] an object-oriented system based on Common Lisp [39].

The unit we mentioned above is entitled *Advanced Functional Programming*.[9] It is an optional unit for 4th-year university students in Computer Science, part of the curriculum proposed at the University of Franche-Comté, at the Faculty of Science and Technics, located at Besançon, in the east of France. Most of these students already know a functional programming language: Scheme, because they attended a unit introducing this language in the 2nd academic year in Computer Science.[10] Other students, who attended the first two university years at Belfort, know CAML. So this unit is not an introductory course; we delve thoroughly into functional programming.

In the next section, we expose the 'philosophy' of our unit. Then we summarise the features of TEX that are useful within this unit and discuss our choice of TEX. Reading this article only requires basic knowledge of programming; readers who would like to go thoroughly into Scheme constructs we have used throughout our examples can refer to [32], very didactic. Of course, the indisputable reference about TeX commands is [17].

```
(define (factorial x)
  ;; Returns x! if x is a natural number, the ‘false’
  ;; value otherwise.
  (and (integer? x) (not (negative? x))
       (let tr-fact ((counter x)
                     (acc 1))
         ;; Returns acc * counter!.
         (if (zero? counter)
             acc
             (tr-fact (- counter 1)
                      (* acc counter))))))
```

Figure 1. The factorial function, written using Scheme.

## Our unit's purpose

Functional programming languages have a common root in the $\lambda$-calculus, a formal system developed in the 1930s by Alonzo Church to investigate function definition, function application, and recursion [3]. However, these programming languages are very diverse, some—e.g., the Lisp dialects[11]—are dynamically typed,[12] some—e.g., Standard ML[13] [27], CAML, Haskell—are strongly typed[14] and include a *type inference mechanism*: end-users do not have to make precise the types of the variables they use, they are inferred by the type-checker; in practice, end-users have to conceive a program using a strongly typed approach because if the type-checker does not succeed in associating a type with an expression, this expression is proclaimed incorrect. As examples, Fig. 1 (resp. 2) show how to program the factorial function in Scheme (resp. Common Lisp). In both cases, the factorial function we give can be applied to any value, but returns the factorial of this value only if it is a non-negative integer, otherwise, the result is the ‘false’ value. Fig. 3 gives the same function in Standard ML: it can only be applied to an integer, as reported by the type-checker (see the line beginning with ‘>’).

A course explaining the general principles of functional programming with an overview of some existing functional programming languages would be indigestible for most students, since they could only with difficulty become familiar with several languages, due to the amount of time that can be allocated to each unit. In addition, theoretical notions without practice would not be very useful. So, our unit's first part is devoted to the $\lambda$-calculus' bases [10]. Then, all the practical exercises are performed with only one language, Scheme, which most students already know. Besides, this unit ends with some advanced features of this language: delayed evaluation, continuations, hygienic macros [9]. In addition, this choice allows us to perform a demonstration of DSSSL[15] [13], initially designed as the stylesheet language for SGML[16] texts. These students attended a unit about XML and XSLT[17] [36] the year before, and DSSSL—which may be viewed as XSLT's ancestor—is based on a subset of Scheme, enriched by specialised libraries.

```
(defun factorial (x)
  "Behaves like the namesake function in Scheme
   (cf. Fig. 1)."
  (and
   (integerp x) (not (minusp x))
   (labels ((tr-fact (counter acc)
              ;; The labels special form of
              ;; Common Lisp introduces local
              ;; recursive functions [33, § 7.5].
              (if (zerop counter)
                  acc
                  (tr-fact (- counter 1)
                           (* acc counter)))))
     (tr-fact x 1))))
```

Figure 2. The factorial function in Common Lisp.

When we begin to program, the language we are learning is always shown as *finite product*. It has precise rules, precise semantics, and is *consistent*. According to the language used, some applications may be easy or difficult to implement. When you put down a statement, running it often results in something predictable. That hides an important point: a language results from some important choices: does it use lexical or dynamic scoping, or both? To illustrate this notion with some examples in TeX, that is the difference between the commands `\firstquestion` and `\secondquestion` in Fig. 4. The former can be related to *lexical* scoping, because it uses the value associated with the `\state` command at definition-time and produces:

You're happy, ain't U?

whereas the latter can be related to *dynamic* scoping, because it uses the value of the `\state` command at run-time and yields:

You're afraid, ain't U?

Students find this notion difficult: some know that they can redefine a variable by means of a `let` form in Emacs Lisp [22], but they do not realise that this would be impossible within lexically-scoped languages such as C or Scheme. In other words, they do not have *transversal* culture concerning programming languages,

```
fun factorial x =
  (* If x is a negative integer, the predefined
     exception Domain is raised [27, §§ 4.5–4.7]. The
     internal function tr_fact is defined by means of
     pattern matching [27, § 4.4].
  *)
  if x < 0 then raise Domain
  else let fun tr_fact 0 acc = acc |
               tr_fact counter acc =
                 tr_fact (counter - 1)
                         acc * counter
         in tr_fact x 1
       end ;
> val factorial = fn : int -> int
```

Figure 3. The factorial function in Standard ML.

```
\def\state{happy}
\edef\firstquestion{You're \state, ain't U?\par}
\def\secondquestion{You're \state, ain't U?\par}
\def\state{afraid}
```

Figure 4. Lexical and dynamic scope within TeX.

```
{\def\firsttwodigits{20}
 \def\lasttwodigits{09}
 \global\edef\thisyear{%
  \firsttwodigits\lasttwodigits}}
```

Figure 5. Using TeX's `\global` command.

they see each of them as an independent cell, a kind of *black box*.

The central part of our unit aims to emphasise these choices: what are the consequences of a lexical (resp. dynamic) scope? If the language is lexical (resp. dynamic), what kinds of applications are easier to be implemented? Likewise, what are the advantages and drawbacks of the call-by-value[18] strategy vs. call-by-name? In the language you are using, what is variables' extent?[19] Of course, all the answers depend on the programming languages considered. But our point of view is that a course based on Scheme and using other examples in TeX may be of interest.

## TeX's features shown

As mentioned above, `\def` and `\edef` allow us to illustrate the difference between lexical and dynamic scope. Most present-day programming languages are lexical, but we can observe that the dynamic scoping allows most TeX commands to be redefined by end-users. The dynamic scope is known to cause *variable captures*,[20] but TeX is protected against undesirable redefinitions by its internal commands, whose names contains the '`@`' character. Of course, forcing these internal commands' redefinition is allowed by the `\makeatletter` command, and restoring TeX's original behaviour is done by the `\makeatother` command.

If we are interested in implementation considerations, the commands within an `\edef`'s body are expanded, so this body is evaluated as far as possible.[21] To show this point, we can get dynamic scope with an `\edef` command by preventing command expansion by means of `\noexpand`:

```
\edef\thirdquestion{%
 You're \noexpand\state, ain't U?\par}
```

and this command `\thirdquestion` behaves exactly like `\secondquestion` (cf. Fig. 4).

A second construct, useful for a point of view related to conception, is `\global`, shown in Fig. 5, because it allows 'global' commands to be defined within local environments. There is an equivalent method in Scheme, but not naturally: see Appendix. Let us go on with this figure; any TeXnician knows that `\thisyear` no longer works if '`\edef`' is replaced by '`\def`'. This illustrates that TeX commands have limited extent.

Nuances related to notion of equality exist in TeX: let `\a` be a command already defined:

```
\let\b\a
\def\c{\a}
```

the former expresses that `\a` and `\b` are 'physically' equal, it allows us to retain `\a`'s definition, even it is changed afterwards; the latter expresses an equality at run-time, ensuring that the commands `\c` and `\a` are identical, even if `\a` changes.[22]

Scheme's standard does not allow end-users to know whether or not a variable `x` is bound.[23] A TeXnician would use:

```
\expandafter\ifx\csname x\endcsname\relax...%
 \else...%
\fi
```

For beginners in programming with TeX, this is quite a complicated statement requiring the commands `\relax` and `\ifx` to be introduced. However, that leads us to introduce not only the construct `\csname...\endcsname`, but also `\expandafter`, which may be viewed as kind of call by value. A simpler example of using this strategy is given by:

```
\uppercase\expandafter{\romannumeral 2009}
```

—which yields 'MMIX'—since this predefined command `\uppercase` is given its only argument as it is; so putting the `\expandafter` command causes this argument to be expanded, whereas removing it would produce 'mmix', because `\uppercase` would leave the group `{\romannumeral 2009}` untouched, then `\romannumeral` would just be applied to 2009. That is, TeX commands are *macros*[24] in the sense of 'more classical' programming languages.

The last feature we are concerned with is *mixfixed* terms, related to parsing problems and priorities. TeX can put mixfixed terms into action by means of *delimiters* in a command's argument, as in:

```
\def\put(#1,#2)#3{...}
```

## Discussion

As shown in the previous section, we use TeX as a 'cultural complement' for alternative constructs and implementations. Sometimes, we explain differences by historical considerations: for example, the difference between `\def` and `\long\def`—that is, the difference in LaTeX between `\textbf{. . . }` and `\begin{bfseries}...\end{bfseries}`—comes from performance considerations, since at the time TeX came out, computers were not as efficient as today. Nevertheless, are there other languages that could be successfully used as support of our unit? Yes and no.

An interesting example could be Common Lisp. Nevertheless, this language is less used now than some years ago, and it is complexified by the use of several *namespaces*.[25] Besides, this language's initial library is as big as possible; it uses old constructs.[26] That is why we give some examples in Common Lisp, but prefer for our course to be based on Scheme, which is 'the' modern Lisp dialect, from our point of view.

Concerning the coexistence of lexical and dynamic variables, the Perl[27] language [37] provides it. In addition, it has been successfully used to develop large software packages, so examples could be credible. However, it seems to us that dynamic variables in Perl are rarely used in practice. In fact, the two dynamic languages mainly used are Emacs Lisp and TeX, in the sense that end-users may perceive this point. From our point of view, using examples in Emacs Lisp requires good knowledge about the emacs[28] editor, whereas we can isolate, among TeX's features, the parts that suit us, omitting additional details about TeX's tasks. Likewise, such an approach would be more difficult with Perl.

## Conclusion

Our unit is viewed as theoretical, whereas other optional units are more practical, so only a few students attend ours. But in general, students who choose it do not regret it, and in fact enjoy it. They say that they have clear ideas about programming after attending it. Some students view our examples in TeX as a historical curiosity since this language is quite old and originates from the 1980s, but they are surprised by its expressive power. Some, that are interested in using TeX more intensively, can connect programming in TeX to concepts present in more modern languages.

## Acknowledgements

When I decided to use TeX to demonstrate 'alternative' implementations of some features related to programming, I was quite doubtful about the result, even if I knew that some were interested. But feedback was positive, some students were encouraged to go thoroughly into implementing new TeX commands for their reports and asked me for some questions about that. Thanks to them, they encouraged me to go on with this way in turn.

## Appendix: `\global` in Scheme

In this appendix, we show that a construct like `\global` in TeX may be needed. Then we will explain why it cannot be implemented in Scheme in a 'natural' way.

Let us consider that we are handling *dimensions*, that is, a number and a measurement unit—given as a symbol—like in TeX or DSSSL. A robust solution consists of using a list prefixed by a *marker*—e.g., `((*dimension*) 1609344 mm)`—such that all the lists representing dimensions—of type *dimension*—share the same head element, defined once. To put this marker only when the components—a number and a unit[29]—are well-formed, it is better for the access to this marker to be restricted to the functions interfacing this structure. So here is a proposal for an implementation of functions dealing with dimensions:

```
(let ((*marker* '(*dimension*)))
  (define (mk-dimension r unit)
    ;; Performs some checks and creates an
    ;; object of type dimension, whose
    ;; components are r and unit.
    ...)
  (define (dimension? x)
    ;; Returns #t if x is of type dimension, #f
    ;; otherwise.
    ...)
  (define (dimension->mm dimension-0)
(define mk-dimension)
(define dimension?)
(define dimension->mm)

(let ((*marker* '(*dimension*)) ; Only the cell's address is relevant.
      (allowed-unit-alist
       `((cm . ,(lambda (r) (* 10 r))) ; Each recognised unit is associated with a function giving the
         (mm . ,values))))             ; corresponding length in millimeters.
  (set! mk-dimension
        (let ((allowed-units (map car allowed-unit-alist)))
          (lambda (r unit)
            (and (real? r) (>= r 0) (memq unit allowed-units) (list *marker* r unit)))))
  (set! dimension? (lambda (x) (and (pair? x) (eq? (car x) *marker*))))
  (set! dimension->mm
        (lambda (dimension-0) ; dimension-0 is supposed to be of type dimension.
          ((cdr (assq (caddr dimension-0) allowed-unit-alist)) (cadr dimension-0)))))
```

Figure 6. Global definitions sharing a common lexical environment in Scheme.

```
    ;; Returns the value of dimension-0,
    ;; expressed in millimiters.
    ...))
```

Unfortunately, this does not work, because `define` special forms inside the scope of a `let` special form are viewed as *local* definitions, like `\def` inside a group in TeX. So, `mk-dimension`, `dimension?`, and `dimension->mm` become inaccessible as soon as this `let` form is processed. The solution is to define these three variables globally, and modify them inside a local environment, as shown in Fig. 6.

This *modus operandi* is quite artificial, because it uses side effects, whereas functional programming aims to avoid such as far as possible. But in reality, from a point of view related to conception, there is no 'actual' side effect, in the sense that variables like `mk-dimension`, `dimension?`, and `dimension->mm` would have been first given values, and then modified. The first bindings may be viewed as *preliminary declarations*;[30] however, using 'global' declarations for variables introduced within a local environment would be clearer, as in TeX. To sum up, such an example illustrates that some use of assignment forms are not related to actual side effects, and TeX's `\global` command allows us to explain how this example could appear using a 'more functional' form, without any side effect.[31]

## References

[1] Harold Abelson and Gerald Jay Sussman, with Julie Sussman: *Structure and Interpretation of Computer Programs.* The MIT Press, McGraw-Hill Book Company. 1985.

[2] Neil Bradley: *The Concise* SGML *Companion.* Addison-Wesley. 1997.

[3] Alonzo Church: *The Calculi of Lambda-Conversion.* Princeton University Press. 1941.

[4] Jean-Christophe Filliâtre and Claude Marché: *The BibTeX2*HTML *Home Page.* June 2006. http://www.lri.fr/~filliatr/bibtex2html/.

[5] Jonathan Fine: "TeX as a Callable Function". In: *EuroTeX 2002*, pp. 26–30. Bachotek, Poland. April 2002.

[6] Michael J. Gordon, Arthur J. Milner and Christopher P. Wadsworth: *Edinburgh* LCF. No. 78 in LNCS. Springer-Verlag. 1979.

[7] Paul Graham: ANSI Common Lisp. Series in Artificial Intelligence. Prentice Hall, Englewood Cliffs, New Jersey. 1996.

[8] Jean-Michel Hufflen : *Programmation fonctionnelle en Scheme. De la conception à la mise en œuvre.* Masson. Mars 1996.

[9] Jean-Michel Hufflen : *Programmation fonctionnelle avancée. Notes de cours et exercices.* Polycopié. Besançon. Juillet 1997.

[10] Jean-Michel Hufflen : *Introduction au λ-calcul (version révisée et étendue).* Polycopié. Besançon. Février 1998.

[11] Jean-Michel Hufflen: "A Tour around MlBibTeX and Its Implementation(s)". *Biuletyn* GUST, Vol. 20, pp. 21–28. In *BachoTeX 2004 conference.* April 2004.

[12] Jean-Michel Hufflen: "Managing Languages within MlBibTeX". TUG*boat*, Vol. 30, no. 1, pp. 49–57. July 2009.

[13] International Standard ISO/IEC 10179:1996(E): DSSSL. 1996.

[14] *Java Technology*. March 2008. `http://java.sun.com`.

[15] Richard Kelsey, William D. Clinger, Jonathan A. Rees, Harold Abelson, Norman I. Adams IV, David H. Bartley, Gary Brooks, R. Kent Dybvig, Daniel P. Friedman, Robert Halstead, Chris Hanson, Christopher T. Haynes, Eugene Edmund Kohlbecker, Jr, Donald Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy Lewis Steele, Jr, Gerald Jay Sussman and Mitchell Wand: "Revised$^5$ Report on the Algorithmic Language Scheme". HOSC, Vol. 11, no. 1, pp. 7–105. August 1998.

[16] Brian W. Kernighan and Dennis M. Ritchie: *The C Programming Language.* 2nd edition. Prentice Hall. 1988.

[17] Donald Ervin Knuth: *Computers & Typesetting. Vol. A: The TeXbook.* Addison-Wesley Publishing Company, Reading, Massachusetts. 1984.

[18] Matthias Köppe: *A BibTeX System in Common Lisp.* January 2003. `http://www.nongnu.org/cl-bibtex`.

[19] Thomas Lachand-Robert : *La maîtrise de TeX et LaTeX.* Masson. 1995.

[20] Leslie Lamport: *LaTeX: A Document Preparation System. User's Guide and Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[21] Xavier Leroy, Damien Doligez, Jacques Garrigue, Didier Rémy and Jérôme Vouillon: *The Objective Caml System. Release 0.9. Documentation and User's Manual.* 2004. `http://caml.inria.fr/pub/docs/manual-ocaml/index.html`.

[22] Bill Lewis, Dan LaLiberte, Richard M. Stallman and the GNU Manual Group: *GNU Emacs Lisp Reference Manual for Emacs Version 21. Revision 2.8.* January 2002. `http://www.gnu.org/software/emacs/elisp-manual/`.

[23] John McCarthy: "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I". *Communications of the ACM*, Vol. 3, no. 4, pp. 184–195. April 1960.

[24] Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LaTeX Companion.* 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[25] Chuck Musciano and Bill Kennedy: *HTML & XHTML: The Definitive Guide.* 5th edition. O'Reilly & Associates, Inc. August 2002.

[26] Oren Patashnik: *BibTeXing.* February 1988. Part of the BibTeX distribution.

[27] Lawrence C. Paulson: *ML for the Working Programmer.* 2nd edition. Cambridge University Press. 1996.

[28] Simon Peyton Jones, ed.: *Haskell 98 Language and Libraries. The Revised Report.* Cambridge University Press. April 2003.

[29] Erik T. Ray: *Learning XML.* O'Reilly & Associates, Inc. January 2001.

[30] Denis B. Roegel : « Anatomie d'une macro » . *Cahiers GUTenberg*, Vol. 31, p. 19–27. Décembre 1998.

[31] Michael Sperber, William Clinger, R. Kent Dybvig, Matthew Flatt, Anton van Straaten, Richard Kelsey, Jonathan Rees, Robert Bruce Findler and Jacob Matthews: *Revised$^6$ Report on the Algorithmic Language Scheme.* September 2007. `hhtp://www.r6rs.org`.

[32] George Springer and Daniel P. Friedman: *Scheme and the Art of Programming.* The MIT Press, McGraw-Hill Book Company. 1989.

[33] Guy Lewis Steele, Jr., with Scott E. Fahlman, Richard P. Gabriel, David A. Moon, Daniel L. Weinreb, Daniel Gureasko Bobrow, Linda G. DeMichiel, Sonya E. Keene, Gregor Kiczales, Crispin Perdue, Kent M. Pitman, Richard Waters and Jon L White: Common Lisp. *The Language. Second Edition.* Digital Press. 1990.

[34] "TeX Beauties and Oddities. A Permanent Call for TeX Pearls". In *TeX: at a turning point, or at the crossroads? BachoTeX 2009*, pp. 59–65. April 2009.

[35] Simon Thompson and Steve Hill: "Functional Programming through the Curriculum". In: *FPLE '95*, pp. 85–102. Nijmegen, The Netherlands. December 1995.

[36] W3C: *XSL Transformations (XSLT). Version 2.0.* W3C Recommendation. Edited by Michael H. Kay. January 2007. `http://www.w3.org/TR/2007/WD-xslt20-20070123`.

[37] Larry Wall, Tom Christiansen and Jon Orwant: *Programming Perl.* 3rd edition. O'Reilly & Associates, Inc. July 2000.

[38] Halina Wątróbska i Ryszard Kubiak: „Od XML-a do TeX-a, używając Emacsa i Haskella". *Biuletyn* GUST, tom 23, strony 35–39. In *BachoTeX 2006 conference.* kweicień 2006.

[39] Jiři Zlatuška: "NTS: Programming Languages and Paradigms". In: *EuroTeX 1999*, pp. 241–245. Heidelberg (Germany). September 1999.

## Notes

1. The most recent pearls can be found in [34].
2. **A**merican **N**ational **S**tandards **I**nstitute.
3. **H**yper**T**ext **M**arkup **L**anguage, the language of Web pages. [25] is a good introduction to it.
4. **C**ategorical **A**bstract **M**achine **L**anguage.
5. **M**ulti**L**ingual BibTEX.
6. This language was named after logician Haskell Brooks Curry (1900–1982).
7. **N**ew **T**ypesetting **S**ystem. It was finally developed using Java [14].
8. **C**OMMON **L**ISP **O**bject System.
9. '*Programmation fonctionnelle avancée*' in French. In 2009, it has been renamed to '*Outils pour le développement*' (*Tools for Development*), but changes in the contents are slight.
10. The program of this 2nd academic year unit can be found in French in [8].
11. 'Lisp' stands for '**LIS**t **P**rocessing, because Lisp dialects' major structure is linked lists. Their syntax is common and based on fully-parenthesised prefixed expressions. Lisp's first version, designed by John McCarthy, came out in 1958 [23]. This language has many descendants, the most used nowadays being COMMON LISP and Scheme.
12. 'Dynamically typed' means that we can know the type of an object at run-time. Examples are given in Figs. 1 & 2.
13. 'ML' stands for '**M**eta**L**anguage' and has been initially developed within the formal proof system LCF (**L**ogic for **C**omputable **F**unctions) [6]. Later on, it appears as an actual programming language, usable outside this system, and its standardisation resulted in the Standard ML language.
14. There are several definitions of *strong typing*. The most used within functional programming is that the variables are typed at compile-time. Some courses at the same level are based on a strongly typed functional programming language, examples being CAML or Haskell. Is that choice better than Scheme? This is a lively debate... but it is certain that these courses do not emphasise the same notions as a course based on a Lisp dialect.
15. **D**ocument **S**tyle **S**emantics **S**pecification **L**anguage.
16. **S**tandard **G**eneralised **M**arkup **L**anguage. Ancestor of XML (e**X**tensible **M**arkup **L**anguage); it is only of historical interest now. Readers interested in SGML (resp. XML) can refer to [2] (resp. [29]).
17. e**X**tensible **S**tylesheet **L**anguage **T**ransformations.
18. Nowadays, the call by value is the most commonly used strategy—in particular, in C and in Scheme—the argument expression(s) of a function are evaluated before applying the function. For example, the evaluation of the expression `(factorial (+ 1 9))`—see Fig. 1—begins with evaluating `(+ 9 1)` into `10`, and then `factorial` is applied to `10`. In other strategies, such as *call by name* or *call by need*, argument expressions are evaluated whilst the function is applied.
19. The *extent* of a variable may be viewed as its lifetime: if it is *limited*, the variable disappears as soon as the execution of the block establishing it terminates; if it is *unlimited*, the variable exists as long as the possibility of reference remains. In Scheme, variables have unlimited extent.
20. A *variable capture* occurs when a binding other than the expected one is used.
21. On the contrary, Scheme interpreters do not evaluate a `lambda` expression's body. They use a technique—so-called *lexical closure*—allowing the function to retrieve its definition environment.
22. There is another distinction in Scheme, between 'physical' (function `eq?`) and 'visual' equality (function `equal?`) [31, § 11.5].
23. COMMON LISP allows that about variables and functions, by means of the functions `boundp` and `fboundp` [33, 7.1.1].
24. Macros exist in Scheme: the best way to implement them is the use of *hygienic* macros, working by pattern-matching [31, §§ 11.2.2 & 11.19].
25. As an example of handling several namespaces in COMMON LISP, `let` is used for local variables, whereas local recursive functions are introduced by `labels`, as shown in Fig. 2.
26. For example, there is no hygienic macro in COMMON LISP.
27. **P**ractical **E**xtraction **R**eport **L**anguage.
28. **E**diting **MAC**ros.
29. ... although we consider only centimeters and millimeters in the example given in Fig. 6, for sake of simplicity.
30. In Scheme's last version, a variable can be defined without an associated value [31, § 11.2.1]. That was not the case in the version before [15, § 5.2], so such a variable declaration was given a *dummy* value, which enforced the use of side effects.
31. Many data structures, comparable with our type *dimension*, are used within MlBibTEX's implementation, as briefly sketched in [12]. Another technique, based on *message-passing*, allows us to avoid side effects. Only one function would be defined to manage dimensions, and the three functionalities implemented by `mk-dimension`, `dimension?`, and `dimension->mm` would be implemented by messages sent to the general function, the result being itself a function.

Jean-Michel Hufflen
LIFC (EA CNRS 4157),
University of Franche-Comté, 16, route de Gray,
25030 Besançon Cedex, France

# Introducing new French-speaking users to LaTeX quickly and convincingly

**Abstract**
For four university years, we had to introduce 2nd-year university students in Mathematics to LaTeX. An important goal was to make them able to use LaTeX when they are given some long homework in Mathematics the year after (3rd-year university). This teaching unit only included lab classes and was 15 hours long. We present our approach in detail and explain how it was perceived by students.

**Keywords**
Teaching LaTeX, successive steps of a course, lab-class-based curriculum, students' perception

## Introduction

When LaTeX [23] came out, it was sometimes viewed as a program hard to use, except for computer scientists familiar with hermetic programming languages. However this word processor has become more and more well-known as a powerful tool that produces high-quality print output. Besides, beginners can learn it now with many books introducing it, in many languages, some—non-limitative—examples are [13] in English, [4, 22, 30] in French, [27] in German, [5, 34] in Hungarian, [3] in Italian, [7] in Polish, [31] in modern Greek, ... In addition, some universities propose introductions to LaTeX within their *curricula.* An example is a unit—entitled *Scientific Tools*—we taught for four academic years (2004–2008), at the Faculty of Science and Technics, part of the University of Franche-Comté and located at Besançon, in the east of France.

Students who attended this unit were in the 2nd academic year of Mathematics.[1] One goal of this teaching unit was to ease the writing of an important homework the year after, that is, within the 3rd academic year in Mathematics, so a substantial part of this unit was devoted to LaTeX's math mode. Let us be precise that this teaching unit was not optional; that is, all the students had to attend it, even if they were not convinced of LaTeX *a priori.* Of course, some had heard about it, some had not. This unit only included lab classes and was 15 hours long. So students actually practised exercises in LaTeX, but we did not have enough time to show very advanced features.

We think that the approach we follow is interesting. In a first section, we make explicit our requirements and the pitfalls we wanted to avoid. Then we show the broad outlines of the steps of our unit and summarise the experience we got. Of course, reading this article only requires basic knowledge about LaTeX.

## What to do? What to avoid?

Many introductions to LaTeX begin with typing a small text and enriching it; some examples are [5, 22]. Our starting point is that this *modus operandi* has too many drawbacks, especially for non-English-speaking future users, in particular for French-speaking ones. First, only a few students are familiar with typing texts quickly and intensively, even if some have already used computers. They may make some typing mistakes in command names. Of course, any LaTeX teacher is able to fix them, but the price to pay is loss of time and dynamic. Besides, students need to be convinced of LaTeX from their first experiments. They should see that this word processor is suitable for large-sized texts, at the beginning, they should be able to observe that it is easy with LaTeX to apply some changes related to layout: changing characters' basic size, switching one-column and two-column layouts, ... All these goals can be reached only if students are given a text already typed and ready to be processed. That is, compiling this text should be successful the first time, so there is no anxiety about this point.

Besides, let us not forget that the most natural choice for a text to be typed by French students is a text in French. But some typographical rules are different from English ones: for example, a thin space—produced by the LaTeX command '\,'—is to be put just before a 'high' punctuation sign,[2] such as an exclamation or question mark:[3]

Joie, bonheur et délectation !

whereas the same signs are glued to the preceding text in English:

there was a lot of fun!

That is, such punctuation signs should be *active*[4] within French fragments. Of course, the simplest solution to this problem is to use the babel package's french option [25, Ch. 9]. So, end-users can type '`Vous␣comprenez?`' or '`Vous␣comprenez␣?`' and the results will be typeset correctly in both cases:

Vous comprenez ?

This point may seem to be a digression, but our purpose is to show how difficult the beginning of an introduction to LaTeX for non-English-speaking people is. Teachers are placed in a dilemma: either students have to typeset texts peppered with commands such as '`\'`' or '`\,`', or they should be given a big preamble, consisting of many `\usepackage` directives, with the advice 'You will understand later.'[5] In the case of the French language, this point is enforced because of accented letters: the most frequently used are directly provided by French keyboards—for example, 'é' or 'è', very frequent within French words—but the keys are unusable if the inputenc package has not been loaded with the latin1 option [25, § 7.11.3]. If French students begin to learn LaTeX by typing their own texts, there is no doubt that these texts will contain accented letters.

Anyway, the best solution seems to be a complete text—in French or in English—and students can perform first exercises by changing some sentences or adding some short fragments. Students can put down some simple sentences in English, so writing in this language avoids some problems related to French typography. When they have become familiar with the commands of LaTeX and its 'philosophy', the tools making it practical to write in the French language—the babel and inputenc packages—will be introduced. From our point of view, a 'good' text, usable as a starting point, should provide the following features:

□ a title, author, and date identified, so students can learn about commands such as `\title`, `\author`, `\date`, and `\maketitle`; an annotation may be a pretext for introducing the `\thanks` command;
□ an average-sized text;
□ a command used without argument, in order to show that a space character following a command's name without explicit delimiter is gobbled up;
□ a word hyphenated incorrectly, so students can realise that some words may be hyphenated, and learn how to fix such a mistake, even if that is rare;
□ a pretext for introducing a new command in LaTeX,
□ a pretext for introducing cross-references.

## The steps of our unit

The source text given to students is [10]. More precisely, the first version, giavitto.tex, does not use the babel package, even though this text is in French, with a short introduction we wrote in English. The inputenc package is not used, either, so we used TeX accent commands, and 'high' punctuation signs are explicitly preceded by a thin space, e.g.:

```
Joie, bonheur et d\'{e}lectation\,!
```

This text, a criticism about a book, came out in a forum. It has seemed to us to be very suitable for such an introduction to LaTeX, because:

□ it is 3 pages long, that is, a small-sized text, but not too short;
□ the introduction's second paragraph reads:

```
... using the \LaTeX\ word processor...
```

□ without the babel package's french option, there is a word hyphenated between a consonant and the following vowel, which is incorrect in French:[6] '`ex-emple`' (for 'example', hyphenated as '`ex-ample`' in English) should be hyphenated as '`exem-ple`';[7]
□ in this text, some words need an emphasis stronger than what is usually expressed by italicised characters: in the original text, typeset using only the standard typewriter font,[8] these words were written using capital letters:

```
Comment pouvait-IL savoir cela ?
```

The source text reads:

```
Comment pouvait-\superemph{il}...
```

and we can illustrate the use of variants of this new command `\superemph`:

```
\newcommand{\superemph}[1]{\uppercase{#1}}
\newcommand{\superemph}[1]{%
 **\uppercase{#1}**}
\newcommand{\superemph}[1]{**\textsc{#1}**}
...
```

That allows a kind of '*semantic markup*' [17], in the sense that this markup is related to a semantic notion, rather than some layout.

The first exercise is to compile the source text of this first version giavitto.tex, the second is to play with some options of the `\documentclass` command: twocolumn, 12pt, … so students can see that adapting a document to different layouts is easy. Guidelines are given in [15]. To sum up the order we follow:

- □ basic notions: commands, environments, preamble;
- □ sectioning commands: `\part`, `\chapter`, …
- □ parsing problem regarding commands without a right delimiter (*cf. supra*);
- □ formatting environments: `center`, `flushleft`, `flushright`;
- □ changing characters' look: commands and environments such as `\textbf` and `bfseries`, `\textsf` and `sffamily`, …
- □ introducing and redefining new commands: `\newcommand` and `\renewcommand`, use of 'semantic' markup, by means of commands such as `\superemph` (*cf. supra*), 'local' definitions—surrounded by additional braces—vs. global ones;
- □ changing size: commands and environments `small`, `footnotesize`, …
- □ list environments: `itemize`, `description`, `enumerate`; counters controlling `enumerate` environments, difference between redefining *values* and *look*—e.g., as done respectively by the commands `\enumi` and `\labelenumi`—insertion of footnotes;
- □ introducing *packages*: examples are indentfirst[9] [25, p. 32] and eurosym [25, pp. 408–409];
- □ notion of *dimensions*, how the page layout parameters are defined [25, Fig. 4.1] and how to customise them;
- □ how sequence of words (resp. successive lines) are split into lines (resp. pages), useful commands such as `\-`, `\linebreak`, `\pagebreak`, `\smallskip`, `\medskip`, `\bigskip`, putting unbreakable space characters by means of the '~' input character;
- □ management of *cross-references* and introduction of *auxiliary* (.aux) files, commands `\label`, `\ref`, `\pageref`; use of an additional .toc file for a table of contents and `\tableofcontents` command;
- □ introducing some basic differences between French and English typography; then we show how the babel package allows LaTeX to typeset texts written in many languages, possibly within the same document; introducing some useful commands of the babel package's french option; the 'standard' preamble of a LaTeX document written in French is given:

  ```
  \documentclass{...}

  \usepackage[...,french]{babel}
  \usepackage[T1]{fontenc}
  \usepackage[latin1]{inputenc}
  ...
  ```

  (see [25, §§ 7.11.3 & 7.11.4] about the packages fontenc and inputenc); as an example taking advantage of LaTeX's multilingual features as much as possible, a second version of [10], giavitto-plus.tex, is given to students;
- □ the document's end is devoted to some complements not demonstrated in lab classes: some converters to HTML[10] (LaTeX2HTML [11, Ch. 3], TeX4ht [11, Ch. 4], HyperLaTeX [19]), BibTeX [28].

Of course, students are not required to master all these items: we make precise the points students must know, and other information is given as a *memorandum*, e.g., the list of commands changing characters' look. A second document [16] is devoted to math mode and is organised as follows:

- □ math mode vs. text mode;
- □ spacing in math mode;
- □ commands changing characters' look in math mode, e.g., `\mathrm`, `\mathit`, …, additional packages such as amssymb or euscript;[11]
- □ commands producing Greek letters for mathematical purpose (`\alpha`, …) and symbols (`\leftarrow`, …) in math mode;
- □ subscripts, superscripts, fractions, radicals;
- □ adjustments: commands `\displaystyle`, `\textstyle`, …, operators *taking limits* or not, horizontal and vertical *struts*, the amsmath package's `\text` command;
- □ definition of operator names, by means of the commands `\mathop`, `\DeclareMathOperator`, `\DeclareMathOperator*`, `\mathbin`, `\mathrel`;
- □ *delimiter* management, by means of the commands `\left`, `\middle`, and `\right`;
- □ environments `cases` and `equation`; more features belonging to the amsmath package, such as the environments `multline`, `split`, `gather`, and the commands `\tag`, `\intertext`;
- □ environments belonging to LaTeX: `eqnarray`, `eqnarray*`;
- □ environments useful for general matrices (`[b|p|v|V]matrix`) and arrays (`[sub]array`), packages multirow, [del]array;
- □ back to LaTeX's text mode and introduction of the `tabular` environment.

Two other documents gently bring this unit to its end:

- □ [29] introduces pdfLaTeX and the hyperref package [11, Ch. 2], taking as much advantage as possible of the features of the PDF[12] format related to hyperlinks;

□ [1] is devoted to image insertion, by means of the packages graphic(s|x) [25, § 10.2].

Of course, all these documents include references—possibly on-line—that allow readers to learn more.

## Lessons learned

Teaching this unit gave good results: it actually seemed to us that students really enjoy discovering LaTeX and using it. Generally they got nice outputs. In addition, practising LaTeX's math mode caused them to realise how diverse 'graphical' expressions of Mathematics are. For example, 'modulo' is both an infixed and prefixed operator, as reflected by the two commands `\bmod` and `\pmod`. Likewise, the notion of operators taking limits separates the layout—the location of each piece of information—and the common notion—an interval's endpoints. That is, the commands of LaTeX's math mode may be viewed as *presentation markup*, comparable to the namesake notion in MathML[13] [33, § 2.1.2].

Anyway, let us recall that we taught students in Mathematics. Such students learned the basics of a programming language,[14] but do not plan to become computer scientists. So, they did not get used to presenting programs nicely, by indenting them, as students in Computer Science learn to do, in order to be able to work on them again. A good exercise to emphasise this point is first to give students a complicated formula to be typeset, then to ask them to change it.

Teachers have to give some advice about organising LaTeX source texts. For example, there should be no other word on a line containing `\begin` or `\end` commands delimiting environments, and nesting environments should be made clear by indenting them:

```
. . .  text before.
\begin{itemize}
␣\item . . .
␣\begin{itemize}
␣␣\item . . .
  . . .
 \end{itemize}
\end{itemize}
Text after. . .
```

Some notations should be avoided when a more systematic markup is available. For example, we think that it is better for students to get used to writing:

```
\begin{small}
...
\end{small}
```

than '`{\small ...}`'. Of course, the latter may appear as simpler for short fragments, but any TeXnician knows that it is possible to use a command like `\small` without additional braces, in which case, this size change runs until the next size change command. If the markup associated with a command is not clearly expressed, some students may be baffled. Besides, let us consider three versions of an end-user defined command typesetting a note using small-sized characters:

```
\newcommand{\note}[1]{%
 \begin{small}#1\end{small}}
\newcommand{\noteone}[1]{\small#1} % Wrong!
\newcommand{\notetwo}[1]{{\small#1}}
```

Of course, any LaTeX teacher can explain why the `\noteone` command does not work as expected, and how to fix this wrong definition as done for the `\notetwo` command. However, a user who is used to `small` as an environment—rather than '`{\small ...}`'—would probably put down this `\note` command as we did, and that is indisputably the simplest solution.

The commands and environments introduced by the LaTeX format have homogeneous taxonomy about delimiting arguments and effects. That is, the markup is very clear, in particular for beginners. That may not be the case for commands originating from TeX's basis: for example, if you would like to put a vertical strut whose length is given, we can use the construct `\vbox to 1.1\baselineskip{}` [16, § 2.7], that is, using a kind of *mixfixed* markup. Of course, dealing with such a command is rare. But other commands belonging to *plain TeX*'s math mode, such as `\over` or `\atop`, are error-prone since they have no argument and are only related to positional relationships. Let us compare plain TeX's `\over` command with LaTeX's `\frac`, that has 2 arguments: the source texts for the numerator and denominator.[15]

Last but not least, we notice some points related to the implementation we used: TeXnicCenter [32], built on top of the MiKTeX typesetting engine [24], and running under the Windows operating system. This graphic interface is very good, especially the correspondence between the editor's cursor and a marker in the resulting .dvi[16] file. The main drawback is that MiKTeX runs in non-stop mode. As a consequence, students may get almost complete texts in case of recoverable errors. So they do not have to be aware of their errors and they perceive only 'serious' ones. It is needed to introduce them to .log files, and ask them to tolerate only warning messages.

## Conclusion

LaTeX being extensible because of its numerous packages, it is impossible for an introductory course to give all the functionalities that already exist. In fact, teachers also have to show how to use LaTeX's documentation—good documents exist in French—to learn more on their own. But it is essential for students to understand LaTeX's philosophy and get good methods. We think our method fulfills these goals. From 2003 to 2005, J.-M. Hufflen taught 4th-year university students enrolled in 'Digital Publishing' program[17] at the Letter Faculty of Besançon, and got initial experiences for writing [15]. A more concise document [2] has been used by A.-M. Aebischer for analogous introductory courses given at the IREM[18] institute for future teachers in Mathematics.

## Acknowledgements

Sincere thanks to Karl Berry and Barbara Beeton, who proofread the first version of this article.

## References

[1] Anne-Marie Aebischer : *Insertion d'images.* Document de support de travaux pratiques. Avril 2008.

[2] Anne-Marie Aebischer : *Créer des documents scientifiques avec LaTeX.* Stage 2008 à l'IREM de Franche-Comté. 2008.

[3] Claudio Beccari: *Introduzione all'arte della composizione tipographica con LaTeX.* September 2009. GUIT.

[4] Denis Bitouzé et Jean-Côme Charpentier : *LaTeX.* Pearson Education. 2006.

[5] Bujdosó Gyöngyi – Fazekas Attila: *TeX kerdőlépések.* Tertia Kiadó, Budapest. április 1997.

[6] *The Chicago Manual of Style.* The University of Chicago Press. The 14th edition of a manual of style revised and expanded. 1993.

[7] Antoni Diller: *LaTeX wiersz po wierszu.* Wydawnictwo Helio, Gliwice. Polish translation of *LaTeX Line by Line* with an additional annex by Jan Jelowicki. 2001.

[8] Bernard Gaulle : *Notice d'utilisation de l'extension frenchpro pour LaTeX. Version V5,995.* Avril 2005. `http://www.frenchpro6.com/frenchpro/french/ALIRE.pdf`.

[9] Bernard Gaulle : *L'extension frenchle pour LaTeX. Notice d'utilisation. Version V5,9993.* Février 2007. `http://www.tug.org/texlive/Contents/live/texmf-dist/doc/latex/frenchle/frenchle.pdf`.

[10] Jean-Louis Giavitto : *Ouverture des veines et autres distractions.* Documents de support de travaux pratiques. `http://lifc.univ-fcomte.fr/home/~jmhufflen/l2s/giavitto.pdf` et `http://lifc.univ-fcomte.fr/home/~jmhufflen/l2s/giavitto-plus.pdf`. Octobre 1986.

[11] Michel Goossens and Sebastian Rahtz, with Eitan M. Gurari, Ross Moore and Robert S. Sutor: *The LaTeX Web Companion.* Addison-Wesley Longman, Inc., Reading, Massachusetts. May 1999.

[12] Maurice Grévisse : *Le bon usage.* Duculot. Grammaire française. $12^{e}$ édition refondue par André Goosse. 1988.

[13] David Griffiths and Desmond Higham: *Learning LaTeX.* SIAM. 1997.

[14] Jean-Michel Hufflen : « Typographie : les conventions, la tradition, les goûts, … et LaTeX ». *Cahiers GUTenberg*, Vol. 35–36, p. 169–214. In *Actes du congrès GUTenberg 2000*, Toulouse. Mai 2000.

[15] Jean-Michel Hufflen : *Premier contact avec LaTeX.* Document de support de travaux pratiques. `http://lifc.univ-fcomte.fr/home/~jmhufflen/l2s/to-do.pdf`. Janvier 2006.

[16] Jean-Michel Hufflen : *Mode mathématique.* Document de support de travaux pratiques. `http://lifc.univ-fcomte.fr/home/~jmhufflen/l2s/to-do-math.pdf`. Février 2006.

[17] Jean-Michel Hufflen: "Writing Structured and Semantics-Oriented Documents: TeX vs. XML". *Biuletyn* GUST, Vol. 23, pp. 104–108. In *BachoTeX 2006 conference.* April 2006.

[18] Jean-Michel Hufflen : *C++… et d'autres outils… pour l'étudiant mathématicien.* Polycopié. Besançon. Janvier 2008.

[19] *HyperLaTeX.* February 2004. `http://hyperlatex.sourceforge.net`.

[20] *Java Technology.* March 2008. `http://java.sun.com`.

[21] Donald Ervin Knuth: *Computers & Typesetting. Vol. A: The TeXbook.* Addison-Wesley Publishing Company, Reading, Massachusetts. 1984.

[22] Thomas Lachand-Robert : *La maîtrise de TeX et LaTeX.* Masson. 1995.

[23] Leslie Lamport: *LaTeX: A Document Preparation System. User's Guide and Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[24] *MiKTeX… Typesetting Beautiful Documents.* 2009. `http://miktex.org/`.

[25] Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The*

*LaTeX Companion.* 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[ 26 ] Chuck Musciano and Bill Kennedy: html*: The Definitive Guide.* 6th edition. O'Reilly & Associates, Inc. October 2006.

[ 27 ] Elke Niedermair und Michael Niedermair: *LaTeX—Das Praxisbuch.* 3. Auflage. Franzis. 2006.

[ 28 ] Oren Patashnik: *BibTeXing.* February 1988. Part of the BibTeX distribution.

[ 29 ] François Pétiard : *Le package* hyperref. Document de travaux pratiques. Mai 2005.

[ 30 ] Christian Rolland : *LaTeX par la pratique.* O'Reilly France, Paris. 1999.

[ 31 ] Apostolos Syropoulos: *LaTeX.* Ενας Πληρης για την Εϰμαϑηση του Συστηματος Στοιχειοϑεσιας *LaTeX.* Παρατηρητης. 1998.

[ 32 ] *TeXnicCenter.* 2008. http://www.texniccenter.org/.

[ 33 ] W3C: *Mathematical Markup Language (MathML) Version 2.0*, 2nd edition. w3c Recommendation. Edited by David Carlisle, Patrick Ion, Robert Miner, and Nico Poppelier. October 2003. http://www.w3.org/TR/2003/REC-MathML2-20031021.

[ 34 ] Wettl Ferenc – Mayer Gyula – Szabó Péter: *LaTeX kézikönyv.* Panem. 2004.

## Notes

1. '*License 2, parcours Mathématiques et Mathématiques appliquées*', w.r.t. French terminology.
2. This notion of 'high' sign of punctuation belongs to French typography's terminology. A short survey of these rules is given in [6, §§ 9.21–9.33], a more complete reference is [14], in French.
3. The following quotations come from [10].
4. This notion is explained in [21, Ch. 7].
5. So do [4, 7, 27, 31] The first example of [34] does not use any package, the \usepackage command being introduced immediately after. In addition, examples given at first are small-sized, so introducing some variants—e.g., twocolumn vs. onecolumn—would not be very convincing. On another subject, French texts can be typeset using the packages french(pro|le) [8, 9], as alternatives to the babel package, but the problem of introducing such a package at the course's beginning remains the same.
6. Except for etymological hyphenation, now hardly used in practice.
7. In fact, this point is debatable, because some French typography manuals consider that a word should not be hyphenated before a silent syllable—'*exemple*' sounds as [ɛgzɑ̃pl]. (That is why this word is not hyphenated in the version processed with the babel package's french option.) But these typography manuals mention that this convention is difficult to follow, in particular when text columns are narrow, as in daily newspapers, for example. More details about this point can be found in [12].
8. Let us recall that this text came out in October 1986; the interfaces used within these forums were not comparable to the Web.
9. Indenting the first paragraph after a display heading is more customary in French text than in English, so introducing this indentfirst package is relevant in our unit.
10. **H**yper**T**ext **M**arkup **L**anguage, the language of Web pages. [26] is a good introduction to it.
11. Most of the math mode's advanced features are described in detail in [25, Ch. 8].
12. **P**ortable **D**ocument **F**ormat, Adobe's format.
13. **MATH**ematical **M**arkup **L**anguage [33] is an xml (e**X**tensible **M**arkup **L**anguage) application for describing mathematical notation regarding either its structure or its content. Let us mention that MathML's broad outlines are taught to 5th-year students in Statistical Modelling ('*Master 2 de Mathématiques, mention Modélisation statistique*', in French) [18, ch. 9], as part of a unit entitled 'Software Engineering'.
14. Java [20], in the *curricula* of the Faculty of Science and Technics located at Besançon.
15. As mentioned in Note 13, there is an introduction to MathML for some 5th-year university students in Mathematics. MathML's content model [33, § 2.1.3], more related to the semantics of mathematical expressions, is easier to understand for these students than the presentation model.
16. **D**e**V**ice-**I**ndependent.
17. '*Master 1 d'Édition numérique*', in French.
18. '***I****nstitut de* ***R****echerche sur l'****E****nseignement des* ***M****athématiques*', that is, 'Research Institute about teaching Mathematics'.

Anne-Marie Aebischer[1]
Bruno Aebischer[1]
Jean-Michel Hufflen[2]
François Pétiard[1]
[1] Department of Mathematics (UMR CNRS 6623),
University of Franche-Comté, 16, route de Gray,
25030 Besançon Cedex, France.
[2] LIFC (EA CNRS 4157),
University of Franche-Comté, 16, route de Gray,
25030 Besançon Cedex, France.

# Oriental TEX by a dummy

**Abstract**
This article is converted from the slides presented at the conference.

## What is Oriental TEX

□ It is a project by Idris Samawi Hamid, Taco Hoekwater and Hans Hagen.
□ The project started shortly after we started the LuaTEX project.
□ It boosted development of LuaTEX thanks to a grant that paid for coding LuaTEX.
□ It also boosted the development of ConTEXt MkIV and was a real good torture test for OpenType font support.
□ This project also costs us a whole lot of time.
□ The main objective is to let TEX typeset high quality (traditional) Arabic.
□ Closely related to this is to extend ConTEXt capabilities to deal with advanced critical editions.
□ In the meantime a high quality Arabic OpenType font has become part of the package.

## How we proceed

□ Of course we were a bit too optimistic when setting the time schedule for this project.
□ This is because we need to have quite some bits and pieces in place beforehand.
□ For instance, making the font and perfecting OpenType support involves a lot of trial and error and testing.
□ This is mostly due to lack of specifications, benchmarks and limitations in tools.
□ We have identified the needs for critital editions but have postponed some of that till we have opened up more of LuaTEX.
□ We are also getting a better picture of what is needed for advanced right-to-left typesetting, especially in mixed directionality.

## Simple OpenType fonts

□ In Latin scripts we have mostly one-to-one and many-to-one substitutions.
□ This can happen in sequence (multiple passes).
□ Sometimes surrounding characters (or shapes) play a role.
□ In some cases glyphs have to be (re)positioned relative to each other.
□ Often the substitution logic is flawed and it is assumed that features are applied selectively (DTP: select and apply).
□ Of course this is unacceptable for what we have in mind.

## The Oriental TEX approach

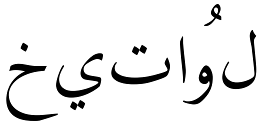
□ We put as much logic in the font as possible, but also provide a dedicated paragraph builder (written in Lua).
□ The so-called First-Order Analysis puts a given character into isolated, initial, middle, or final state.
□ The Second-order Analysis looks at the characters and relates this state to what characters precede or succeed it.

- □ Based on that state we do character substitutions. There can be multiple analysis and replacements in sequence.
- □ We can do some simple aesthetic stretching and additional related replacements.
- □ We need to attach identity marks and vowels in proper but nice looking places.
- □ In most cases we're then done. Contrary to other fonts we don't use many ligatures but compose characters.

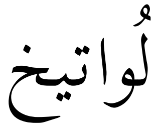
## But we go further

- □ The previous steps already give reasonable results and implementing it also nicely went along with the development of LuaTEX and ConTEXt MkIV.
- □ Currently we're working on extending and perfecting the font to support what we call Third-Order Contextual Analysis.
- □ This boils down to an interplay between the paragraph builder and additional font features.
- □ In order to get pleasing spacing we apply further substitutions, this time with wider or narrower shapes.
- □ When this is done we need to reattach identity marks and vowels.
- □ Optionally we can apply HZ-like stretching as a finishing touch.

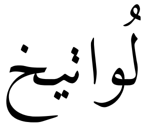
## Look at luatex (kheetawul)



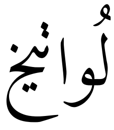
- □ no order (kh ī t ā w [u] l)



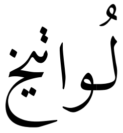
- □ first order



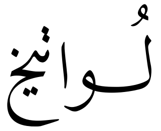
- □ second order



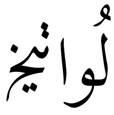
- □ second order (Jeem-stacking)



- □ minimal stretching



- □ maximal stretching (level 3)



- □ chopped letter khaa (for e.g. underlining)

Hans Hagen
Pragma ADE, Hasselt

# Writing Pitman shorthand with Metafont and LaTeX

**Abstract**
With pen shorthand, the traditional speech-recording method, unwritten speech is at first manually captured and then transliterated into a digital text. We have built programs which reverse the second step of this process, i.e. transform text into shorthand.
Here we present as a special case an online system, which converts English text into Pitman 2000 shorthand using Metafont and LaTeX. The impact of our system on pattern recognition of handwritten shorthand and on stenography teaching is discussed.

In order to approximate the speed of speech, alphabet based shorthand systems make use of phonetic writing, abbreviations and simplified writing, thus reducing the redundancy of the orthographic code and the graphic redundancy of longhand characters.

In the following sections we exemplify these principles with the *Pitman shorthand language* (abbreviated as PSL) and describe how the Pitman 2000 shorthand system can be implemented in Metafont [4].

## Elements of PSL

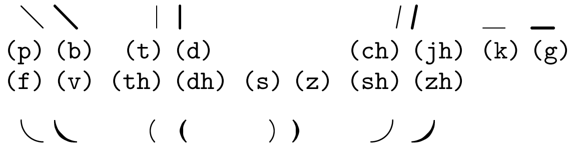
A glyph of one or more words as denoted with PSL, the so-called **stenem** is composed of

**an outline** consisting of joined consonant signs, written without lifting the pen from the paper, and
**diacritics** corresponding to vowel phonemes.

The stenem components are written in this order.

**An example**: The stenem of the word 'rote' ∕|, pronounced as `r * ou t` is built of the outline ∕|, formed by joining the strokes `(r)`=∕ and `(t)`=|, the signs[1] of the consonant phonemes `r` and `t` and the heavy dash sign `[ou]`, the diacritical mark of the vowel ou, following ∕.

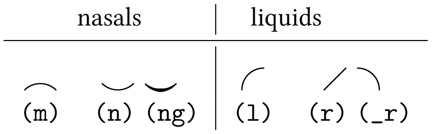
*The signs of consonant phonemes.* These signs, also called **strokes**, are either line segments or quarter circles:


(p) (b) (t) (d) (ch) (jh) (k) (g)
(f) (v) (th) (dh) (s) (z) (sh) (zh)

Though invented in 1837, the PSL design is guided by modern phonological classifications and principles [7, 8].

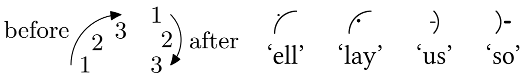
Thus the signs of voiced consonants are more firmly written variants of their unvoiced counterparts. Friction vs. occlusion of a consonant is denoted by rounding the corresponding sign (cf. the rows). A change of the place of articulation causes a change of slant in consonant signs (cf. the columns).

Remaining strokes[2] are:

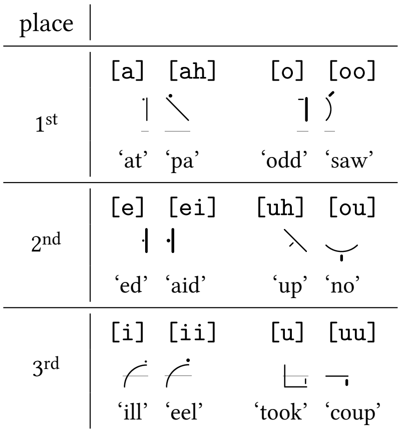


The signs ⌒ ◡ ◡ — — are horizontals, ( ∕ are upstrokes;[3] all other consonant signs are downstrokes.

*Vowel, diphthong and triphone signs.* These diacritical signs are placed alongside a consonant sign, before or after it, depending on whether the vowel is read before or after the consonant, i.e. going from the beginning of a stroke on the left-hand or the right-hand side of upstrokes and horizontals if the vowel is read before or after the consonant. Places are changed for downstrokes.



Twelve **vowel diacritics** are realized in PSL. They are differentiated by their glyph (light or heavy, dot or dash) and its position. Any consonant sign has three places for a vowel sign to be located according to the direction in which the consonant stroke is written: at the beginning (1st), in the middle (2nd) or at the end (3rd place).

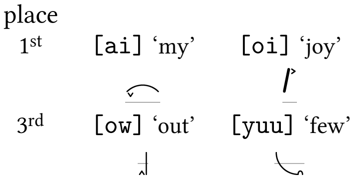


| place | | | | |
|---|---|---|---|---|
| 1st | [a]<br>'at' | [ah]<br>'pa' | [o]<br>'odd' | [oo]<br>'saw' |
| 2nd | [e]<br>'ed' | [ei]<br>'aid' | [uh]<br>'up' | [ou]<br>'no' |
| 3rd | [i]<br>'ill' | [ii]<br>'eel' | [u]<br>'took' | [uu]<br>'coup' |

It can be seen from this table that the light vowel signs are reserved for the short vowels and are put in the same places as the heavy vowel signs for the long vowels.

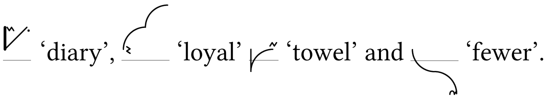
The table proceeds row-wise (over the position) from signs for opened vowels to signs for closed vowels and column-wise from dots for front vowels to dashes[4]) for back vowels. Compare a such as in 'at', which is an opened front vowel with the closed back vowel uu, such as the one in 'coup' at the opposite vertices of the table.[5]

There are four **diphthong signs** at 1st and 3rd places:



| place | | |
|---|---|---|
| 1st | [ai] 'my' | [oi] 'joy' |
| 3rd | [ow] 'out' | [yuu] 'few' |

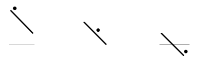
The **triphone signs**, indicated by a small tick attached to a diphthong sign, represent any vowel following the diphthong, as in:



'diary', 'loyal' 'towel' and 'fewer'.

There is also a special diphone sign for other vowel combinations put in the place of the first vowel. Consider the second mark for the diphone ia in 'idea' put at the 3rd place – the place of [i].

Observe also, that the first vowel in a word decides where the first upstroke or downstroke of the outline will be written – above, on or through the line.

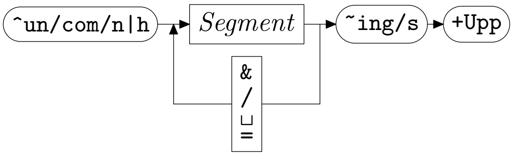


'pa' 'pay' 'pea'

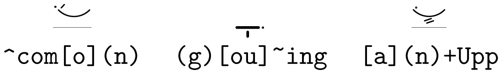
## Stenems

We propose here a complete PSL grammar and describe it by means of syntax diagrams.[6] A terminal resulting from grammar productions is called **metaform**, e.g. (r)[ou]&(t) is the metaform[7] corresponding to , the stenem of the word 'rote'.

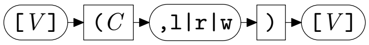
Stenems are composed of segments with (mostly) joined (&-Notation) outline.

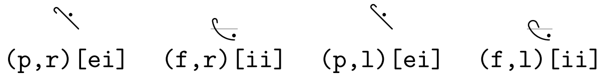


Stenems can start with a morphological prefix such as ^com, ^con, ... and they can end with a verbal suffix, such as ~ing. Both are realized by a mark, such as a light dot, before the first and/or after the last segment outline, respectively. Last suffix +Upp indicates proper names, whose glyphs are underlined in PSL.

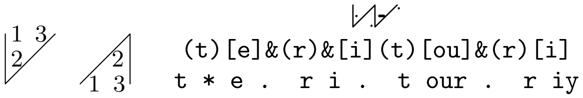


*Segments.* The **core of a segment** is built of an obligatory consonant sign (*C*) framed by optional vowel signs [*V*]. The strokes of Section 1 can be modified to express the frequent case of a consonant followed by an r or an l – written by an initial right or left hook,[8] respectively:

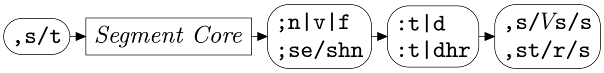




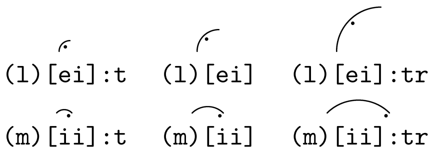
The segments are the counterparts of the syllables, hence there is a provision for vowel signs occurring between two strokes – 1st and 2nd place vowel signs are written after the first stroke whereas the otherwise ambiguous 3rd place vowel signs are written before the second stroke:



The following syntax diagram completes the definition of a PSL segment



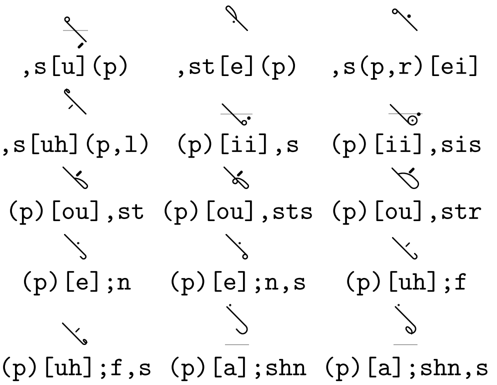
At first PSL strokes come at three sizes – half-sized (suffix :t/d), normal or double-sized (suffix :tr/dr/dhr), e.g.:



(l)[ei]:t (l)[ei] (l)[ei]:tr

(m)[ii]:t (m)[ii] (m)[ii]:tr

Additionally strokes can be prefixed and/or suffixed by left or right, small or larger hooks, circles, loops and cracknels,[9] for example:



,s[u](p) ,st[e](p) ,s(p,r)[ei]

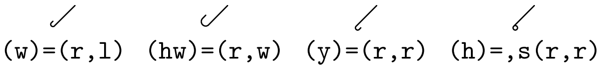
,s[uh](p,l) (p)[ii],s (p)[ii],sis

(p)[ou],st (p)[ou],sts (p)[ou],str

(p)[e];n (p)[e];n,s (p)[uh];f

(p)[uh];f,s (p)[a];shn (p)[a];shn,s

Observe also how the (not previously mentioned) signs of the consonants w, hw, y and y are defined:

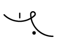
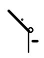


(w)=(r,l) (hw)=(r,w) (y)=(r,r) (h)=,s(r,r)

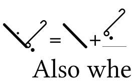
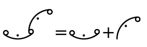
*Joining and disjoining the segments.* Stenem outlines are written mostly without lifting the pen; typically the segments are joined at obtuse outer angles, e.g.:

'get' or 'tick'.

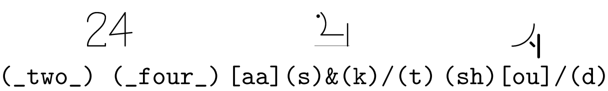
Special care must be taken for ,s-circles. As an ,s-circle is normally written within the curves or as a left circle otherwise, and as writing the circle outside an angle is the simplest way of joining two segments – the circle direction must be sometimes reversed:

| left ,s | | right ,s | |
|---|---|---|---|
| |  | |  |
| 'cassette' | 'unsafe' | 'task' | 'bestow' |

However there are singular cases, where a continuous connection is not possible, consider

 ('be+half')  ('sense+less')

Also when writing numbers or when writing the endings t or d in past tense of regular verbs, the segments are disjoined:[10]

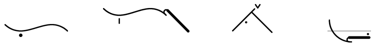


(_two_) (_four_) [aa](s)&(k)/(t) (sh)[ou]/(d)

*Metafont implementation.* Elementary strokes (circle arcs or straight lines) and the circles, hooks, … used for prefixes/suffixes are realized as splines with curvature=0 at both ends. Thus trivially consonantal parts of segments, when joined tangent continuously are curvature continuous, too [5].

Technically speaking the diacritics are an array of discontinuous marker paths, while the outline is an array of (mostly) continuous only[11] Metafont-paths.[12]

Besides the circled connections also the m/n joinings were made curvature continuous; joinings with cusps exist, too:

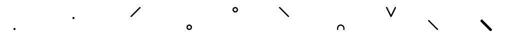


'name' 'number' 'reply' 'figure'

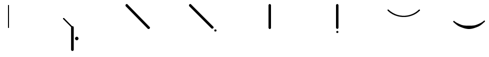
## Abbreviations

In PSL, short forms, phrases and intersections are used for frequently occurring words.

**Short forms** are either arbitrary strokes or shortened outlines largely without diacritics:

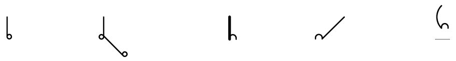


'the' 'a/n' 'and' 'is' 'as' 'of' 'you' 'I' 'to' 'too'



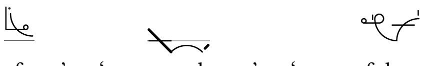
'it' 'today' 'be' 'being' 'do' 'doing' 'in' 'thing'

**Phrases** in PSL are simply stenems of two and more words connected together:[13]



'it is' 'it is possible' 'do you' 'you are' 'thank you'

Also one stroke may be struck through a preceding one in commonly used collocations. Examples of such **intersections** are:[14]



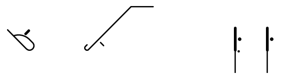
'tax form' 'company boom' 'successful company'

Short forms, phrases and intersections are to be learned by heart. Our system maintains an abbreviation dictionary of (word, metaform) tuples written in lexc [1].

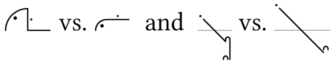
## text2Pitman

text2Pitman is an online system,[15] which records input text as Pitman 2000 shorthand. Just as in [5, 6] the conversion is done in four steps:

1. The input is tokenized. Tokens with a metaform entry in the abbreviation dictionary are separated from other words.
2. For a word in the latter category we find its pronunciation in Unisyn accent-independent keyword lexicon [2]. The non/writing of minor vowels, the so-called schwas (@),[16] is guided by special PSL rules: in secondary stress syllables most of them are ignored ('poster'), rhotic schwas are written out ('work') and some others are to be back-transformed[17] ('data' vs. 'date'):


'poster' 'work' 'data' vs. 'date'

The pronunciation string is then transformed to a metaform by the stenemizer – a program coded as the tokenizer above in the XEROX-FST tool `xfst` [1]. The transformation is carried out by a series of cascaded context dependent rewrite rules, realized by finite state transducers (FSTs). Decomposition of a stenem into its constituent segments as done by the stenemizer is unique, but as the underlying PSL grammar allows ambiguities,[18] the metaform found is not always the one commonly used:[19]

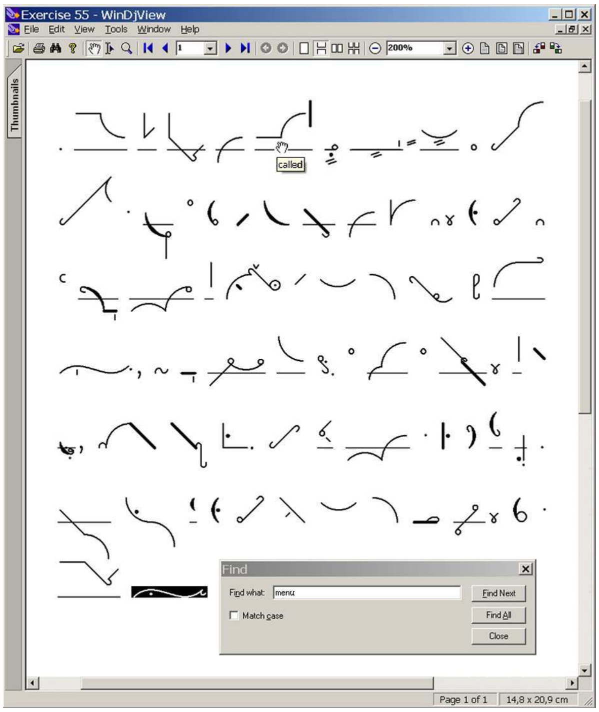


3. In a `mf` run for each of the tokens using the code resulting from its metaform a Metafont character is generated.
4. The text is then set with LaTeX, rendered with `dvips`, … and sent as an image to the browser.

## Remarks on pattern recognition

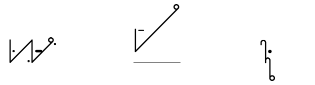
text2Pitman can provide test samples for the reversed procedure – the pattern recognition of handwritten PSL. This task is done in three major steps:[20]

1. **Shape recognition yielding the metaform.**
   This step requires at first the recognition of the mid points of segments and of the slope as well of the curvature sign there. Then the prefixes and suffixes have to be found and classified.
2. **Conversion of the metaform into pronunciation strings.**
   As our stenemizer is a two-level[21] `xfst` transducer, this could be accomplished by reversing its order of operation, but it is more elaborate. Shorthand writers often omit vowel diacritics in some words, such as:

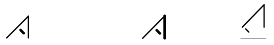


Find what: menu

Figure 1.


'territories' 'tortures' 'traitors'

It is not harmful in long outlines,[22] but for short stenems the correct use of diacritics and observing the right overall position is essential. Consider an example of words with nearly the same PSL outline:


'rote/wrote' 'rode/road' 'rot'

As most recognizers do not detect line thickness, still more shorthand homographs result. Thus, this complex task can be handled only by taking into account the word frequencies and using a weighted transducer. Nevertheless our system could automatically create a knowledge base of (metaform, pronunciation string(s)) entries.

3. The transliteration of the pronunciation strings into English with correct orthography is difficult because of the numerous and very frequent English language homophones.[23]

## Educational uses of the software

A novel dvitype-based DjVu-backend of our software to produce a text-annotated and searchable shorthand record, which can be viewed with a standard DjVu-plugin to a browser or a standalone viewer. Moving with the mouse over a stenem displays the originating word(s), as can be seen in figure 1.

Compare our “live”‘ record with a printed textbook, where the writing or reading “Exercises” are separated from the “Keys to Exercises”.

It is probable that as shorthand usage declines, publishers of shorthand books will not, as in the past, insist on their proprietary solutions. In any case, our web server based software suggests a future with centralized dictionaries and textbooks utilized and maintained by an interested user community.

## References

[1] Kenneth R. Beesley and Lauri Karttunen. *Finite State Morphology.* CSLI Publications, Stanford, 2003.

[2] Susan Fitt. Unisyn multi-accent lexicon, 2006. `http://www.cstr.ed.ac.uk/projects/unisyn/`.

[3] Swe Myo Htwe, Colin Higgins, Graham Leedham, and Ma Yang. Knowledge based transcription of handwritten Pitman's Shorthand using word frequency and context. In *7th International Conference on Development and Application Systems*, pages 508–512, Suceava, Romania, 2004.

[4] Donald E. Knuth. *The Metafontbook*, volume C of *Computers and Typesetting.* Addison-Wesley Publishing Company, Reading, Mass., 5th edition, 1990.

[5] Stanislav J. Šarman. Writing Gregg Shorthand with Metafont and LATEX. *TUGboat*, 29(3):458–461, 2008. TUG 2008 Conference Proceedings.

[6] Stanislav J. Šarman. DEK-Verkehrsschrift mit Metafont und LATEX. *Die TEXnische Komödie*, 21(1):7–20, 2009.

[7] Bohumil Trnka. *A Phonological Analysis of Present-day Standard English.* Prague, 1935. Revised Edition, 1966.

[8] Bohumil Trnka. *Pokus o vědeckou teorii a praktickou reformu těsnopisu. An Attempt at the Theory of Shorthand and its Practical Application to the Czech Language.* Facultas Philosophica Universitatis Carolinæ Pragensis, Sbírka pojednání a rozprav XX, Prague, 1937.

## Notes

[1] In the following, phonemes are denoted in typewriter type, the corresponding consonant signs are parenthesized, and vowel, diphthong and triphone signs are bracketed.

[2] There are two signs for `r`. For the signs of `h`, `w`, `wh` and `y` see Section 2.1.

[3] `(l)` can be written in both directions.

[4] Dashes are written at right angles to the stroke at the point where they are placed.

[5] which is nearly Jones' IPA vowel quadrilateral reflected.

[6] Optional vs. obligatory parts are enclosed in rounded boxes; nonterminals are written in cursive, terminals in typewriter type.

[7] The metaform without intervening non-letters corresponds linearly (stress and schwas excluded), to the pronunciation of a word, e.g. `(r)[ou]&(t)` ↔ `r * ou t`

[8] `,r` is written within the rounded curves while `,l` is symbolized by a larger hook.

[9] Not all of the $3\times 2^4\times 2^4$ thinkable prefix/suffix combinations can actually occur, e.g. at the beginning of English words only the following three consonant sequences `spr`, `str`, `skr`, `spl` and `skw` are possible [7]. Segments starting/ending with `,s`-circles are very common.

[10] then the notation ␣ or / is used

[11] PSL is classified as one of the so-called geometric shorthand systems, which contrast with cursive systems resembling smooth longhand writing.

[12] drawn either with thick or thin Metafont pens or filled.

[13] The most common “consonant sign” is the word space.

[14] With strokes `(f)` for ‘form’ and `(k)` for ‘company’, resp.

[15] See our project web site, and also `DEK.php` for the German shorthand DEK and `Gregg.php` for Gregg shorthand counterparts.

[16] the most frequent “(non)vowels”

[17] both to their spelling equivalent

[18] ‘LaTeX’ as `(l)[ei]&(t)[e]&(k)` vs. `(l)[ei]:t&[e](k)` and ‘computer’ as `^com(p)[yuu]&(t,r)` vs. `^com(p)[yuu]:tr`. The metaform can be interactively adjusted.

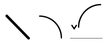
[19] here the first variant

[20] See [3] and the references there. We comment on these steps using our terminology.

[21] Lexical transducers carry out both (e.g. morphological) analysis and synthesis.

[22] Although the words shown have the same sequence of consonants, their outlines are distinct.

[23] ‘I, eye’, ‘wright, right, rite, write’, ‘hear, here’, ‘by, buy, bye’ are the most frequent.



Stanislav Jan Šarman
Computing Centre
Clausthal University of Technology
Erzstr. 51
38678 Clausthal
Germany
Sarman (at) rz dot tu-clausthal dot de
http://www3.rz.tu-clausthal.de/~rzsjs/steno/Pitman.php

# Optimizing PDF output size of TEX documents

**Abstract**
There are several tools for generating PDF output from a TEX document. By choosing the appropriate tools and configuring them properly, it is possible to reduce the PDF output size by a factor of 3 or even more, thus reducing document download times, hosting and archiving costs. We enumerate the most common tools, and show how to configure them to reduce the size of text, fonts, images and cross-reference information embedded into the final PDF. We also analyze image compression in detail.
We present a new tool called pdfsizeopt.py which optimizes the size of embedded images and Type 1 fonts, and removes object duplicates. We also propose a workflow for PDF size optimization, which involves configuration of TEX tools, running pdfsizeopt.py and the Multivalent PDF compressor as well.

## 1 Introduction

### 1.1 What does a PDF document contain

PDF is a popular document file format designed for printing and on-screen viewing. PDF faithfully preserves the design elements of the document, such as fonts, line breaks, page breaks, exact spacing, text layout, vector graphics and image resolution. Thus the author of a PDF document has precise control over the document's appearance—no matter what operating system or renderer software is used for viewing or printing the PDF. From the viewer's perspective, a PDF document is a sequence of rectangular pages containing text, vector graphics and pixel-based images. In addition, some rectangular page regions can be marked as hyperlinks, and Unicode annotations can also be added to the regions, so text may be copy-pasted from the documents. (Usually the copy-paste yields only a sequence of characters, with all formatting and positioning lost. Depending on the software and the annotation, the bold and italics properties can be preserved.) A tree-structured table of contents can be added as well, each node consisting of an unformatted caption and a hyperlink within the document.

Additional features of PDF include forms (the user fills some fields with data, clicks on the submit button, and the data is sent to a server in an HTTP request), event handlers in JavaScript, embedded multimedia files, encryption and access protection.

PDF has almost the same 2D graphics model (text, fonts, colors, vector graphics) as PostScript, one of the most widespread page description and printer control language. So it is possible to convert between PDF and PostScript without loss of information, except for a few constructs, e.g. transparency and color gradients are not supported by PostScript. Conversion from PDF to PostScript may blow up the file size if there are many repetitions in the PDF (e.g. a logo drawn to each page). Some of the interactive features of PDF (such as forms, annotations and bookmarks) have no PostScript equivalent either; other nonprintable elements (such as hyperlinks and the document outline) are supported in PostScript using `pdfmark`, but many PDF-to-PostScript converters just ignore them.

### 1.2 How to create PDF

Since PDF contains little or no structural and semantic information (such as in which order the document should be read, which regions are titles, how the tables are built and how the charts generated), word processors, drawing programs and typesetting systems usually can export to PDF, but for loading and saving they keep using their own file format which preserves semantics. PDF is usually not involved while the author is composing (or typesetting) the document, but once a version of a document is ready, a PDF can be exported and distributed. Should the author distribute the document in the native file format of the word processor, he might risk that the document doesn't get rendered as he intended, due to software version differences or because slightly different fonts are installed on the rendering computer, or the page layout settings in the word processor are different.

Most word processors and drawing programs and image editors support exporting as PDF. It is also possible to generate a PDF even if the software doesn't have a PDF export feature. For example, it may be possible to install a printer driver, which generates PDF instead of sending the document to a real printer. (For example, on Windows, PDFCreator [22] is such an open-source driver.) Some old programs can emit PostScript, but not PDF. The ps2pdf [28] tool (part of Ghostscript) can be used to convert the PostScript to PDF.

There are several options for PDF generation from TEX documents, including pdfTEX, dvipdfmx and dvips + ps2pdf. Depending on how the document uses hyperlinks and PostScript programming in graphics, some of these would not work. See the details in Subsection 2.1. See [13] for some more information about PDF and generating it with LATEX.

### 1.3 Motivation for making PDF files smaller

Our goal is to reduce the size of PDF files, focusing on those created from TEX documents. Having smaller PDF files reduces download times, web hosting costs and storage costs as well. Although there is no urgent need for reducing PDF storage costs for personal use (since hard drives in modern PCs are large enough), storage costs are significant for publishing houses, print shops, e-book stores and hosting services, libraries and archives [26]. Usually lots of copies and backups are made of PDF files originating from such places; saving 20% of the file size right after generating the PDF would save 20% of all future costs associated with the file.

Although e-book readers can store lots of documents (e.g. a 4 GB e-book reader can store 800 PDF books of 5 MB average reasonable file size), they get full quickly if we don't pay attention to optimized PDF generation. One can easily get a PDF file 5 times larger than reasonable by generating it with software which doesn't pay attention to size, or not setting the export settings properly. Upgrading or changing the generator software is not always feasible. A PDF recompressor becomes useful in these cases.

It is not our goal to propose or use alternative file formats, which support a more compact document representation or more aggressive compression than PDF. An example for such an approach is the Multivalent *compact PDF* file format [25], see Section 5 for more details. There is no technical reason against using a compact format for storage, and converting it on the fly to regular PDF before processing if needed. The disadvantage of a nonstandard compact format is that most PDF viewers and tools don't support it by default, so the user has to install and run the conversion tool, which some users can't or won't do just for viewing a PDF. When archiving compact PDF files for a long term, we have to make sure that we'll have a working converter at restore time. With Multivalent, this is possible by archiving the `.jar` file containing the code of the converter. But this may not suit all needs, because Multivalent is not open source, there are no alternative implementations, and there is no detailed open specification for its compact PDF file format.

A pixel-based (fixed resolution) alternative of PDF is DjVu (see Section 5).

It is possible to save space in a PDF by removing non-printed information such as hyperlinks, document outline elements, forms, text-to-Unicode mapping or user annotations. Removing these does not affect the output when the PDF is printed, but it degrades the user experience when the PDF is viewed on a computer, and it may also degrade navigation and searchability. Another option is to remove embedded fonts. In such a case, the PDF viewer will pick a font with similar metrics if the font is not installed on the viewer machine. Please note that unembedding the font doesn't change the horizontal distance between glyphs, so the page layout will remain the same, but maybe glyphs will look funny or hard-to-read. Yet another option to save space is to reduce the resolution of the embedded images. We will not use any of the techniques mentioned in this paragraph, because our goal is to reduce redundancy and make the byte representation more effective, while preserving visual and semantic information in the document.

### 1.4 PDF file structure

It is possible to save space in the PDF by serializing the same information more effectively and/or using better compression. This section gives a high-level introduction to the data structures and their serialization in the PDF file, focusing on size optimization. For a full description of the PDF file format, see [3].

PDF supports integer, real number, boolean, null, string and name as simple data types. A string is a sequence of 8-bit bytes. A name is also a sequence of 8-bit bytes, usually a concatenation of a few English words in CamelCase, often used as a dictionary key (e.g. `/MediaBox`) or an enumeration value (e.g. `/DeviceGray`). Composite data types are the list and the dictionary. A dictionary is an unordered sequence of key–value pairs, where keys must be names. Values in dictionaries and list items can be primitive or composite. There is a simple serialization of values to 8-bit strings, compatible with PostScript LanguageLevel 2. For example,

```
<</Integer 5 /Real -6.7 /Null null
  /StringInHex <Face> /String ((C)2009\\))
  /Boolean true /Name /Foo /List [3 4 5]>>
```

defines a dictionary with values of various types. All data types are immutable.

It is possible to define a value for future use by defining an *object*. For example, `12 0 obj [/PDF /Text] endobj` defines object number 12 to be an array of two items (`/PDF` and `/Text`). The number 0 in the definition is the so-called generation number, signifying that the object has not been modified since the PDF was generated. PDF makes it possible to store old versions of an object with different generation numbers, the one with the highest number being the most recent. Since most of the tools just create a new PDF instead of updating parts of an existing one, we can assume for simplicity that the generation number is always zero. Once an object is defined it is possible to refer to it (e.g. `12 0 R`) instead of typing its value. It is possible to define self-referential lists and dictionaries using object definitions. The PDF specification requires that some PDF structure elements (such as the `/FontDescriptor` value) be an indirect reference, i.e. defined as an object. Such elements cannot be inlined into other object, but they must be referred to.

A PDF file contains a header, a list of objects, a *trailer* dictionary, cross-reference information (offsets of object definitions, sorted by object number), and the end-of-file marker. The header contains the PDF version (PDF-1.7 being the latest). All of the file elements above except for the PDF version, the list of objects and the trailer are redundant, and can be regenerated if lost. The parsing of the PDF starts at the trailer dictionary. Its `/Root` value refers to the catalog dictionary object, whose `/Pages` value refers to a dictionary object containing the list of pages. The interpretation of each object depends on the reference path which leads to that object from the trailer. In addition, dictionary objects may have the `/Type` and/or `/Subtype` value indicating the interpretation. For example, `<</Subtype/Image ...>>` defines a pixel-based image.

In addition to the data types above, PDF supports streams as well. A *stream* object is a dictionary augmented by the stream data, which is a byte sequence. The syntax is `X Y obj << dict-items >> stream stream-data endstream endobj`. The stream data can be compressed or otherwise encoded (such as in hex). The `/Filter` and `/DecodeParms` values in the dictionary specify how to uncompress/decode the stream data. It is possible to specify multiple such filters, e.g. `/Filter [/ASCIIHexDecode /FlateDecode]` says that the bytes after `stream` should be decoded as a hex string, and then uncompressed using PDF's ZIP implementation. (Please note that the use of `/ASCIIHexDecode` is just a waste of space unless one wants to create an ASCII PDF file.) The three most common uses for streams are: image pixel data, embedded font files and content streams. A content stream contains the instructions to draw the contents of the page. The stream data is ASCII, with a syntax similar to PostScript, but with different operators. For example, `BT/F 20 Tf 1 0 0 1 8 9 Tm(Hello world)Tj ET` draws the text "Hello World" with the font `/F` at size 20 units, shifted up by 8 units, and shifted right by 9 units (according to the transformation matrix `1 0 0 1 8 9`).

Streams can use the following generic compression methods: ZIP (also called flate), LZW and RLE (run-length encoding). ZIP is almost always superior. In addition to those, PDF supports some image-specific compression methods as well: JPEG and JPEG2000 for true-color images and JBIG2 and G3 fax (also called CCITT fax) for bilevel (two-color) images. JPEG and JPEG2000 are lossy methods, they usually yield the same size at the same quality settings—but JPEG2000 is more flexible. JBIG2 is superior to G3 fax and ZIP for bilevel images. Any number of compression filters can be applied to a stream, but usually applying more than one yields a larger compressed stream size than just applying one. ZIP and LZW support predictors as well. A predictor is an easy-to-compute, invertible filter which is applied to the stream data before compression, to make the data more compressible. One possible predictor subtracts the previous data value from the current one, and sends the difference to the compressor. This helps reduce the file size if the difference between adjacent data values is mostly small, which is true for some images with a small number of colors.

There is cross-reference information near the end of the PDF file, which contains the start byte offset of all object definitions. Using this information it is possible to render parts of the file, without reading the whole file. The most common format for cross-reference information is the *cross-reference table* (starting with the keyword `xref`). Each item in the table consumes 20 bytes, and contains an object byte offset. The object number is encoded by the position of the item. For PDFs with several thousand objects, the space occupied by the cross-reference table is not negligible. PDF 1.5 introduces *cross-reference streams,* which store the cross-reference information in compact form in a stream. Such streams are usually compressed as well, using ZIP and a predictor. The benefit of the predictor is that adjacent offsets are close to each other, so their difference will contain lots of zeros, which can be compressed better.

Compression cannot be applied to the PDF file as a whole, only individual parts (such as stream data and cross-reference information) can be compressed. However, there can be lots of small object definitions in the file which are not streams. To compress those, PDF 1.5 introduces *object streams.* The data in an object stream contains a concatenation of any number of non-stream object definitions. Object streams can be compressed just as regular stream data. This makes it possible to squeeze repetitions spanning over multiple object definitions. Thus, with PDF 1.5, most of the PDF file can be stored in compressed streams. Only a few dozen header bytes and end-of-file markers and the stream dictionaries remain uncompressed.

Table 1: Output file sizes of PDF generation from *The TEXbook*, with various methods. The PDF was optimized with pdfsizeopt.py, then with Multivalent.

| *method* | *PDF bytes* | *optimized PDF bytes* |
|---|---|---|
| pdfTEX | 2283510 | 1806887 |
| dvipdfm | 2269821 | 1787039 |
| dvipdfmx | 2007012 | 1800270 |
| dvips+ps2pdf | 3485081 | 3181869 |

# 2 Making PDF files smaller

## 2.1 How to prepare a small, optimizable PDF with TEX

When aiming for a small PDF, it is possible to get it by using the best tools with the proper settings to create the smallest possible PDF from the start. Another approach is to create a PDF without paying attention to the tools and their settings, and then optimize PDF with a PDF size optimizer tool. The approach we suggest in this paper is a mixture of the two: pay attention to the PDF generator tools and their fundamental settings, so generating a PDF which is small enough for temporary use and also easy to optimize further; and use an optimizer to create the final, even smaller PDF.

This section enumerates the most common tools which can generate the temporary PDF from a `.tex` source. As part of this, it explains how to enforce the proper compression and font settings, and how to prepare vector and pixel-based images so they don't become unnecessarily large.

*Pick the best PDF generation method.* Table 2 lists features of the 3 most common methods (also called *drivers*) which produce a PDF from a TEX document, and Table 1 compares the file size they produce when compiling *The TEXbook*. There is no single best driver because of the different feature sets, but looking at how large the output of dvips is, the preliminary conclusion would be to use pdfTEX or dvipdfm(x) except if advanced PostScript features are needed (such as for psfrag and pstricks).

We continue with presenting and analyzing the methods mentioned.

**dvips** This approach converts TEX source → DVI → PostScript → PDF, using dvips [29] for creating the PostScript file, and ps2pdf [28] (part of Ghostscript) for creating the PDF file. Example command-lines for compiling `doc.tex` to `doc.pdf`:

```
$ latex doc
$ dvips doc
$ ps2pdf14 -d{\PDF}SETTINGS=/prepress doc.ps
```

Table 2. Features supported by various PDF output methods.

| *Feature* | pdfTEX | dvipdfm(x) | dvips |
|---|---|---|---|
| hyperref | + | + | + |
| Ti*k*Z | + | + | + |
| beamer.cls | + | $+^{o}$ | $+^{u}$ |
| include PDF | + | $+^{b}$ | + |
| embed bitmap font | + | + | + |
| embed Type 1 font | + | + | + |
| embed TrueType font | + | + | − |
| include EPS | − | + | + |
| include JPEG | + | $+^{x}$ | − |
| include PNG | + | $+^{x}$ | − |
| include MetaPost | $+^{m}$ | $+^{m}$ | $+^{r}$ |
| psfrag | $-^{f}$ | $-^{f}$ | + |
| pstricks | $-^{f}$ | $-^{f}$ | + |
| pdfpages | + | − | − |
| line break in link | + | + | − |

*b:* bounding box detection with ebb or pts-graphics-helper
*f:* see [21] for workarounds
*m:* convenient with `\includegraphicsmps` defined in pts-graphics-helper
*r:* rename file to `.eps` manually
*o:* with `\documentclass[dvipdfm]{beamer}`
*u:* use `dvips -t unknown doc.dvi` to get the paper size right.
*x:* with `\usepackage[dvipdfmx]{graphics}` and shell escape running extractbb

**dvipdfmx** The tool dvipdfmx [7] converts from DVI to PDF, producing a very small output file. dvipdfmx is part of TEX Live 2008, but since it's quite new, it may be missing from other TEX distributions. Its predecessor, dvipdfm has not been updated since March 2007. Notable new features in dvipdfmx are: support for non-latin scripts and fonts; emitting the Type 1 fonts in CFF (that's the main reason for the size difference in Table 2); parsing pdfTEX-style font `.map` files. Example command-lines:

```
$ latex doc
$ dvipdfmx doc
```

**pdfTEX** The commands pdftex or pdflatex [41] generate PDF directly from the `.tex` source, without any intermediate files. An important advantage of pdfTEX over the other methods is that it integrates nicely with the editors TEXShop and TEXworks. The single-step approach ensures that there would be no glitches (e.g. images misaligned or not properly sized) because the tools are not integrated properly. Example command-line:

```
$ pdflatex doc
```

The command `latex doc` is run for both dvips and dvipdfm(x). Since these two drivers expect a bit different \specials in the DVI file, the driver name has to be communicated to the TEX macros generating the \specials. For LATEX, dvips is the default. To get dvipdfm(x) right, pass dvipdfm (or dvipdfmx) as an option to \documentclass or to both \usepackage{graphicx} and \usepackage{hyperref}. The package pts-graphics-helper [34] sets up dvipdfm as default unless the document is compiled with pdflatex.

Unfortunately, some graphics packages (such as psfrag and pstricks) require a PostScript backend such as dvips, and pdfTEX or dvipdfmx don't provide that. See [21] for a list of workarounds. They rely on running dvips on the graphics, possibly converting its output to PDF, and then including those files in the main compilation. Most of the extra work can be avoided if graphics are created as external PDF files (without text replacements), Ti*k*Z [8] figures or MetaPost figures. Ti*k*Z and MetaPost support text captions typeset by TEX. Inkscape users can use textext [46] within Inkscape to make TEX typeset the captions.

The \includegraphics command of the standard graphicx LATEX-package accepts a PDF as the image file. In this case, the first page of the specified PDF will be used as a rectangular image. With dvipdfm(x), one also needs a .bb (or .bbx) file containing the bounding box. This can be generated with the ebb tool (or the extractbb tool shipping with dvipdfm(x). Or, it is possible to use the pts-graphics-helper package [34], which can find the PDF bounding box directly (most of the time).

dvipdfm(x) contains special support for embedding figures created by MetaPost. For pdfTEX, the graphicx package loads supp-pdf.tex, which can parse the output of MetaPost, and embed it to the document. Unfortunately, the graphicx package is not smart enough to recognize MetaPost output files (jobname.1, jobname.2 etc.) by extension. The pts-graphics-helper package overcomes this limitation by defining \includegraphicsmps, which can be used in place of \includegraphics for including figures created by MetaPost. The package works consistently with dvipdfm(x) and pdfTEX.

With pdfTEX, it is possible to embed page regions from an external PDF file, using the pdfpages LATEX-package. Please note that due to a limitation in pdfTEX, hyperlinks and outlines (table of contents) in the embedded PDF will be lost.

Although dvipdfm(x) supports PNG and JPEG image inclusion, calculating the bounding box may be cumbersome. It is recommended that all external images should be converted to PDF first. The recommended software for that conversion is sam2p [38, 39], which creates a small PDF (or EPS) quickly.

Considering all of the above, we recommend using pdfTEX for compiling TEX documents to PDF. If, for some reason, using pdfTEX is not feasible, we recommend dvipdfmx from TEX Live 2008 or later. If a 1% decrease in file size is worth the trouble of getting fonts right, we recommend dvipdfm. In all these cases, the final PDF should be optimized with pdfsizeopt.py (see later).

*Get rid of complex graphics.* Some computer algebra programs and vector modeling tools emit very large PDF (or similar vector graphics) files. This can be because they draw the graphics using too many little parts (e.g. they draw a sphere using several thousand triangles), or they draw too many parts which would be invisible anyway since other parts cover them. Converting or optimizing such PDF files usually doesn't help, because the optimizers are not smart enough to rearrange the drawing instructions, and then skip some of them. A good rule of thumb is that if a figure in an optimized PDF file is larger than the corresponding PNG file rendered in 600 DPI, then the figure is too complex. To reduce the file size, it is recommended to export the figure as a PNG (or JPEG) image from the program, and embed that bitmap image.

*Downsample high-resolution images.* For most printers it doesn't make a visible difference to print in a resolution higher than 600 DPI. Sometimes even the difference between 300 DPI and 600 DPI is negligible. So converting the embedded images down to 300 DPI may save significant space without too much quality degradation. Downsampling before the image is included is a bit of manual work for each image, but there are a lot of free software tools to do it (such as GIMP [10] and the convert tool of ImageMagick ). It is possible to downsample after the PDF has been created, for example with the commercial software PDF Enhancer [20] or Adobe Acrobat. ps2pdf (using Ghostscript's -dDEVICE=pdfwrite, and setdistillerparams to customize, see parameters in [28]) can read PDF files, and downsample images within as well, but it usually grows other parts of the file too much (15% increase in file size for *The TEXbook*), and it may lose some information (it does keep hyperlinks and the document outline, though).

*Crop large images.* If only parts of a large image contain useful and relevant information, one can save space by cropping the image.

*Choose the JPEG quality.* When using JPEG (or JPEG2000) compression, there is a tradeoff between quality and file size. Most JPEG encoders based on libjpeg accept an integer quality value between 1 and 100. For true color photos, a quality below 40 produces a severely degraded, hard-to-recognize image, with 75 we get some harmless glitches, and with 85 the degradation is hard to notice. If the document contains lots of large JPEG images, it is worth reencoding those with a lower quality setting to get a smaller PDF file. PDF Enhancer can reencode JPEG images in an existing PDF, but sometimes not all the images have to be reencoded. With GIMP it is possible to get a real-time preview of the quality degradation before saving, by moving the quality slider.

Please note that some cameras don't encode JPEG files effectively when saving to the memory card, and it is possible to save a lot of space by reencoding on the computer, even with high quality settings.

*Optimize poorly exported images.* Not all image processing programs pay attention to size of the image file they save or export. They might not use compression by default; or they compress with suboptimal settings; or (for EPS files) they try to save the file in some compatibility mode, encoding and compressing the data poorly; or they add lots of unneeded metadata. These poorly exported images make TeX and the drivers run slowly, and they waste disk space (both on the local machine and in the revision control repository). A good rule of thumb to detect a poorly exported image is to use sam2p to convert the exported image to JPEG and PNG (`sam2p -c ijg:85 exported.img test.jpg`; `sam2p exported.img test.png`), and if any of these files is a lot smaller than the exported image, then the image was exported poorly.

Converting the exported image with sam2p (to any of EPS, PDF, JPEG and PNG) is a fast and effective way to reduce the exported image size. Although sam2p, with its default settings, doesn't create the smallest possible file, it runs very quickly, and it creates an image file which is small enough to be embedded in the temporary PDF.

*Embed vector fonts instead of bitmap fonts.* Most fonts used with TeX nowadays are available in Type 1 vector format. (These fonts include the Computer Modern families, the Latin Modern families, the URW versions of the base 14 and some other Adobe fonts, the TeX Gyre families, the Vera families, the Palatino family, the corresponding math fonts, and some symbol and drawing fonts.) This is a significant shift from the original TeX (+ dvips) concept, which used bitmap fonts generated by MetaFont. While drivers still support embedding bitmap fonts to the PDF, this is not recommended, because bitmaps (at 600 DPI) are larger than their vector equivalent, they render more slowly and they look uglier in some PDF viewers.

If a font is missing from the font `.map` file, drivers tend to generate a bitmap font automatically, and embed that. To make sure this didn't happen, it is possible to detect the presence of bitmap fonts in a PDF by running `grep -a "/Subtype */Type3" doc.pdf`. Here is how to instruct pdfTeX to use bitmap fonts only (for debugging purposes): `pdflatex "\pdfmapfile\input" doc`. The most common reason for the driver not finding a corresponding vector font is that the `.map` file is wrong or the wrong map file is used. With TeX Live, the updmap tool can be used to regenerate the `.map` files for the user, and the updmap-sys command regenerates the system-level `.map` files. Table 3 shows which driver reads which `.map` file. Copying over pdftex_dl14.map to the current directory as the driver-specific `.map` file usually makes the driver find the font. Old TeX distributions had quite a lot of problems finding fonts, upgrading to TeX Live 2008 or newer is strongly recommended.

Table 3: Font `.map` files used by various drivers and their symlink targets (default first) in TeX Live 2008.

| *Driver* | Font `.map` file |
|---|---|
| xdvi | ps2pk.map |
| dvips | psfonts.map →<br>psfonts_t1.map \| (psfonts_pk.map) |
| pdfTeX | pdftex.map →<br>pdftex_dl14.map \| (pdftex_ndl14.map) |
| dvipdfm(x) | dvipdfm.map →<br>dvipdfm_dl14.map \| (dvipdfm_ndl14.map) |

Some other popular fonts (such as the Microsoft web fonts) are available in TrueType, another vector format. dvipdfm(x) and pdfTeX can embed TrueType fonts, but dvips cannot (it just dumps the `.ttf` file to the `.ps` file, rendering it unparsable).

OpenType fonts with advanced tables for script and feature selection and glyph substitution are supported by Unicode-aware TeX-derivatives such as XeTeX, and also by dvipdfmx.

*Omit the base 14 fonts.* The base 14 fonts are Times (in 4 styles, Helvetica (in 4 styles), Courier (in 4 styles), Symbol and Zapf Dingbats. To reduce the size of the PDF, it is possible to omit them from the PDF file, because PDF viewers tend to have them. However, omitting the base 14 fonts is deprecated since PDF 1.5. Adobe Reader 6.0 or newer, and other PDF viewers (such as xpdf and evince) don't contain those fonts either, but they can find them as system fonts. On Debian-based Linux systems, those fonts are in the gsfonts package.

In TeX Live, directives *pdftexDownloadBase14* and *dvipdfmDownloadBase14* etc. in the configuration file `texmf-config/web2c/updmap.cfg` specify whether to embed the base 14 fonts. After modifying this file (either the system-wide one or the one in `$HOME/.texlive2008`) and running the updmap command, the following font map files would be created:

**pdftex_dl14.map** Font map file for pdfTEX with the base 14 fonts embedded. This is the default.

**pdftex_ndl14.map** Font map file for pdfTEX with the base 14 fonts omitted.

**pdftex.map** Font map file used by pdfTEX by default. Identical to one of the two above, based on the *pdftexDownloadBase14* setting.

**dvipdfm_dl14.map** Font map file for dvipdfm(x) with the base 14 fonts embedded. This is the default.

**dvipdfm_ndl14.map** Font map file for dvipdfm(x) with the base 14 fonts omitted.

**dvipdfm.map** Font map file used by dvipdfm(x) by default. Identical to one of the two above, based on the *dvipdfmDownloadBase14* setting.

It is possible to specify the base 14 embedding settings without modifying configuration files or generating .map files. Example command-line for pdfTEX (type it without line breaks):

```
pdflatex "\pdfmapfile{pdftex_ndl14.map}
         \input" doc.tex
```

However, this will display a warning *No flags specified for non-embedded font.* To get rid of this, use

```
pdflatex "\pdfmapfile{=
         pdftex_ndl14_extraflag.map}
         \input" doc.tex
```

instead. Get the .map file from [34].

The .map file syntax for dvipdfm is different, but dvipdfmx can use a .map file of pdfTEX syntax, like this:

```
dvipdfmx -f pdftex_dl14.map doc.dvi
```

Please note that dvipdfm loads the *.map* files specified in dvipdfmx.cfg first, and the .map files loaded with the -f flag override entries loaded previously, from the configuration file. To have the base 14 fonts omitted, run (without a line break):

```
dvipdfmx -f pdftex_ndl14.map
    -f dvipdfmx_ndl14_extra.map doc.tex
```

Again, you can get the last .map file from [34]. Without dvipdfmx_ndl14_extra.map, a bug in dvipdfm prevents it from writing a PDF file without the font—it would embed a rendered bitmap font instead.

*Subset fonts. Font subsetting* is the process when the driver selects and embeds only the glyphs of a font which are actually used in the document. Font subsetting is turned on by default for dvips, dvipdfm(x) and pdfTEX when emitting glyphs produced by TEX.

## 2.2 Extra manual tweaks on TEX-to-PDF compilation

This sections shows a couple of methods to reduce the size of the PDF created by a TEX compilation manually. It is not necessary to implement these methods if the temporary PDF gets optimized by pdfsizeopy.py + Multivalent, because this combination implements the methods discussed here.

*Set the ZIP compression level to maximum.* For pdfTEX, the assignment \pdfcompresslevel9 selects maximum PDF compression. With TEX Live 2008, this is the default. Here is how to specify it on the command-line (without line breaks):

```
pdflatex "\pdfcompresslevel9
         \input" doc.tex
```

For dvipdfm(x), the command-line flag -z9 can be used to maximize compression. This is also the default. PDF itself supports redundancy elimination in many different places (see in Subsection 2.3) in addition to setting the ZIP compression level.

There is no need to pay attention to this tweak, because Multivalent recompresses all ZIP streams with maximum effort.

*Generate object streams and cross-reference streams.* pdfTEX can generate object streams and cross-reference streams to save about 10% of the PDF file size, or even more if the file contains lots of hyperlinks. (The actual saving depends on the file structure.) Example command-line for enabling it (without line breaks):

```
pdflatex "\pdfminorversion5
         \pdfobjcompresslevel3
         \input" doc.tex
```

According to [27], if ZIP compression is used to compress the object streams, in some rare cases it is possible to save space by starting a new block within the ZIP stream just at the right points.

There is no need to pay attention to this tweak, because Multivalent generates object streams and cross-reference streams by default.

*Encode Type 1 fonts as CFF.* CFF [2] (Type 2 or /Subtype /Type1C) is an alternative, compact, highly compressible binary font format that can represent Type 1 font data without loss. By embedding vector fonts in CFF instead of Type 1, one can save significant portion of the PDF file, especially if the document is 10 pages or less (e.g. reducing the PDF file size from 200 kB to 50 kB). dvipdfmx does this by default, but the other drivers (pdfTEX, dvipdfm, ps2pdf with dvips) don't support CFF embedding so far.

There is no need to pay attention to this tweak, because pdfsizeopt.py converts Type 1 fonts in the PDF to CFF.

*Create graphics with font subsetting in mind.* For glyphs coming from external sources such as the included

PostScript and PDF graphics, the driver is usually not smart enough to recognize the fonts already embedded, and unify them with the fonts in the main document. Let's suppose that the document contains included graphics with text captions, each graphics source PostScript or PDF having the font subsets embedded. No matter whether dvips, dvipdfm(x) or pdfTEX is the driver, it will not be smart enough to unify these subsets to a single font. Thus space would be wasted in the final PDF file containing multiple subsets of the same font, possibly storing duplicate versions of some glyphs.

It is possible to avoid this waste by using a graphics package implemented in pure TEX (such as Ti*k*Z) or using MetaPost (for which there is special support in dvips, dvipdfm(x) and pdfTEX to avoid font and glyph duplication). The package psfrag doesn't suffer from this problem either if the EPS files don't contain any embedded fonts.

There is no need to pay attention to this tweak, because pdfsizeopt.py unifies font subsets.

*Disable font subsetting before concatenation.* If a PDF document is a concatenation of several smaller PDF files (such as in journal volumes and conference proceeding), and each PDF file contains its own, subsetted fonts, then it depends on the concatenator tool whether those subsets are unified or not. Most concatenator tools (pdftk, Multivalent, pdfpages, ps2pdf; see [32] for more) don't unify these font subsets.

However, if you use ps2pdf for PDF concatenation, you can get font subsetting and subset unification by *disabling* font subsetting when generating the small PDF files. In this case, Ghostscript (run by ps2pdf) will notice that the document contains the exact same font many times, and it will subset only one copy of the font.

There is no need to pay attention to this tweak, because pdfsizeopt.py unifies font subsets.

*Embed each graphics file once.* When the same graphics file (such as the company logo on presentation slides) is included multiple times, it depends on the driver whether the graphics data is duplicated in the final PDF. pdfTEX doesn't duplicate, dvipdfm(x) duplicates only MetaPost graphics, and dvips always duplicates.

There is no need to pay attention to this tweak, because both pdfsizeopt.py and Multivalent eliminate duplicates of identical objects.

## 2.3 How PDF optimizers save space

This subsection describes some methods PDF optimizers use to reduce the file size. We focus on ideas and methods relevant to TEX documents.

*Use cross-reference streams compressed with the $y$-predictor.* Each offset entry in an (uncompressed) cross-reference table consumes 20 bytes. It can be reduced by using compressed cross-reference streams, and enabling the $y$-predictor. As shown in column *xref* of Table 4, a reduction factor of 180 is possible if the PDF file contains many objects (e.g. more than $10^5$ objects in *pdfref*, with less than 12000 bytes in the cross-reference stream).

The reason why the $y$-predictor can make a difference of a factor of 2 or even more is the following. The $y$-predictor encodes each byte in a rectangular array of bytes by subtracting the original byte above the current byte from the current byte. So if each row of the rectangular array contains an object offset, and the offsets are increasing, then most of the bytes in the output of the $y$-predictor would have a small absolute value, mostly zero. Thus the output of the $y$-predictor can be compressed better with ZIP than the original byte array.

Some tools such as Multivalent implement the $y$-predictor with PNG predictor 12, but using TIFF predictor 2 avoids stuffing in the extra byte per each row—pdfsizeopt.py does that.

*Use object streams.* It is possible to save space in the PDF by concatenating small (non-stream) objects to an object stream, and compressing the stream as a whole. One can even sort objects by type first, so similar objects will be placed next to each other, and they will fit to the 32 kB long ZIP compression window.

Please note that both object streams and cross-reference streams are PDF 1.5 features, and cross-reference streams must be also used when object streams are used.

*Use better stream compression.* In PDF any stream can be compressed with any compression filter (or a combination of filters). ZIP is the most effective general-purpose compression, which is recommended for compressing content streams, object streams, cross-reference streams and font data (such as CFF). For images, however, there are specialized filters (see later in this section).

Most PDF generators (such as dvipdfm(x) and pdfTEX) and optimization tools (such as Multivalent) use the zlib code for general-purpose ZIP compression. zlib lets the user specify the *effort* parameter between 0 (no compression) and 9 (slowest compression, smallest output) to balance compression speed versus compressed data size. There are, however alternative ZIP compressor implementations (such as the one in KZIP [30] and PNGOUT [31, 9]), which provide an even higher effort—but the author doesn't know of any PDF optimizers using those algorithms.

*Recompress pixel-based images.* PDF supports more than 6 compression methods (and any combination of them) and more than 6 predictors, so there are lots of possibilities to make images smaller. Here we focus on lossless compression (thus excluding JPEG and JPEG2000 used for compressing photos). An image is rectangular array of pixels. Each pixel is encoded as a vector of one or more components in the color space of the image. Typical color spaces are RGB (/DeviceRGB), grayscale (/DeviceGray), CMYK (/DeviceCMYK), color spaces where colors are device-independent, and the palette (indexed) versions of those. Each color component of each pixel is encoded as a nonnegative integer with a fixed number of bits (bits-per-component, BPC; can be 1, 2, 4, 8, 12 or 16). The image data can be compressed with any combination of the PDF compression methods.

Before recompressing the image, usually it is worth extracting the raw RGB or CMYK (or device-independent) image data, and then compressing the image the best we can. Partial approaches such as optimizing the palette only are usually suboptimal, because they may be incapable of converting an indexed image to grayscale to save the storage space needed by the palette.

To pick the best encoding for the image, we have to decide which color space, bits-per-component, compression method(s) and predictor to use. We have to choose a color space which can represent all the colors in the image. We may convert a grayscale image to an RGB image (and back if all pixels are grayscale). We may also convert a grayscale image to a CMYK image (and maybe back). If the image doesn't have more than 256 different colors, we can use an indexed version of the color space. A good rule of thumb (no matter the compression) is to pick the color space + bits-per-component combination which needs the least number of bits per pixel. On a draw, pick the one which doesn't need a palette. These ideas can also be applied if the image contains an alpha channel (which allows for transparent or semi-transparent pixels).

It is possible to further optimize some corner cases, for example if the image has only a single color, then it is worth encoding it as vector graphics filling a rectangle of that color. Or, when the image is a grid of rectangles, where each rectangle contains a single color, then it is worth encoding a lower resolution image, and increase the scale factor in the image transformation matrix to draw the larger image.

High-effort ZIP is the best compression method supported by PDF, except for bilevel (two-color) images, where JBIG2 can yield a smaller result for some inputs. JBIG2 is most effective on images with lots of 2D repetitions, e.g. images containing lots of text (because the letters are repeating). Other lossless compression methods supported by PDF (such as RLE, LZW and G3 fax) are inferior to ZIP and/or JBIG2. Sometimes the image is so small (like $10 \times 10$ pixels) that compressing would increase its size. Most of the images don't benefit from a predictor (used together with ZIP compression), but some of them do. PDF supports the PNG predictor image data format, which makes it possible to choose a different predictor for scanline (image row). The heuristic default algorithm in pnmtopng calculates all 5 scanline variations, and picks the one having the smallest sum of absolute values. This facilitates bytes with small absolute values in the uncompressed image data, so the Huffman coding in ZIP can compress it effectively.

Most of the time it is not possible to tell in advance if ZIP or JBIG2 should be used, or whether a predictor should be used with ZIP or not. To get the smallest possible output, it is recommended to run all 3 variations and pick the one yielding the smallest image object. For very small images, the uncompressed version should be considered as well. If the image is huge and it has lots repetitive regions, it may be worth to apply ZIP more than once. Please note that metadata (such as specifying the decompression filter(s) to use) also contributes to the image size.

Most PDF optimizers use the zlib code for ZIP compression in images. The output of some other image compressors (most notably PNGOUT [31], see also OptiPNG [43] and [42] for a list of 11 other PNG optimization tools, and more tools in [15]) is smaller than what zlib produces with its highest effort, but those other compressors usually run a 100 times or even slower than zlib.

How much a document size decreases because of image recompression depends on the structure of the document (how many images are there, how large the images are, how large part of the file size is occupied by images) and how effectively the PDF was generated. The percentage savings in the *image* column of Table 4 suggests that only a little saving is possible (about 5%) if the user pays attention to embed the images effectively, according to the image-related guidelines presented in Section 2.1. It is possible to save lots of space by decreasing the image resolution, or decreasing the image quality by using some lossy compression method (such as JPEG or JPEG2000) with lower quality settings. These kinds of optimizations are supported by Adobe Acrobat Pro and PDF Enhancer, but they are out of scope of our goals to decrease the file size while not changing its rendered appearance.

JPEG files could benefit from a lossless transformation, such as removing EXIF tags and other metadata. Compressing JPEG data further with ZIP wouldn't save space. The program packJPG [33] applies custom lossless compression to JPEG files, saving about 20%. Unfortunately, PDF doesn't have a decompression filter for that.

*Convert some inline images to objects.* It is possible to inline images into content streams. This PDF feature saves about 30 bytes per image as compared to having the image as a standalone image object. However, inline images cannot be shared. So in order to save the most space, inline images which are used more than once should be converted to objects, and image objects used only once should be converted to inline images. Images having palette duplication with other images should be image objects, so the palette can be shared.

*Unify duplicate objects.* If two or more PDF objects share the same serialized value, it is natural to save space by keeping only the first one, and modifying references to the rest so that they refer to the first one. It is possible to optimize even more by constructing equivalence classes, and keeping only one object per class. For example, if the PDF contains

```
5 0 obj << /Next 6 0 R /Prev 5 0 R >> endobj
6 0 obj << /Next 5 0 R /Prev 6 0 R >> endobj
7 0 obj << /First 6 0 R >> endobj
```

then objects 5 and 6 are equivalent, so we can rewrite the PDF to

```
5 0 obj << /Next 5 0 R /Prev 5 0 R >> endobj
7 0 obj << /First 5 0 R >> endobj
```

PDF generators usually don't emit duplicate objects on purpose, but it just happens by chance that some object values are equal. If the document contains the same page content, font, font encoding, image or graphics more than once, and the PDF generator fails to notice that, then these would most probably become duplicate objects, which can be optimized away. The method dvips + ps2pdf usually produces lots of duplicated objects if the document contains lots of duplicate content such as \includegraphics loading same graphics many times.

*Remove image duplicates, based on visible pixel value.* Different color space, bits-per-pixel and compression settings can cause many different representations of the same image (rectangular pixel array) to be present in the document. This can indeed happen if different parts of the PDF were created with different (e.g. one with pdfTEX, another with dvips), and the results were concatenated. To save space, the optimizer can keep only the smallest image object, and update references.

*Remove unused objects.* Some PDF files contain objects which are not reachable from the /Root or trailer objects. These may be present because of incremental updates, concatenations or conversion, or because the file is a linearized PDF. It is safe to save space by removing those unused objects. A linearized PDF provides a better web experience to the user, because it makes the first page of the PDF appear earlier. Since a linearized PDF can be automatically generated from a non-linearized one any time, there is no point keeping a linearized PDF when optimizing for size.

*Extract large parts of objects.* Unifying duplicate objects can save space only if a whole object is duplicated. If a paragraph is repeated on a page, it will most probably remain duplicated, because the duplication is within a single object (the content stream). So the optimizer can save space by detecting content duplication in the sub-object level (outside stream data and inside content stream data), and extracting the duplicated parts to individual objects, which can now be unified. Although this extraction would usually be too slow if applied to all data structures in the PDF, it may be worth applying it to some large structures such as image palettes (whose maximum size is 768 bytes for RGB images).

*Reorganize content streams and form XObjects.* Instructions for drawing a single page can span over multiple content streams and form XObjects. To save space, it is possible to concatenate those to a single content stream, and compress the stream at once. After all those concatenations, large common instruction sequences can be extracted to form XObjects to make code reuse possible.

*Remove unnecessary indirect references.* The PDF specification defines whether a value within a compound PDF value must be an indirect reference. If a particular value in the PDF file is an indirect reference, but it doesn't have to be, and other objects are not referring to that object, then inlining the value of the object saves space. Some PDF generators emit lots of unnecessary indirect references, because they generate the PDF file sequentially, and for some objects they don't know the full value when they are generating the object—so they replace parts of the value by indirect references, whose definitions they give later. This strategy can save some RAM during the PDF generation, but it makes the PDF about 40 bytes larger than necessary for each such reference.

*Convert Type 1 fonts to CFF.* Since drivers embed Type 1 fonts to the PDF as Type 1 (except for dvipdfmx, which emits CFF), and CFF can represent the same font with less bytes (because of the binary format and the smart defaults), and it is also more compressible (because it doesn't have encryption), it is natural to save space by converting Type 1 fonts in the PDF to CFF.

*Subset fonts.* This can be done by finding unused glyphs in fonts, and getting rid of them. Usually this doesn't save any space for TEX documents, because drivers subset fonts by default.

*Unify subsets of the same font.* As discussed in Section 2.1, a PDF file may end up containing multiple subsets of the same font when typesetting a collection of articles (such as a journal volume or a conference proceedings) with LaTeX, or embedding graphics containing text captions. Since these subsets are not identical, unifying duplicate objects will not collapse them to a single font. A font-specific optimization can save file size by taking a union of these subsets in each font, thus eliminating glyph duplication and improving compression effectiveness by grouping similar data (font glyphs) next to each other.

*Remove data ignored by the PDF specification.* For compatibility with future PDF specification versions, a PDF viewer or printer must accept dictionary keys which are not defined in the PDF specification. These keys can be safely removed without affecting the meaning of the PDF. An example for such a key is `/PTEX.Fullbanner` emitted by pdfTeX.

*Omit explicitly specified default values.* The PDF specification provides default values for many dictionary keys. Some PDF generators, however, emit keys with the default value. It is safe to remove these to save space.

*Recompress streams with ZIP.* Uncompressing a stream and recompressing it with maximum-effort ZIP makes the stream smaller most of the time. That's because ZIP is more effective than the other general purpose compression algorithms PDF supports (RLE and LZW).

For compatibility with the PostScript language, PDF supports the `/ASCIIHexDecode` and `/ASCII85Decode` filters on streams. Using them just makes the stream in the file longer (by a factor of about 2/1 and 5/4, respectively). These filters make it possible to embed binary stream data in a pure ASCII PDF file. However, there is no significant use case for an ASCII-only PDF nowadays, so it is recommended to get rid of these filters to decrease to file size.

*Remove page thumbnails.* If the PDF file has page thumbnails, the PDF viewer can show them to the user to make navigation easier and faster. Since page thumbnails are redundant information which can be regenerated any time, it is safe to save space by removing them.

*Serialize values more effectively.* Whitespace can be omitted between tokens, except between a name token and a token starting with a number or a letter (e.g. `/Ascent 750`). Whitespace in front of `endstream` can be omitted as well. The binary representation of strings should be used instead of the hexadecimal, because it's never longer and it's shorter most of the time if used properly. Only the characters `( \ )` have to be escaped with a backslash within strings, but parentheses which nest can be left unescaped. So, e.g. the string `a(()))(()\b` can be represented as `(a(())\)(\(\\b)`.

*Shrink cross-reference data.* Renumbering objects (from 1, consecutively) saves space in the cross-reference data, because gaps don't have to be encoded. (Each gap of consecutive missing objects costs about 10 bytes.) Also if an object is referenced many times, then giving it a small object number reduces the file size by a few bytes.

*Remove old, unused object versions.* PDF can store old object versions in the file. This makes incremental updates (e.g. the *File / Save* action in Adobe Acrobat) faster. Removing the old versions saves space.

*Remove content outside the page.* `/MediaBox`, `/CropBox` and other bounding box values of the page define a rectangle where drawing takes place. All content (vector graphics or parts of it, images or parts of them, or text) than falls outside this rectangle can be removed to save space. Implementing this removal can be tricky for partially visible content. For example, 8-pixel wide bars can be removed from the edge of a JPEG image without quality loss in the remaining part.

*Remove unused named destinations.* A named destination maps a name to a document location or view. It can be a target of a hyperlink within the document, or from outside. Some PDF generator software (such as FrameMaker) generates lots of named destinations never referenced. But care has to be taken when removing those, because then hyperlinks from outside the document wouldn't work.

*Flatten structures.* To facilitate incremental updates, PDF can store some structures (such as the page tree and the content streams within a page) spread to more objects and parts than necessary. Using the simplest, single-level or single-part structure saves space.

## 3 PDF size optimization tools

### 3.1 Test PDF files

In order to compare the optimization effectiveness of the tools presented in this section, we have compiled a set of test PDF files, and optimized them with each tool. The *totals* column of Table 4 shows the size of each file (the + and − percentages can be ignored for now), and other columns show the bytes used by different object types. The test files can be downloaded from [36]. Some more details about the test files:

**cff** 62-page technical documentation about the CFF file format. Font data is a mixture of Type 1, CFF and TrueType. Compiled with FrameMaker 7.0, PDF generated by Distiller 6.0.1.

**beamer1** 75 slide-steps long presentation created with

Table 4. PDF size reduction by object type, when running pdfsizeopy.py + Multivalent.

| *document* | *contents* | *font* | *image* | *other* | *xref* | *total* |
|---|---:|---:|---:|---:|---:|---:|
| cff | 141153 – 02% | 25547 – 02% | 0 | 178926 – 91% | 174774 – 100% | 521909 – 65% |
| beamer | 169789 – 03% | 44799 – 54% | 115160 – 00% | 445732 – 96% | 56752 – 98% | 832319 – 62% |
| eu2006 | 1065864 – 01% | 3271206 – 91% | 3597779 – 06% | 430352 – 80% | 45792 – 94% | 8411464 – 43% |
| inkscape | 10679156 – 20% | 230241 – 00% | 6255203 – 20% | 943269 – 79% | 122274 – 94% | 18245172 – 24% |
| lme2006 | 1501584 – 14% | 314265 – 73% | 678549 – 06% | 176666 – 91% | 31892 – 93% | 2703119 – 25% |
| pdfref | 6269878 – 05% | 274231 – 04% | 1339264 – 00% | 17906915 – 79% | 6665536 – 100% | 32472771 – 65% |
| pgf2 | 2184323 – 03% | 275768 – 51% | 0 | 1132100 – 84% | 190832 – 96% | 3783193 – 36% |
| texbook | 1507901 – 01% | 519550 – 48% | 0 | 217616 – 84% | 35532 – 87% | 2280769 – 21% |
| tuzv | 112145 – 03% | 201155 – 84% | 0 | 21913 – 77% | 2471 – 88% | 337764 – 57% |

The first number in each cell is the number of bytes used in the original document.
The –...% value indicates the percentage saved by optimization.
The data in this table was extracted from the original and optimized PDF files using `pdfsizeopy.py --stats`.

*contents:* content streams
*font:* embedded font files
*image:* pixel-based image objects and inline images, the latter created by sam2p
*other:* other objects
*xref:* cross-reference tables or streams
*total:* size of the PDF file

beamer.cls [40]. Contains hyperlinks, math formulas, some vector graphics and a few pixel-based images. Compiled with pdfTEX. Font data is in Type 1 format.

**eu2006** 126-page conference proceedings (of EuroTEX 2006) containing some large images. Individual articles were compiled with pdfTEX, and then PDF files were concatenated. Because of the concatenation, many font subsets were embedded multiple times, so a large part of the file is font data. Font data is mostly CFF, but it contains some Type 1 and TrueType fonts as well. Most fonts are compressed with the less effective LZW instead of ZIP.

**inkscape** 341-page software manual created with codeMantra Universal PDF [5]. Contains lots of screenshots and small images. Font data is a mixture of Type 1, CFF and TrueType.

**lme2006** 240-page conference proceedings in Hungarian. Contains some black-and-white screenshot images. Individual articles were compiled with LATEX and dvips (without font subsetting), and the PostScript files were concatenated and converted to PDF in a single run of a modified ps2pdf. Since font subsetting was disabled in dvips, later ps2pdf was able to subset fonts without duplication. Font data is in CFF.

**pdfref** 1310-page reference manual about PDF 1.7 containing quite a lot of duplicate `xref` tables and XML metadata of document parts. Optimization gets rid of both the duplicate `xref` tables and the XML metadata. Font data is in CFF. Compiled with FrameMaker 7.2, PDF generated by Acrobat Distiller 7.0.5.

**pgf2** 560-page software manual about Ti*k*Z, with lots of vector graphics as examples, with an outline, without hyperlinks. Compiled with pdfTEX. Font data is in Type 1 format.

**texbook** 494-page user manual about TEX (*The TEXbook*), compiled with pdfTEX. No pixel images, and hardly any vector graphics.

**tuzv** Mini novel in Hungarian, typeset on 20 A4 pages in a 2-column layout. Generated by dvipdfm. It contains no images or graphics. Font data is in Type 1 format.

None of the test PDF files used object streams or cross-reference streams.

## 3.2 ps2pdf

The ps2pdf [28] script (and its counterparts for specific PDF versions, e.g. ps2pdf14) runs Ghostscript with the flag `-sDEVICE=pdfwrite`, which converts its input to PDF. Contrary to what the name suggests, ps2pdf accepts not only PostScript, but also PDF files as input.

ps2pdf works by converting its input to low-level PostScript drawing primitives, and then emitting them as a PDF document. ps2pdf wasn't written to be a PDF size optimizer, but it can be used as such. Table 5 shows that ps2pdf increases the file size many times. For the documents *cff* and *pdfref*, we got a file size decrease because ps2pdf got rid of some metadata, and for *pdfref*, it optimized the cross-reference table. For *eu2006* it saved space by recompressing fonts with ZIP. The document *tuzv* became smaller because ps2pdf converted Type 1 fonts to CFF. The reason for the extremely large growth in *beamer1* is that ps2pdf blew up images, and it also embedded multiple instances of the same image as separate images. (It doesn't always do so: if the two instances of the image are close to each other, then ps2pdf reuses the same object in the PDF for representing the image.)

ps2pdf keeps all printable features of the original PDF, and hyperlinks and the document outline as well. However, it recompresses JPEG images (back to a different JPEG, sometimes larger than the original), thus losing quality. The only way to disable this is specifying the flags `-dEncodeColorImages=false` `-dEncodeGrayImages=false`, but it would blow up the file size even more, because it will keep photos uncompressed. Specifying `-dColorImageFilter=/FlateEncode` would convert JPEG images to use ZIP compression without quality loss, but this still blows up the file size. Thus, it is not possible to set up pdf2ps to leave JPEG images as is: it will either blow up the image size (by uncompressing the image or recompressing it with ZIP), or it will do a transformation with quality loss. The Distiller option `/PassThroughJPEGImages` would solve this problem, but Ghostscript doesn't support it yet.

ps2pdf doesn't remove duplicate content (although it removes image duplicates if they are close by), and it also doesn't minimize the use of indirect references (e.g. it emits the `/Length` of content streams as an indirect reference). The only aspects ps2pdf seems to optimize effectively is converting Type 1 fonts to CFF and removing content outside the page. Since this conversion is also done by pdfsizeopt.py, it is not recommended to use ps2pdf to optimize PDF files.

Table 5. PDF optimization effectiveness of ps2pdf.

| *document* | *input* | *ps2pdf* | *psom* |
|---|---:|---:|---:|
| cff | 521909 | 264861 | 180987 |
| beamer1 | 832319 | *3027368* | 317351 |
| eu2006 | 8411464 | 6322867 | 4812306 |
| inkscape | 18245172 | failed | 13944481 |
| lme2006 | 2703119 | *3091842* | 2033582 |
| pdfref | 32472771 | 15949169 | 11237663 |
| pgf2 | 3783193 | *4023581* | 2438261 |
| texbook | 2280769 | *2539424* | 1806887 |
| tuzv | 337764 | 199279 | 146414 |

All numeric values are in bytes. Italic values indicate that the optimizer increased the file size.
*ps2pdf:* Ghostscript 8.61 run as
`ps2pdf14 -dPDFSETTINGS=/prepress`
*psom:* pdfsizeopt.py + Multivalent

## 3.3 PDF Enhancer

PDF Enhancer [20] is commercial software which can concatenate, split, convert and optimize PDF documents, and remove selected PDF parts as well. It has lots of conversion and optimization features (see the table in [4]), and it is highly configurable. With its default settings, it optimizes the PDF without removing information. It is a feature-extended version of the PDF Shrink software from the same company. The use of the GUI version of PDF Enhancer is analyzed in [12]. A single license for the *server edition*, needed for batch processing, costs about $1000, and the *advanced server edition* (with JBIG2 support) costs about twice as much. The *standard edition* with the GUI costs only $200.

Columns *input* and *pdfe* of Table 6 show how effectively PDF Enhancer optimizes. The *server edition* was used in our automated tests, but the *standard edition* generates PDF files of the same size. Looking at columns *pdfe* and *a9p4* we can compare PDF Enhancer to Adobe Acrobat Pro. Please note that PDF Enhancer doesn't generate object streams or cross-reference streams, that's why we compare it to *a9p4* instead of *a9p5* in the table. Feeding the output of PDF Enhancer to Multivalent decreases the file size even further, because Multivalent generates those streams. The column *epsom* of Table 6 shows the PDF output file sizes of the PDF Enhancer + pdfsizeopt.py + Multivalent combination, which seems to be the most effective for TEX documents.

According to the messages it prints, PDF Enchancer optimizes content streams within the page. Most other optimizers (except for Adobe Acrobat Pro) don't do this. Text-only content streams generated from TEX don't benefit from such an optimization, but for the *pgf2* document, which contains lots of graphics, this optimization saved about 10% of the content streams.

It is worth noting that PDF Enhancer failed when optimizing one of the test documents (see in Table 6). The developers of PDF Enhancer reply quickly to bug reports, and they are willing to track and fix bugs in the software.

## 3.4 Adobe Acrobat Pro

Adobe's WYSIWYG PDF manipulation program, Adobe Acrobat Pro [1] also contains a PDF optimizer (menu item *Advanced / PDF Optimizer*). A single license of the whole software costs $450; it is not possible to buy only the optimizer. There seems to be no direct way to run the optimizer on multiple files in batch mode. Columns *a9p4* and *a9p5* of Table 6 shows the effectiveness of the optimizer: values in the column *a9p4* are for PDF 1.4 output, and column *a9p5* belongs to PDF 1.5 output. The PDF 1.5 files are much smaller because they make use of object streams and cross-reference streams. The optimizer lets

Table 6. PDF optimization effectiveness of PDF Enhancer and Adobe Acrobat Pro.

| *document* | *input* | *pdfe* | *epsom* | *psom* | *apsom* | *a9p4* | *a9p5* |
|---|---:|---:|---:|---:|---:|---:|---:|
| cff | 521909 | 229953 | 174182 | 180987 | 158395 | 548181 | 329315 |
| beamer1 | 832319 | 756971 | 296816 | 317351 | 317326 | 765785 | 363963 |
| eu2006 | 8411464 | failed | n/a | 4812306 | 3666315 | 8115676 | 7991997 |
| inkscape | 18245172 | 14613044 | 12289136 | 13944481 | 11807680 | 14283567 | 13962583 |
| lme2006 | 2703119 | 2263227 | 1781574 | 2033582 | 1830936 | 2410603 | 2279985 |
| pdfref | 32472771 | 23794114 | 11009960 | 11237663 | 9360794 | 23217668 | 20208419 |
| pgf2 | 3783193 | 3498756 | 2245797 | 2438261 | n/a | failed | failed |
| texbook | 2280769 | 2273410 | 1803166 | 1806887 | 1804565 | 2314025 | 2150899 |
| tuzv | 337764 | *338316* | *147453* | 146414 | *150813* | *344215* | 328843 |

All numeric values are in bytes. Italic values indicate that the optimizer increased the file size.
*pdfe:* PDF Enhancer 3.2.5 (1122r) server edition
*epsom:* PDF Enhancer + pdfsizeopt.py + Multivalent
*psom:* pdfsizeopt.py + Multivalent
*apsom:* Adobe Acrobat Pro 9 creating PDF 1.4 + pdfsizeopt.py + Multivalent
*a9p4:* Adobe Acrobat Pro 9 creating PDF 1.4
*a9p5:* Adobe Acrobat Pro 9 creating PDF 1.5

the user specify quite a few settings. For the tests we have enabled all optimizations except those which lose information (such as image resampling). It turned out that we had to disable *Discard User Data / Discard all comments, forms and multimedia*, otherwise the optimizer removed hyperlinks from the document *beamer1*.

It is worth noting that Adobe Acrobat Pro 9 failed with an image-related error when optimizing document *pgf2*. Oddly enough, that PDF file doesn't contain any images.

### 3.5 pdfcompress

pdfcompress [45] is the command-line version of the PDF optimizer in Advanced PDF Tools. It is commercial software, a single-computer license costs less than $80. It can resample and recompress images based on a few set of settings for monochrome, gray and color images. It can also recompress streams, and it can remove some PDF features (such metadata, JavaScript, page thumbnails, comments, embedded files, outlines, private data and forms). We haven't analyzed it, because PDF Enhancer seems to have all the features of pdfcompress.

### 3.6 Multivalent tool.pdf.Compress

Multivalent [17] is a collection of programs for document viewing, annotation, organization, conversion, validation, inspection, encryption and text extraction (etc.). It supports multiple file formats such as HTML, PDF, DVI and man pages. It is implemented in Java; the 2006 January version is available for download [18] as a single .jar file, and it needs Java 1.4 or later. It contains a PDF optimizer [24, 27], which can be invoked like this at the command-line (without line breaks):

```
java -cp Multivalent20060102.jar
    tool.pdf.Compress doc.pdf
```

Table 7: PDF optimization effectiveness of Multivalent and pdfsizeopt.py.

| *document* | *input* | *multi* | *psom* | *pso* |
|---|---:|---:|---:|---:|
| cff | 521909 | 181178 | 180987 | 230675 |
| beamer1 | 832319 | 341732 | 317351 | 443253 |
| eu2006 | 8411464 | 7198149 | 4812306 | 4993913 |
| inkscape | 18245172 | 13976597 | 13944481 | 17183194 |
| lme2006 | 2703119 | 2285956 | 2033582 | 2349035 |
| pdfref | 32472771 | 11235006 | *11237663* | 23413875 |
| pgf2 | 3783193 | 2584180 | 2438261 | 3449386 |
| texbook | 2280769 | 2057755 | 1806887 | 1992958 |
| tuzv | 337764 | 314508 | 146414 | 166863 |

All numeric values are in bytes. The Italic value indicates that Multivalent alone was better than with pdfsizeopt.py.
*multi:* Multivalent 20060102 *tool.pdf.Compress*
*psom:* pdfsizeopt.py + Multivalent
*pso:* pdfsizeopt.py without Multivalent

This creates the optimized PDF in file doc-o.pdf. If we don't indicate otherwise, by the term Multivalent we mean its PDF optimizer. Although the 2006 January version of Multivalent with full functionality is available for download, Multivalent is not free software or open source. For example, its license allows running the PDF optimizer from the command-line. For other uses of the optimizer, a commercial license has to be acquired. The web site doesn't show details about commercial licenses.

According to [27], the Mutivalent did the following optimizations in 2003: remove object duplicates; recompress LZW to ZIP; generate object streams; generate a cross-reference stream; serialize values more effectively; remove old object versions; remove page thumbnails;

remove some obsolete values such as /ProcSet; inline small objects such as stream lengths; remove unused objects; omit default values; shrink cross-reference data. In addition to those above, Multivalent recompresses all streams with maximum-effort ZIP, and it also moves up /MediaBox etc. in the page tree.

Column *multi* of Table 7 shows how effectively Multivalent optimizes. The column *psom* indicates that running pdfsizeopt.py before Multivalent usually decreases the file size even more. That's because pdfsizeopt.py can convert Type 1 fonts to CFF, unify CFF font subsets, and it also has a more effective image optimizer than Multivalent.

## 3.7 pdfsizeopt.py

pdfsizeopt.py [37] was written as part of this work. Its purpose is to implement the most common optimizations typical TEX documents benefit from, but only those which are not already done by Multivalent. As described in Section 4, to get the smallest PDF, the optimizations done by pdfsizeopt.py should be applied first, and the result should be processed by Multivalent. The 20060102 version of Multivalent optimizes images, and it replaces the image even if the optimized version is larger than the original, so pdfsizeopt.py implements a final step to put those original images back which are smaller.

pdfsizeopt.py can be used as a stand-alone PDF optimizer (without Multivalent), but the final PDF will be much smaller if Multivalent is run as well.

pdfsizeopt.py is free software licensed under the GPL. It is written in Python. It needs Python 2.4 (or 2.5 or 2.6). It uses only the standard Python modules, but it invokes several external programs to help with the optimizations. These are: Ghostscript (8.61 or newer is recommended), sam2p [38] (0.46 is needed), pngtopnm, tool.pdf.Compress of Multivalent [24] (which needs Sun's JDK or OpenJDK), optionally jbig2 [14], optionally PNGOUT [31]. Installation instructions are given in [35]. Most of these are free software, except for the Multivalent tools, which are not free software or open source, but they can be downloaded and used on the command line free of charge; for other uses they have to be licensed commercially. PNGOUT is not free software or open source either, but the binaries available free of charge can be used without restriction.

pdfsizeopt.py implements these PDF size optimization methods:

**Convert Type 1 fonts to CFF** It is done by generating a PostScript document with all fonts, converting it to PDF with Ghostscript (just like ps2pdf), and extracting the CFF fonts from the PDF. Another option would be to use dvipdfmx, which can read Type 1 fonts, and emit them as CFF fonts. Please note that Ghostscript inlines subroutines (/Subrs) in the Type 1 font, so the CFF becomes larger—but we are compressing the font with ZIP anyway, which eliminates most of the repetitions.

**Unify subsets of the same CFF font** Ghostscript is used for parsing CFF to a font dictionary, and also for serializing the modified dictionary as CFF. Again, the latter is done by generating a PostScript file with all the fonts, then converting it to a PDF using Ghostscript. Limitations: it only works for CFF (and former Type 1) fonts; it doesn't unify fonts with different names; it won't unify some fonts if one of them has slightly different metrics.

**Convert inline images to objects** We need this because most tools (including pdfsizeopy.py) do not optimize inline images. Limitations: it only detects inline images generated by sam2p; it only detects inline images within a form XObject (not in a content stream).

**Optimize individual images** First the data gets decompressed (with Ghostscript if the image data is compressed with anything other than simple ZIP), then it is recompressed with high-effort ZIP, then it is converted to PNG, then several external PNG compressors are run to get the optimized PNG, and finally the smallest representation (among the optimized PNG files, intermediate images and the original image) is picked, i.e. the one with the smallest PDF image object representation, counting the stream dictionary and the compressed stream as well. The following PNG optimizers are used: sam2p without predictor, sam2p with PNG predictor, PNGOUT (very slow, but generates a few percent smaller PNG files) and jbig2 (only for bilevel images). Limitations: no CMYK support; no device-independent color space support (only RGB with or without palette and grayscale is supported); no images with an alpha channel; only some types of transparency; images with lossy compression (JPEG or JPEG2000) are not optimized.

**Remove object duplicates** Equivalence classes are used, so duplicate subtrees referring to objects between themselves or each other are also removed. (Multivalent also has this feature.)

**Remove image duplicates** Images are compared based on RGB pixel data, so duplicates using a different compression or color space or bits-per-component are also detected and removed. This is useful if the PDF is a concatenation of PDF files in the same collection, each PDF compiled with a different method, and then concatenated. The newest version of sam2p (0.46) produces exactly the same output file for two images with identical RGB pixel data, so image duplicates are identified by comparing the files created by sam2p. There are also several early checks in the optimization algorithm to detect the duplicate before wasting time on running the many different optimizers.

**Remove unused objects** All objects unreachable from the trailer object are removed.

**Serialize values more effectively** Extra spaces are removed; hex strings are converted to binary; strings are serialized without extra backslashes; comments are removed; garbage between object definitions is removed; gaps in the cross-reference table are removed; objects with high reference counts are given low numbers.

The column *pso* of Table 7 shows how effectively pdfsizeopt.py optimizes. The column *psom* shows the combined effectiveness of pdfsizeopt.py + Multivalent. Please note that it is not with running pdfsizeopt.py alone, because pdfsizeopt.py was designed to do only those optimizations which Multivalent does not provide (or, such as image compression, does suboptimally). On the other hand, it is almost always worth running pdfsizeopt.py before Multivalent, rather than running Multivalent alone. The only exception we could find was the document *pdfref*, where the combined approach yielded a 0.02% larger file size.

pdfsizeopt.py can count the total byte size of various object types in a PDF. Table 4 shows the results on our test PDF files. The percentages in the table cells are savings by running pdfsizeopt.py + Multivalent. Although it is not visible in the table, most of the savings come from Multivalent, except in the *font* and *image* columns, where the contributions of pdfsizeopt.py are important. The large font savings for the document *tuzv* are because the document is short and it contains many Type 1 fonts. For the document *eu2006* we get an even larger saving, because there was lots of glyph duplication across the articles in the collection, and also because LZW was used instead of ZIP to compress the fonts. Only a few of our test documents benefit from image optimization, and even there the contribution of pdfsizeopt.py is small because the original PDF contains the images emitted effectively, and also Multivalent does a decent (though suboptimal) job at image optimization. So for the document *eu2006* Multivalent alone saves about 1.55%, and *pdfsizeopt.py* alone saves 6.14%. (There is no data on the extra size reduction by combining the two tools, because pdfsizeopt.py disables Multivalent's image optimizations since most images won't benefit.) For the document *lme2006* Multivalent alone saves 3.41%, and pdfsizeopy.py alone saves 6.39%. The document *inkscape* benefits most from image recompression: Multivalent alone saves 19.87%, and pdfsizeopy.py alone saves 20.35%.

Columns *psom*, *apsom* and *epsom* of Table 6 show that optimizing with PDF Enhancer or Adobe Acrobat Pro before running the pdfsizeopt.py + Multivalent combination almost always decreases the file size, sometimes by a few percent, but in the case of document *beamer1* the extra gain of running PDF Enhancer first was 6.46%. It seems that for TEX documents PDF Enhancer (with the combination) is the more effective, and Adobe Acrobat Pro is more effective for other documents.

See ideas for improving pdfsizeopt.py in Section 6.

## 4 Suggested PDF optimization workflow

Based on the optimization tests in Section 3 we suggest the following PDF generation and optimization workflow:

1. Upgrade Ghostscript to at least 8.61, and upgrade to TEX Live 2008.
2. For TEX documents, create the PDF using pdfTEX or dvipdfmx, with the settings discussed in Subsection 2.1. Use dvips + ps2pdf only if absolutely necessary, because of the large PDF files it produces.
3. Use pdftk or Multivalent's PDF merge tool (as shown in [32]) to concatenate PDF files if necessary. Pay attention to the hyperlinks and the document outline after concatenation. Don't concatenate with Ghostscript, because that it would blow up the file size.
4. If you have access to PDF Enhancer, optimize the PDF with it. Otherwise, if you have access to Adobe Acrobat Pro, optimize the PDF with it.
5. Optimize the PDF with pdfsizeopt.py, including the last step of running Multivalent as well.

Most of the optimization steps above can be fully automated and run in batch, except if Adobe Acrobat Pro is involved.

## 5 Related work

There are several documents discussing PDF optimization. [23] gives a list of ideas how to generate small PDF files. Most of those are present in this work as well. PDF Enhancer and Adobe Acrobat Pro are analyzed in [12], but that article focuses on reducing image resolution and unembedding fonts, which are not information-preserving optimizations. [44] gives a simple introduction to (possibly lossy) PDF image compression and content stream compression.

Since web browsers can display PNG images, several PNG optimization tools [15, 43, 31] have been developed to reduce web page loading times. These tools can be used for optimizing (mainly non-photo) images in PDF documents as well. But since PDF has a more generic image and compression model than PNG, it would be possible to save a little bit more by developing PDF-specific tools, which take advantage of e.g. using the TIFF predictor and ZIP compression together.

An alternative document file format is DjVu [6], whose most important limitation compared to PDF is that it doesn't support vector graphics. Due to the sophisticated image layer separation and compression, the size of a 600 DPI DjVu file is comparable to the corresponding optimized PDF document: if the PDF contains text with embedded vector fonts and vector graphics, the DjVu file can be about 3 times larger than the PDF. If the PDF contains mainly images (such as a sequence of scanned sheets), the DjVu file will become slightly smaller than the PDF. Of course these ratios depend on the software used for encoding as well. There are only a few DjVu encoders available: pdf2djvu and djvudigital are free, and Document Express is a commercial application. PDF is more complex than DjVu: the PDF 1.7 reference [3] itself is 1310 pages long, and it relies on external specifications such as ZIP, JBIG2, G3 fax, JPEG, JPEG2000, Type 1, CFF, TrueType, OpenType, CMap, CID font, XML, OPI, DSA, AES, MD5, SHA-1, PKCS, PANOSE, ICC color profiles, JavaScript and more. PDF 1.7 became an ISO standard [11] in 2008, which adds additional long documents. Having to understand many of these makes PDF viewers hard to implement and complex. This problem can become more severe for long-term archiving if we want to view a PDF 20 or 50 years from now; maybe today's PDF viewers won't work on future architectures, so we have to implement our own viewer. In contrast, the DjVu specification [16] is only 71 pages long, and more self-contained. Since the DjVu file format uses very different technologies than PDF, one can archive both the PDF and the DjVu version of the same document, in case a decent renderer won't be available for one of the formats decades later.

The PDF Database [19] contains more than 500 PDF documents by various producers, with different sizes and versions. These PDF files can be used can be used for testing PDF parsers and optimizers.

Multivalent introduced the custom file format *compact PDF* [25, 27], which is about 30% to 60% smaller than a regular PDF. The disadvantage is that only Multivalent can read or write this format so far (but it supports fast and lossless conversion to regular PDF). Compact PDF achieves the size reduction by grouping similar objects next to each other, and compressing the whole document as one big stream with bzip2, which is superior to ZIP. Another improvement is that compact PDF stores Type 1 fonts unencrypted, with boilerplate such as the 512-byte font tailer and random bytes for encryption stripped out.

## 6 Conclusion and future work

Since it is not the primary goal for most PDF generators to emit the smallest possible PDF, simple techniques done by Multivalent and pdfsizeopt.py can yield significant size reduction (up to a factor of 3) depending on the generator and the PDF features used. Rearranging the drawing instructions (contents streams and form XObjects, as done by Adobe Acrobat Pro and PDF Enhancer) is a more complicated optimization, and saves some more space in addition to the simple techniques. It also matters how the PDF was generated (e.g. pdfTEX generates a smaller and more optimizable PDF than dvips + ps2pdf).

The workflow proposed in this article has too many dependencies. Python (for pdfsizeopt.py) and Java (for Multivalent) runtimes, and Ghostscript (needed by pdfsizeopt.py for Type 1 and CFF font parsing, CFF generation and arbitrary stream filtering) are the heaviest ones. It is possible to get rid of these by reimplementing pdfsizeopt.py from scratch. To get rid of Python, we could use Lua, and build a statically linked C binary with the Lua interpreter, zlib and all the Lua bytecode linked in. We could reimplement the optimizations done by Multivalent in Lua. (This would include reading and writing object streams and cross-reference streams.) Gradually we could move some functionality to C or C++ code to speed up the optimizer. We could reuse the xpdf codebase to be able to use all PDF filters without invoking Ghostscript. We would have to implement Type 1 and CFF parsing and CFF generation, possibly relying on the dvipdfmx codebase. Other dependencies such as jbig2, sam2p, pngtopnm, PNGOUT and PDF Enhancer are not so problematic, because they can be compiled to small, statically linked, stand-alone executables.

Some optimizations of pdfsizeopt.py could be generalized to cover more cases. Examples are: add CMYK image optimization; make CFF matching more permissive (before unification); recognize more inline images (not only those created by sam2p, and not only in form XObjects). pdfsizeopt.py would also benefit from compiling a test set of PDF files (possibly based on the PDF Database [19]), and adding a framework which automatically checks that pdfsizeopt.py detected the opportunity to optimize, and did the optimization properly in each case.

When preparing a collection (such as a journal volume or a conference proceedings) with TEX, in a typical workflow individual articles are compiled to PDF, and the PDF files are then concatenated. See [32] for tools which can do PDF concatenation. The concatenated document can be optimized using pdfsizeopt.py + Multivalent to get rid of redundancy (such as duplicate glyphs in fonts and duplicate images) across articles. Not all concatenators can preserve hyperlinks and the document outline for TEX documents. Adding concatenation support to pdfsizeopt.py would make creating small and interactive collections more straightforward.

## References

[1] Adobe. Adobe Acrobat Pro 9 (project page). http://www.adobe.com/products/acrobatpro/.

[2] Adobe. *The Compact Font Format Specification*, 1.0 edition, 4 December 2003. http://www.adobe.com/devnet/font/pdfs/5176.CFF.pdf.

[3] Adobe. *PDF Reference, Adobe Portable Document Format Version 1.7.* Adobe, 6th edition, November 2006. http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf.

[4] Apago. Which features are in what PDF Enhancer edition?, 29 July 2009. http://www.apagoinc.com/prod_feat.php?feat_id=30&feat_disp_order=7&prod_id=2.

[5] codeMantra Universal PDF, a PDF generator. http://codemantra.com/universalpdf.htm.

[6] DjVu: A tutorial. http://www.djvuzone.org/support/tutorial/chapter-intro.html.

[7] DVIPDFMx, an extended DVI-to-PDF translator. http://project.ktug.or.kr/dvipdfmx/.

[8] Till Tantau ed. *The TikZ and PGF Packages.* Institute für Theoretische Informatik, Universität zu Lübeck, 2.00 edition, 20 February 2008. http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf.

[9] Jonathon Fowler. PNGOUT port for Unix systems, 2007. http://www.jonof.id.au/kenutils.

[10] Gimp, the GNU Image Manipulation Program. http://www.gimp.org/.

[11] ISO 32000-1:2008 Document management—Portable document format—part 1: PDF 1.7, 2008. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51502.

[12] Andy King. Optimize PDF files. http://websiteoptimization.com/speed/tweak/pdf/, 25 September 2006.

[13] Ralf Koening. Creative use of PDF files in LATEX environments. http://www.tu-chemnitz.de/urz/anwendungen/tex/stammtisch/chronik/pdf_interna.pdf, 18 June 2004.

[14] Adam Langley. jbig2enc, a JBIG2 encoder (project page). http://github.com/agl/jbig2enc/tree/master.

[15] List of PNG recompressors. http://en.wikipedia.org/wiki/OptiPNG#See_also.

[16] Lizardtech. *DjVu Reference*, djVu v3 edition, November 2005. http://djvu.org/docs/DjVu3Spec.djvu.

[17] Multivalent, digital documents research and development. http://multivalent.sourceforge.net/.

[18] Multivalent, download location. http://sourceforge.net/projects/multivalent/files/.

[19] PDF database, 20 April 2005. http://www.stillhq.com/pdfdb/db.html.

[20] PDF Enhancer, a PDF converter, concatenator and optimizer. http://www.apagoinc.com/prod_home.php?prod_id=2.

[21] Workarounds for PDF output with the PSTricks LATEX package. http://tug.org/PSTricks/main.cgi?file=pdf/pdfoutput.

[22] PDFCreator, a free tool to create PDF files from nearly any Windows application. http://www.pdfforge.org/products/pdfcreator.

[23] Shlomo Perets. Best practices #1: Reducing the size of your PDFs, 7 August 2001. http://www.planetpdf.com/creative/article.asp?ContentID=6568.

[24] Thomas A. Phelps. Compress, the Multivalent PDF compression tool. http://multivalent.sourceforge.net/Tools/pdf/Compress.html.

[25] Thomas A. Phelps. Compact PDF specification, March 2004. http://multivalent.sourceforge.net/Research/CompactPDF.html.

[26] Thomas A. Phelps and P.B. Watry. A no-compromises architecture for digital document preservation. In *Proceedings of European Conference on Digital Libraries*, September 2005. http://multivalent.sourceforge.net/Research/Live.pdf.

[27] Thomas A. Phelps and Robert Wilensky. Two diet plans for fat PDF. In *Proceedings of ACM Symposium on Document Engineering*, November 2003. http://multivalent.sourceforge.net/Research/TwoDietPlans.pdf.

[28] ps2pdf, a PostScript-to-PDF converter. http://pages.cs.wisc.edu/~ghost/doc/svn/Ps2pdf.htm.

[29] Tomas Rokicki. Dvips: A DVI-to-PostScript translator, January 2007. http://mirror.ctan.

org/info/doc-k/dvips.pdf.

[30] Ken Silverman. KZIP, a PKZIP-compatible compressor focusing on space over speed. http://advsys.net/ken/utils.htm#kzip.

[31] Ken Silverman. PNGOUT, a lossless PNG size optimizer, 2009. http://advsys.net/ken/utils.htm#pngout.

[32] Matthew Skala. How to concatenate PDFs without pain, 13 May 2008. http://ansuz.sooke.bc.ca/software/pdf-append.php.

[33] Matthias Stirner and Gerhard Seelmann. packJPG, a lossless compressor for JPEG images (project page), 21 November 2007. http://www.elektronik.htw-aalen.de/packjpg/.

[34] Péter Szabó. Extra files related to PDF generation and PDF size optimization. http://code.google.com/p/pdfsizeopt/source/browse/#svn/trunk/extra.

[35] Péter Szabó. Installation instructions for pdfsizeopt.py. http://code.google.com/p/pdfsizeopt/wiki/InstallationInstructions.

[36] Péter Szabó. PDF test files for pdfsizeopt.py. http://code.google.com/p/pdfsizeopt/wiki/ExamplePDFsToOptimize.

[37] Péter Szabó. pdfsizeopt.py, a PDF file size optimizer (project page). http://code.google.com/p/pdfsizeopt.

[38] Péter Szabó. sam2p, a pixel image converter which can generate small PostScript and PDF. http://www.inf.bme.hu/~pts/sam2p/.

[39] Péter Szabó. Inserting figures into TEX documents. In *EuroBachoTEX*, 2002. http://www.inf.bme.hu/~pts/sam2p/sam2p_article.pdf.

[40] Till Tantau. *The beamer class*, 3.07 edition, 11 March 2007. http://www.ctan.org/tex-archive/macros/latex/contrib/beamer/doc/beameruserguide.pdf.

[41] Hàn Thế Thành, Sebastian Rahtz, Hans Hagen, et al. *The pdfTEX manual*, 1.671 edition, 25 January 2007. http://www.ctan.org/get/systems/pdftex/pdftex-a.pdf.

[42] Cosmin Truţa. A guide to PNG optimization. http://optipng.sourceforge.net/pngtech/optipng.html, 2008.

[43] Cosmin Truţa. OptiPNG, advanced PNG optimizer (project page), 9 June 2009. http://optipng.sourceforge.net/.

[44] VeryPDF.com. Compressing your PDF files, 13 July 2006. http://www.verypdf.com/pdfinfoeditor/compression.htm.

[45] VeryPDF.com. *PDF Compress Command Line User Manual*, 13 July 2006. http://www.verypdf.com/pdfinfoeditor/pdfcompress.htm.

[46] Pauli Virtanen. TexText, an Inkscape extension for adding LATEX markup, 6 February 2009. http://www.elisanet.fi/ptvirtan/software/textext/.

Péter Szabó
Google
Brandschenkestrasse 110
CH-8002, Zürich, Switzerland
pts (at) google dot com
http://www.inf.bme.hu/~pts/

# Generating PDF for e-reader devices

**Abstract**
NotuDoc is a commercial Internet application that uses ConTEXt for the on-the-fly generation of PDF documents for, amongst other things, the e-reader devices of iRex technologies. This articles offers a glimpse behind the scenes.

## Introduction

This article is about the generation of PDF documents for e-reader devices. Before we delve into that, let's look at the controlling application (NotuDoc) that this process is a part of, and do a short introduction of the e-reader devices that are used.

Generating PDF from within NotuDoc is a fairly small part of the application, but even so there is a fair amount of complexity.

## NotuDoc

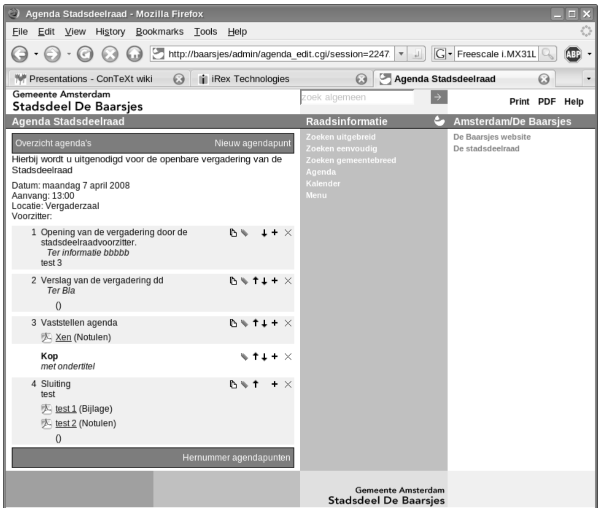
The Dutch company NotuBiz (http://www.notubiz.nl) focuses on everything related to the recording and publishing of (council) meeting reports using modern media. For example, NotuBiz takes care of live streaming and the publication of digital meeting reports on the Internet. The clients of NotuBiz are local government bodies in The Netherlands, but increasingly also in neighboring countries.

In practical use, it turned out that the information stream *leading up to* the actual meetings was far from optimal. NotuDoc was born out of this realization: NotuDoc is an Internet application that links (preliminary or final) meeting agendas to the corresponding meeting documents like commission reports, presentations and quotations. Afterwards, the resulting combination is made available to the relevant meeting parties via the Internet and/or PDF document export functionality.

By gathering and combining all the necessary documents in one place, it becomes easier for the participants to prepare for the meeting. As a bonus, afterwards the official meeting report can be linked to already existing meeting data easily and therefore everything is set up for near-effortless publication to the members of the community (as is required by law).

NotuDoc is a plug-and-go commercial product with quite extensive configuration possibilities. This is important because the web interface has to integrate nicely with the layout of the client's website, but it goes further than that: not all clients have the same meeting structures and only a very few use the same internal document management system (DMS). NotuDoc comes with pre-installed support for the most commonly used systems in the Netherlands, and the extensibility ensures that support for other DMS-s is easily added.

The NotuDoc code is implemented and maintained by Elvenkind in close cooperation with NotuBiz and is based on Elvenkind's development framework, written using Perl 5.



Hierbij wordt u uitgenodigd voor de openbare vergadering van de Stadsdeelraad

Datum: maandag 7 april 2008
Aanvang: 13:00
Locatie: Vergaderzaal
Voorzitter:

1 Opening van de vergadering door de stadsdeelraadvoorzitter.
*Ter informatie bbbbb*
test 3

2 Verslag van de vergadering dd
*Ter Bla*
()

3 Vaststellen agenda
Xen (Notulen)

**Kop**
*met ondertitel*

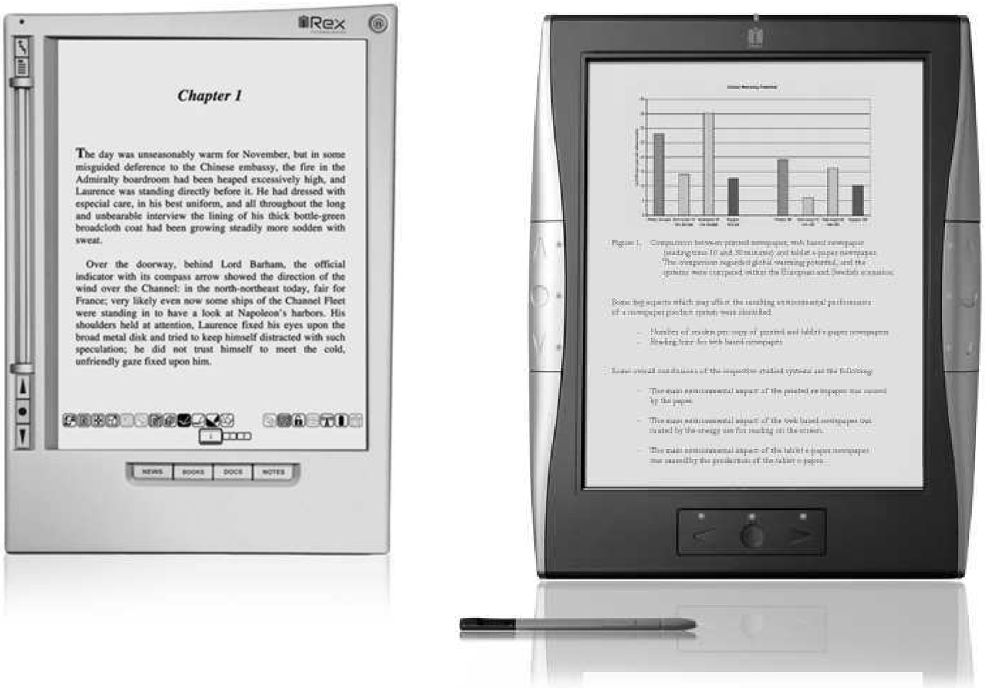
4 Sluiting
test
test 1 (Bijlage)
test 2 (Notulen)
()

Figure 1. An example main screen of the NotuDoc Internet application

## E-reader devices

At present, NotuDoc comes prepared for the generation of PDF documents for two specific e-reader devices, both developed by iRex Technologies (http://www.irextechnologies.com), a spin-off of Philips International. Besides these two specific devices (iLiad and DR1000) it is naturally also possible to generate PDF for printing and for interactive work on a computer / notebook.

Both the iLiad and the newer DR1000 are based on the same core technology. The iLiad has been around for a few years now and amongst other things it is the publishing platform for the digital version of 'NRC Handelsblad' (a Dutch newspaper comparable to the Wall Street Journal). The DR1000 has a new design and it



Chapter 1

The day was unseasonably warm for November, but in some misguided deference to the Chinese embassy, the fire in the Admiralty boardroom had been heaped excessively high, and Laurence was standing directly before it. He had dressed with especial care, in his best uniform, and all throughout the long and unbearable interview the lining of his thick bottle-green broadcloth coat had been growing steadily more sodden with sweat.

Over the doorway, behind Lord Barham, the official indicator with its compass arrow showed the direction of the wind over the Channel: in the north-northeast today, fair for France; very likely even now some ships of the Channel Fleet were standing in to have a look at Napoleon's harbors. His shoulders held at attention, Laurence fixed his eyes upon the broad metal disk and tried to keep himself distracted with such speculation; he did not trust himself to meet the cold, unfriendly gaze fixed upon him.

Figure 2. iLiad (left) and DR1000 (right).

offers a larger screen and somewhat faster hardware, but technologically there are very few differences between the two devices.

Both devices are based on 'digital paper', a technology whereby the displayed data stays visible without the need to refresh the screen many times a second.

The key advantage of this technology is that it uses far less power, resulting in much longer battery life when compared to traditional TFT or LCD screens. Another good thing is that because there is no need for a back light, actually looking at the screen is a lot easier on the eyes.

On the other hand, there are downsides to electronic paper. The two largest of these: the reaction time, which is much slower than for conventional computer displays, and the output is grayscale only. Hopefully, future developments will remove both limitations.

Both devices make use of 'Wacom Penabled' technology (http://wacom.com/tabletpc/what_is_penabled.cfm) that makes it possible to write and sketch directly on the display, so that you can create notes directly into the PDF. The companion software (for Windows) is able to merge these notes with the original PDF document into a new PDF document for later use.

The supported formats for both devices are the same as well: PDF, HTML, mobipocket, and a few bitmap image formats. All software used and developed by iReX is open source, based on a Linux distribution for embedded devices. Connection to the PC for exchanging documents is done via USB or optionally (for iLiad) using a wireless network. Both iLiad and DR1000 use removable memory cards as storage medium.

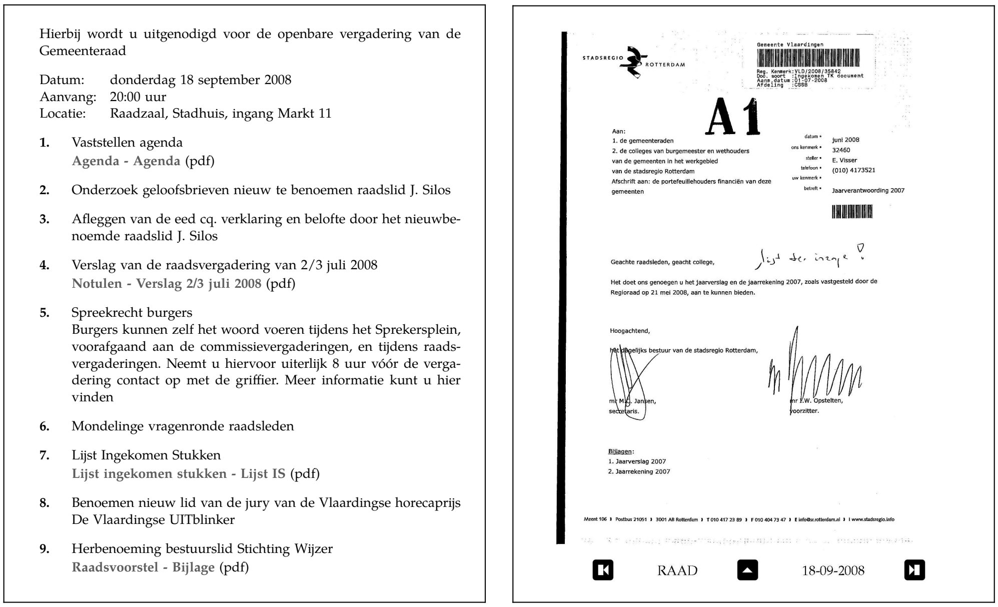
## PDF generation

The PDF generation in NotuDoc is handled by a Perl script that is completely template driven. It uses near-identical code both for the generation of TEX input files and for HTML pages. Only the character escape functions and filenames are adjusted specifically for TEX. Just like the web pages, the PDF documents are generated at runtime by calling `texexec`. The TEX subsystem uses ConTEXt supplied by the 'contextgarden' distribution (http://minimals.contextgarden.net/).

### ConTEXt templates

For each client, the application stores a setting for the desired PDF output type. The following example uses `iliad`, but it can also be something else like `dr1000` or just `a4`. Separate from this global preference, it is possible to define a client-specific layout that creates a document layout that corresponds to the desired house style.

Layout definitions are implemented via ConTEXt macros that are separated out from the Internet application. The application only takes care of converting the database records of a meeting agenda into a ConTEXt input source and exporting the required meeting doc-



Hierbij wordt u uitgenodigd voor de openbare vergadering van de Gemeenteraad

Datum: donderdag 18 september 2008
Aanvang: 20:00 uur
Locatie: Raadzaal, Stadhuis, ingang Markt 11

1. Vaststellen agenda
   Agenda - Agenda (pdf)
2. Onderzoek geloofsbrieven nieuw te benoemen raadslid J. Silos
3. Afleggen van de eed cq. verklaring en belofte door het nieuwbenoemde raadslid J. Silos
4. Verslag van de raadsvergadering van 2/3 juli 2008
   Notulen - Verslag 2/3 juli 2008 (pdf)
5. Spreekrecht burgers
   Burgers kunnen zelf het woord voeren tijdens het Sprekersplein, voorafgaand aan de commissievergaderingen, en tijdens raadsvergaderingen. Neemt u hiervoor uiterlijk 8 uur vóór de vergadering contact op met de griffier. Meer informatie kunt u hier vinden
6. Mondelinge vragenronde raadsleden
7. Lijst Ingekomen Stukken
   Lijst ingekomen stukken - Lijst IS (pdf)
8. Benoemen nieuw lid van de jury van de Vlaardingse horecaprijs De Vlaardingse UITblinker
9. Herbenoeming bestuurslid Stichting Wijzer
   Raadsvoorstel - Bijlage (pdf)

STADSREGIO ROTTERDAM

Gemeente Vlaardingen
Reg. Kenmerk:VLD/2008/35842
Doc. soort :Ingekomen TK document
Aanm.datum :01-07-2008
Afdeling :CSSB

A1

Aan:
1. de gemeenteraden
2. de colleges van burgemeester en wethouders
van de gemeenten in het werkgebied
van de stadsregio Rotterdam
Afschrift aan: de portefeuillehouders financiën van deze gemeenten

datum • juni 2008
ons kenmerk • 32460
steller • E. Visser
telefoon • (010) 4173521
uw kenmerk •
betreft • Jaarverantwoording 2007

Geachte raadsleden, geacht college, Lijst ter inzage!

Het doet ons genoegen u het jaarverslag en de jaarrekening 2007, zoals vastgesteld door de Regioraad op 21 mei 2008, aan te kunnen bieden.

Hoogachtend,

het dagelijks bestuur van de stadsregio Rotterdam,

mr M.C. Jansen,
secretaris.

mr I.W. Opstelten,
voorzitter.

Bijlagen:
1. Jaarverslag 2007
2. Jaarrekening 2007

Meent 106 › Postbus 21051 › 3001 AB Rotterdam › T 010 417 23 89 › F 010 404 73 47 › E info@sr.rotterdam.nl › I www.stadsregio.info

Figure 3. The first and one of the following pages of a PDF generated for the iLiad

uments to PDF files. Everything else is handled by ConTeXt macros; the application only copies from a template include file into the TeX file. The used files are:

**agenda-iliad.tex**

This is the main TeX file, and in this file two different types of replacements take place.

In the listing below you see two lines that look like HTML syntax for so-called 'server side includes'. That is not a coincidence: as mentioned above, the application uses the same code for TeX files and HTML page generation.

The two `#include` files are read by the Perl script, and inserted in that place in the TeX output. The contents of those files are explained in the next paragraph.

The second type of replacement deals with the words in all caps between `#` markers. These keywords are replaced by the actual content (and meta-data) of the meeting. The keyword `#LIST#` is the most important of those because effectively that contains the whole content of the agenda, which is built up recursively.

A meeting agenda consists of meta-information like place and time, and a variable number of meeting items. Items can be organized into categories, and for each item there can be an optional number of related meeting documents. All of this is controlled by small template files that are inserted at various levels. Their names are predefined system constants.

```
\unprotect
<!--#include src='agenda-macros-00.tex' -->
<!--#include src='agenda-macros-iliad.tex' -->
\protect

\starttext
\startagenda[Gremium={#GREMIUM#},
             Datum={#DATUM#},
             Datumkort={#DATUMKORT#},
             Categorie={#CATEGORIE#},
             Aanvang={#AANVANG#},
             Locatie={#LOCATIE#},
             Aanhef={#AANHEF#},
             Koptitel={#TITLE2#},
             Titel={#TITLE#}]

\startpunten
#LIST#
\stoppunten

\stopagenda
\stoptext
```

**header_line.tinc**

This template is used for any category section headings.

```
\startheaderline
 [Titel={#TEXT#},Pagina=#PAGE#,Aard={#AARD#}]
\startheaderbody
#BODY#
\stopheaderbody
\stopheaderline
```

**puntnr_line.tinc**

This is the template for each of the separate agenda items.
#BODY# contains the explanatory text for this item, #DOCS# is a placeholder for the list of relevant meeting documents. The latter is itself built up programmatically because there can be any number of relevant documents per item.

```
\startpunt
 [Nummer={#NR#},Titel={#PUNT#},Aard={#AARD#}]
\startpuntbody
#BODY#
\stoppuntbody
\startpuntdocs
#DOCS#
\stoppuntdocs
\stoppunt
```

**puntdoc_line.tinc**

This is the first of three possible templates for a meeting document. It is used for meeting documents (i.e. exported PDF files) that will be included in the generated PDF as appendices.

```
\agendadocument[#ICON#]{#LINK#}{#LABEL#}
```

**puntnodoc_line.tinc**

This template will be used for meeting documents that should be included as appendices, but for which it is decided (based on a configuration parameter) that they are too large for actual inclusion. A separate macro is used so that an explanatory text can be printed in the output.

```
\agendanodocument[#ICON#]{#LABEL#}
```

**puntdoc_line_noembed.tinc**

This is the third possibility, intended for non-PDF meeting documents like Microsoft Word documents and PowerPoint presentations. Because files in these formats cannot be handled in the PDF output, no hyperlink is possible; thus the keyword #LINK# is not present in this case.

```
\agendadocument[#ICON#]{}{#LABEL#}
```

## ConTeXt macros

As mentioned above, the used ConTeXt macros are split over two separate files.

The first has the name agenda-macros-00.tex, and is used unaltered for all clients and all PDF layout. It contains a generic implementation of the macros we saw earlier in the template files. These macros only take care of the infrastructure; they don't do any layout themselves. Handling the layout is passed on to other macros via the ConTeXt command \directsetup.

Typical for the content of this file are macro definitions like this:

```
\def\dostartagenda[#1]%
  {\getparameters
      [Agenda]
      [Gremium=,Datum=,Datumkort=,
       Categorie=,Aanvang=,Locatie=,
       Aanhef=,Titel=,Koptitel=,
       Voorzitter=,
       #1]%
   \pagereference[firstpage]
   \directsetup{agenda:start}}

\def\stopagenda
  {\directsetup{agenda:stop}}
```

and this:

```
\def\agendadocument[#1]#2#3%
  {\doifnotempty {#2}
     {\doglobal \appendtoks
        \addimage{#2}{#3}\to \everyendagenda }%
   \def\DocumentType{#1}%
   \def\DocumentFile{#2}%
   \def\DocumentBody{#3}%
   \pagereference[#2-referer]
   \directsetup{agenda:document}}
```

The macro \addimage is the most interesting macro in this file. It receives the id and file name of an exported PDF document as arguments, and ensures that that PDF document is added page by page via \externalfigure. In slightly simplified form it looks like this:

```
\unexpanded\def\addimage#1#2{%
  \pagereference[#1]
  \xdef\previouspdf{\currentpdf}%
  \gdef\currentpdf{#1}%i
  \getfiguredimensions[#1.pdf]%
  \imgcount=\noffigurepages
  \dorecurse
      {\the\imgcount}
      {\externalfigure
          [#1.pdf]
          [page=\recurselevel,
           factor=max,
           size=cropbox]%
       \page}%
    \pagereference[#1-last]
}
```

In the appendices of the generated PDF (see figure 3) there is an extra interaction line at the bottom of the page containing three buttons that jump to the first page of the current appendix, the first page of the next appendix, and to the reference to this appendix in the meeting agenda itself. These hyperlinks use the values of \currentpdf and \previouspdf.

The needed setups and the general layout definitions are in the file agenda-macros-iliad.tex. This file can be instantiated with a specific version for a single client, but otherwise a generic version will be used: there is a default implementation file for each of the predefined PDF output types.

The PDF output layout for the e-reader devices differ from the layouts for PC screens or printer, but most of those differences are obvious. Of course there is a different (smaller) paper format. The room on an e-reader screen is limited, so it must be used optimally and therefore very small margins are used. PDF object compression is turned off because the e-reader hardware is very limited compared to a PC. The color support in ConTEXt is turned on, but only using grayscale.

The biggest difference is that the meeting documents that would normally remain separate files are included into the main output document. This makes moving the result file to the e-reader easier, but most important is that it improves the user friendliness of the result: external PDF links on the e-reader are either not supported at all (on the iLiad) or extremely slow (on the DR1000).

Parts of the content of agenda-macros-iliad.tex:

```
\definepapersize [iliad]
                 [width=124mm,height=152mm]

\setuppapersize [iliad] [iliad]

\enableregime[utf8]

\pdfminorversion = 4

\setuplayout[height=14.5cm,
             footer=12pt,
             footerdistance=6pt,
             width=11cm,
             topspace=12pt,
             header=0pt,
             backspace=24pt,
             leftmargin=12pt,
             rightmargin=12pt]

\setupcolors[state=start,conversion=yes,
             reduction=yes,rgb=no,cmyk=no]

\definecolor[papercolor][r=1,b=1,g=1]
...
\setupbackgrounds[page]
                 [state=repeat,
                  background=color,
                  backgroundcolor=papercolor]

...
\startsetups agenda:start
  \blank
  \setupfootertexts[\dofooteragenda]
  \setupfooter[state=high]
  \AgendaGremium
  \blank
  \starttabulate[|l|p|]
  \NC Datum:   \NC \ss\AgendaDatum\NC \NR
  \NC Aanvang: \NC \ss\AgendaAanvang\NC\NR
  \NC Locatie: \NC \ss\AgendaLocatie \NC\NR
  \stoptabulate
  \blank
\stopsetups

\startsetups agenda:stop
  \page
  \the\everyendagenda
  \everyendagenda={}
\stopsetups

\startsetups punten:start
  \startitemize[width=24pt]
\stopsetups

\startsetups punten:stop
  \stopitemize
\stopsetups

....

\startsetups agenda:nodocument
  {\DocumentBody
    {\tfx bestand te groot voor inclusie}\par }%
\stopsetups
```

## Summary

The generation of PDF documents is a small but important part of NotuDoc. We chose to use TEX for the high quality of the output and ConTEXt in particular because of the simple methods it offers to separate the layout definitions from the actual data.

Taco Hoekwater
Elvenkind BV
taco (at) elvenkind dot com

# LuaTeX says goodbye to Pascal

**Abstract**
LuaTeX 0.50 features a complete departure from Pascal source code. This article explains a little of the why and how of this change.

## Introduction

For more than a quarter of a century, all implementations of the TeX programming language have been based on WEB, a literate programming environment invented by Donald Knuth. WEB input files consist of a combination of Pascal source code and TeX explanatory texts, and have to be processed by special tools to create either input that is suitable for a Pascal compiler to create an executable (this is achieved via a program called `tangle`) or input for TeX82 to create a typeset representation of the implemented program (via a program called weave).

A WEB file is split into numerous small building blocks called ‘modules’ that consist of an explanatory text followed by source code doing the implementation. The explanations and source are combined in the WEB input in a way that attempts to maximise the reader’s understanding of the program when the typeset result is read sequentially. This article will, however, focus on the program source code.

## Pascal WEB

The listing that follows shows a small part of the WEB input of TeX82, defining two ‘modules’. It implements the function that scans for a left brace in the TeX input stream, for example at the start of a token list.

```
@ The |scan_left_brace| routine is called when a left brace is supposed to
be the next non-blank token. (The term ‘‘left brace’’ means, more precisely,
a character whose catcode is |left_brace|.) \TeX\ allows \.{\\relax} to
appear before the |left_brace|.

@p procedure scan_left_brace; {reads a mandatory |left_brace|}
begin @<Get the next non-blank non-relax non-call token@>;
if cur_cmd<>left_brace then
  begin print_err("Missing { inserted");
@.Missing \{ inserted@>
  help4("A left brace was mandatory here, so I’ve put one in.")@/
    ("You might want to delete and/or insert some corrections")@/
    ("so that I will find a matching right brace soon.")@/
    ("(If you’re confused by all this, try typing ‘I}’ now.)");
  back_error; cur_tok:=left_brace_token+"{"; cur_cmd:=left_brace;
  cur_chr:="{"; incr(align_state);
  end;
end;

@ @<Get the next non-blank non-relax non-call token@>=
repeat get_x_token;
until (cur_cmd<>spacer)and(cur_cmd<>relax)
```

It would take too much space here to explain everything about the WEB programming, but it is necessary to explain a few things. Any `@` sign followed by a single character introduces a WEB command that is interpreted by `tangle`, `weave` or both. The commands used in the listing are:

`@` This indicates the start of a new module, and the explanatory text that follows uses special TEX macros but is otherwise standard TEX input (this command is followed by a space).
`@p` This command starts the Pascal implementation code that will be transformed into the compiler input.
`@<` When this command is seen after a `@p` has already been seen, the text up to the following `@>` is a reference to a named module that is to be inserted in the compiler output at this spot. If a `@p` has not been seen yet, then instead it defines or extends that named module.
`@.` This defines an index entry up to the following `@>`, used by `weave` and filtered out by `tangle`.
`@/` This is a command to control the line breaks in the typeset result generated by `weave`.

If you are familiar with Pascal, the code above may look a bit odd to you. The main reason for that apparent weirdness of the Pascal code is that the WEB system has a macro processor built in. The symbol `help4` is actually one of those macros, and it handles the four sets of parenthesized strings that follow. In expanded form, its output would look like this:

```
begin
  help_ptr:=4;
  help_line[3]:=”A left brace was mandatory here, so I’ve put one in.”;
  help_line[2]:=”You might want to delete and/or insert some corrections”;
  help_line[1]:=”so that I will find a matching right brace soon.”;
  help_line[0]:=”(If you’re confused by all this, try typing ‘I}’ now.)”;
end
```

The symbol `print_err` is another macro, and it expand into this:

```
begin if interaction=error_stop_mode then wake_up_terminal;
  print_nl(”! ”); print(”Missing { inserted”);
  end
```

## Tangle output

Now let’s have a look at the generated Pascal.

```
{:381}{403:}procedure scanleftbrace;begin{404:}repeat getxtoken;
until(curcmd<>10)and(curcmd<>0){:404};
if curcmd<>1 then begin begin if interaction=3 then;
if filelineerrorstylep then printfileline else printnl(262);print(671);
end;begin helpptr:=4;helpline[3]:=672;helpline[2]:=673;helpline[1]:=674;
helpline[0]:=675;end;backerror;curtok:=379;curcmd:=1;curchr:=123;
incr(alignstate);end;end;{:403}{405:}procedure scanoptionalequals;
```

That looks even weirder! Don’t panic. Let go through the changes one by one.

□ First, `tangle` does not care about indentation. The result is supposedly only read by the Pascal compiler, so everything is glued together.
□ Second, `tangle` has added Pascal comments before and after each module that give the sequence number of that module in the WEB source: those are 403 and 404.
□ Third, `tangle` removes the underscores from identifiers, so `scan_left_brace` became `scanleftbrace` etcetera.

□ Fourth, many of the identifiers you thought were present in the WEB source are really macros that expand to constant numbers, which explains the disappearance of `left_brace`, `spacer`, `relax`, `left_brace_token`, and `error_stop_mode`.

□ Fifth, macro expansion has removed `print_err`, `help4` and `wake_up_terminal` (which expands to nothing).

□ Sixth, WEB does not make use of Pascal strings. Instead, the strings are collected by `tangle` and output to a separate file that is read by the generated executable at runtime and then stored in a variable that can be indexed as an array. This explains the disappearance of all the program text surrounded by `"`.

□ Seventh, addition of two constants is optimized away in some cases. Since single character strings in the input have a fixed string index (its ASCII value), `left_brace_token+"{"` becomes `379` instead of `256+123`.

## Web2c

Assuming you have generated the Pascal code above via `tangle`, you now run into a small practical problem: the limited form of Pascal that is used by the WEB program is not actually understood by any of the current Pascal compilers and has not for a quite some time. The build process of modern TEX distributions therefore make use of a different program called `web2c` that converts the `tangle` output into C code.

The result of running `web2c` on the above snippet is in the next listing.

```
void scanleftbrace ( void )
{
  do {
      getxtoken () ;
  } while ( ! ( ( curcmd != 10 ) && ( curcmd != 0 ) ) ) ;
  if ( curcmd != 1 )
  {
    {
      if ( interaction == 3 )
      ;
      printnl ( 262 ) ;
      print ( 671 ) ;
    }
    {
      helpptr = 4 ;
      helpline [3 ]= 672 ;
      helpline [2 ]= 673 ;
      helpline [1 ]= 674 ;
      helpline [0 ]= 675 ;
    }
    backerror () ;
    curtok = 379 ;
    curcmd = 1 ;
    curchr = 123 ;
    incr ( alignstate ) ;
  }
}
```

This output is easier to read for us humans because of the added indentation and the addition of parentheses after procedure calls, but does not significantly alter the code.

Actually, some more tools are used by the build process of modern TEX distributions, for example there is a small program that splits the enormous source file into a few smaller C source files, and another program converts Pascal `write` procedure calls to C `printf` function calls.

## Issues with WEB development

Looking at the previous paragraphs, there are number of issues when using WEB for continuous development.

- Compiling WEB source, especially on a modern platform without a suitable Pascal compiler, is a fairly complicated and lengthy process requiring a list of dedicated tools that have to be run in the correct order.
- Literate programming environments like WEB never really caught on, so it is hard to find programmers that are familiar with the concepts, and there are nearly no editors that support its use with the editor features that modern programmers have come to depend on, like syntax highlighting.
- Only a subset of an old form of Pascal is used in the Knuthian sources, and as this subset is about the extent of Pascal that is understood by the web2c program, that is all that can be used.

  This makes interfacing with external programming libraries hard, often requiring extra C source files just to glue the bits together.
- The ubiquitous use of WEB macros makes the external interface even harder, as these macros do not survive into the generated C source.
- Quite a lot of useful debugging information is irretrievably lost in the tangle stage. Of course, TEX82 is so bug-free that this hardly matters, but when one is writing new extensions to TEX, as is the case in LuaTEX, this quickly becomes problematic.
- Finally, to many current programmers WEB source simply feels over-documented and even more important is that the general impression is that of a finished book: sometimes it seems like WEB actively discourages development. This is a subjective point, but nevertheless a quite important one.

## Our solution

In the winter of 2008–2009, we invested a lot of time in hand-converting the entire LuaTEX code base into a set of C source files that are much closer to current programming practices. The big WEB file has been split into about five dozen pairs of C source files and include headers.

During conversion, quite a bit of effort went into making the source behave more like a good C program should: most of the WEB macros with arguments have been converted into C #defines, most of the numerical WEB macros are now C enumerations, and many of the WEB global variables are now function arguments or static variables. Nevertheless, this part of the conversion process is nowhere near complete yet.

The new implementation of scan_left_brace in LuaTEX looks like this:

```
/*
The |scan_left_brace| routine is called when a left brace is supposed
to be the next non-blank token. (The term ‘‘left brace’’ means, more
precisely, a character whose catcode is |left_brace|.) \TeX\ allows
\.{\\relax} to appear before the |left_brace|.
*/

void scan_left_brace(void)
{                               /* reads a mandatory |left_brace| */
    /* Get the next non-blank non-relax non-call token */
    do {
        get_x_token();
    } while ((cur_cmd == spacer_cmd) || (cur_cmd == relax_cmd));

    if (cur_cmd != left_brace_cmd) {
        print_err("Missing { inserted");
        help4("A left brace was mandatory here, so I’ve put one in.",
              "You might want to delete and/or insert some corrections",
              "so that I will find a matching right brace soon.",
              "If you’re confused by all this, try typing ‘I}’ now.");
        back_error();
        cur_tok = left_brace_token + ’{’;
        cur_cmd = left_brace_cmd;
        cur_chr = ’{’;
        incr(align_state);
    }
}
```

I could write down all of the differences with the previously shown C code, but that is not a lot of fun so I will leave it to the reader to browse back a few pages and spot all the changes. The only thing I want to make a note of is that we have kept all of the WEB explanatory text in C comments, and we are actively thinking about a way to reinstate the ability to create a beautifully typeset source book.

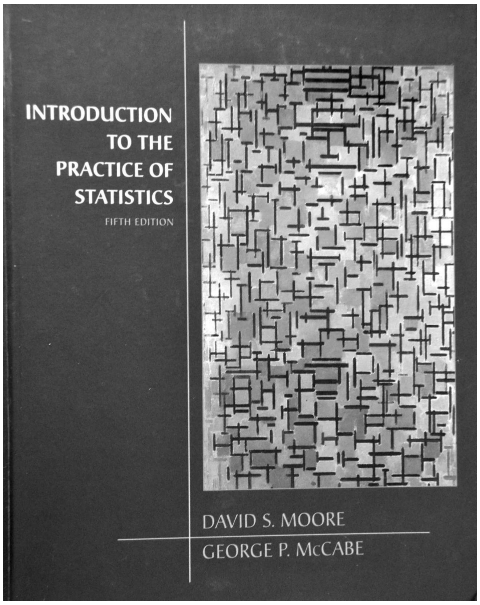
Taco Hoekwater
taco (at) luatex dot org

# The Typesetting of Statistics

**Abstract**
The Dutch translation of the 750 page textbook 'Introduction to the Practice of Statistics' is typeset using a set of ConTeXt macros. This article gives a short impression of the production process of this book, showing that the use of TeX for the actual typesetting was perhaps the least cumbersome part of the process.

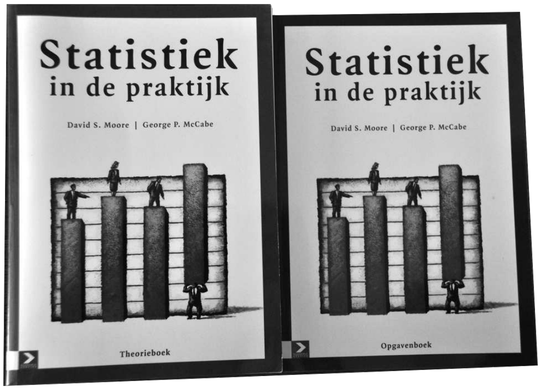
## The original version

'Introduction to the Practice of Statistics' is a fairly hefty textbook in what I consider a typical 'United States' style: it is full-color throughout, in a fairly large format that combines theory and practice, bound in a single hardcover book, and is printed on rather thin paper. It has been typeset using LaTeX.



## The Dutch version

'Statistiek in de praktijk' in turn is a fairly typical adaptation to the Dutch market: The body is printed in grayscale, it has a slightly smaller format, the original is split into two paperbacks (a textbook and a workbook) and it is printed on standard thickness paper. It is typeset using ConTeXt.



## History of the Dutch version

The first edition was printed elsewhere, and the original Dutch sources that I received used macroTeX by Amy Hendrickson. This source was converted to plain TeX for the second and third editions, using a bunch of special purpose extension macros.

During this time, there was one TeX source file for each chapter body, one for each chapter title page, and one for each chapter preface.

The Dutch publisher skipped the fourth edition to save on expenses.

For the fifth edition, I switched to ConTeXt for easier use of graphic objects and references. In the process, the source file layout was converted to the standard ConTeXt model for projects so that a single `texexec` call can be used to generate each of the books. However, makeindex is still being used for the index generation instead of the normal ConTeXt methods.

## The fifth edition

The Dutch translation of the fifth edition was sent to us as a bunch of Microsoft Word files (one per original chapter), and we were lucky enough to also receive the original sources of the English version.

The Word data format itself was fine with us, but actual content posed a number of problems:

□ There were quite a lot of vocabulary differences between the chapters because four different translators had been working simultaneously.

- □ Parts of the translation were missing: anything in the fifth edition that was identical to the text in the third edition was not translated. We were expected to copy these blocks over from the Dutch sources of the third edition.
- □ There were accidental omissions in the translation.
- □ The graphics also needed to be translated, converted to grayscale, and corrected in Adobe Illustrator, then exported as EPS, and finally converted to PDF.
- □ The page location remarks in the translation files referred to the Dutch version of the third edition, instead of to the actual page numbers in the English fifth edition.

The last item above was by far the most problematic, because after the first proofs came back, it became necessary to have five pretty large piles of paper on the desk at all times:

1. The printout of the original Word files (because of the meta data).
2. The Dutch third edition (for continuous reference).
3. The English fifth edition (also for reference).
4. The editor-corrected printout of the Dutch fifth edition proofs (for correction).
5. A stack of printouts of various email messages and documents with corrections and addenda sent in by the translators.

## Handling Word files

For the main text, the Word files were exported to HTML in OpenOffice, followed by a cleanup script written by Siep Kroonenberg.

The tabular material was imported from the English LaTeX sources using a dedicated ConTeXt module that implements the LaTeX `tabular` environment. Using that, much of the tabular data could be copied straight over. The exceptions were tables containing decimal points (those were converted to commas) and tables dealing with money (where all amounts had to be converted into euros).

Mathematical formulas were re-keyed. This was not a big problem, as almost all formulas needed editing for localization anyway.

Taco Hoekwater
Elvenkind BV
taco (at) elvenkind dot com

# MetaPost 2 project goals

**Abstract**
Now that MetaPost 1.200 has been released the time has finally come to focus on the numerical precision extensions that we have been hinting at for some years already. Version 2.000 of MetaPost will have a runtime configurable precision and infinite numeric input range.

## Introduction

A few years ago John Hobby transferred MetaPost development to a team of users of which Taco Hoekwater is now responsible for the coding. Some pending bugs were resolved and a few extensions made.

A major step was made when the code base was converted to C and MetaPost became a library component. This made usage in, for instance, luaTEX possible, and we could also get rid of some dependencies of external programs. This project was funded by TEX user groups and the results have been reported at user group meetings and in journals.

Recently an extra backend was added (SVG) which was also partially funded. The version that ships with TEX Live 2009 is the first version that is built on top of the library and it has these extensions on board.

## More extensions

However, we are not yet finished as it has been a long standing wish to free MetaPost from one particularly significant limitation. In the original MetaPost library proposal we wrote in May 2007, one of the big user-side problem points mentioned was:

> “All number handling is based on fractions of a 32-bit integer and user input often hits one of the many boundaries that are the result of that. For instance, no numbers can be larger than 16384 and there is a noticeable lack of precision in the intersection point calculations.”

It is for this reason that we will start the next stage in the development of the MetaPost library. The aim is to resolve the mentioned limitation once and for all.

In order to do so, we will have to replace the MetaPost internal 32-bit numeric values with something more useful, and to achieve that, the plan is to incorporate one or both of the following external libraries.

### GNU MPFR library

From the web site:

> “The *MPFR* library is a C library for multiple-precision floating-point computations with correct rounding. MPFR has continuously been supported by the INRIA and the current main authors come from the CACAO and Arénaire project-teams at Loria (Nancy, France) and LIP (Lyon, France) respectively; see more on the credit page. MPFR is based on the GMP multiple-precision library.
>
> The main goal of MPFR is to provide a library for multiple-precision floating-point computation which is both efficient and has a well-defined semantics. It copies the good ideas from the ANSI/IEEE-754 standard for double-precision floating-point arithmetic (53-bit mantissa).”

See http://www.mpfr.org/ for more details.

### IBM decNumber

From the web site:

> “decNumber is a high-performance decimal arithmetic library in ANSI C, especially suitable for commercial and human-oriented applications.”

See http://www.alphaworks.ibm.com/tech/decnumber/ for more details.

We have not decided yet which one will be used, and possibly we will include both. MPFR will likely be faster and has a larger development base, but decNumber is more interesting from a user interface point of view because decimal calculus is generally more intuitive. For both libraries the same internal steps need to be taken, so that decision can be safely postponed until later in the project. The final decision will be based on a discussion to be held on the MetaPost mailing list.

## The goals of the project

The project can be split up into a number of subsidiary goals.

### Dynamic memory allocation

Because values in any numerical calculation library are always expressed as C pointers, it is necessary to move away from the current array-based memory structure with overloaded members to a system using dynamic allocation (using `malloc()`) and named structure components everywhere, so that all internal MetaPost values can be expressed as C pointers internally.

As a bonus, this will remove the last bits of static allocation code from MetaPost so that it will finally be able to use all of the available RAM.

### Internal API

An internal application programming interface layer will need to be added for all the internal calculation functions and the numeric parsing and serialization routines. All such functions will have to be stored in an array of function pointers, thus allowing a run-time switch between 32-bit backward-compatible calculation and an arbitrary precision library.

This internal interface will also make it possible to add additional numerical engines in the future.

The external application programming interface layer will be extended to make direct access to the numerical and path handling functions possible.

### Backends

The SVG and PostScript backends need to be updated to use double precision float values for exported points instead of the current 32-bit scaled integers.

In the picture export API, doubles are considered to be the best common denominator because there is an acceptable range and precision and they are simple to manipulate in all C code. This way, the actual SVG and PostScript backend implementations and the Lua bindings can remain small and simple.

### Input language

The input language needs to be extended in order to fulfil the following project objectives.

1. It must be possible to select which numerical engine to use.
2. In the case of an arbitrary precision engine, it has to be possible to set up the actual precision.
3. It has to be possible to read (and write) numerical values in scientific notation.
4. Some mathematical functions (especially trigonometry) and constants (like $\pi$) will have to be added to make sure all the numerical engines present a unified interface while still offering the best possible precision.

Here is a tentative input example that would select the decNumber library with 50 decimal digits of precision:

```
mathengine := "decNumber";
mathprecision := 50;
beginfig(1);
z1 = (1/3,0.221533334556544532266322);
v = 23.45678888E512;
x2 = v/1E511;
endfig;
end.
```

## Timeline

Thanks to funding received from various local user groups, we hope to be able to have MetaPost 2 ready in time for the TUG and EuroTEX conferences in 2010.

Hans Hagen & Taco Hoekwater
taco (at) metapost dot org

# Using LaTeX as a computing language

## Introduction

It is possible to use LaTeX as a primitive computing language to do some repetitive calculations, examples of which are

- calculation of dates for a timetable
- vector and scalar products
- solution of second-order constant coefficient linear differential equations
- cancelling common factors in a fraction
- adding up marks on a student’s script

My question is whether these are useful? If so, whether it is worth extending this?

## Dates on a timetable

*Introduction.* I produce a timetable for the lecture course I give. The dates on the timetable are a constant number of days relative to the start date (e.g. if the start date is 8 January, then the first computing practical is on 22 January, a fortnight later).

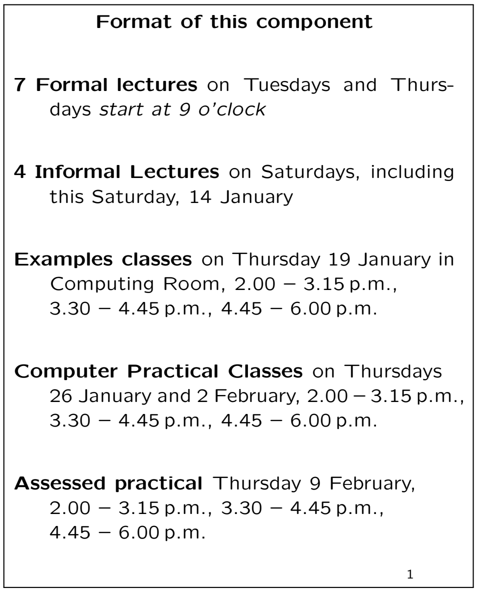
I had done my own programming before I was aware of the `datenumber` and `ifthen` packages, and would use those instead now.

The material is used for a slide at the beginning of the course, and in LaTeX 2.09 format—don’t change a winning formula.

*Coding.* The coding is within the actual input file, and not a separate package.

```
\makeatletter
   \newcount{\@startdate}
   \@startdate=0
   \newcount{\@tempdate}
   \newcount{\@dayplus}
   \newcommand{\startdate}[1]%
      {\advance \@startdate by #1}%
   \newcommand{\dayplus}[1]%
      {\@tempdate = \@startdate \advance \@tempdate by #1%
      \ifnum \@tempdate < 32 \the\@tempdate~January%
         \else \advance \@tempdate by -31 \the\@tempdate~February\fi}%
\makeatother

\begin{slide}{}
   \begin{center} \bf Format of this component\end{center}
   \begin{description}
      \item[7 Formal lectures] on Tuesdays and Thursdays
         {\em start at 9 o’clock}
         \startdate{15}
      \item[4 Informal Lectures] on Saturdays,
         including this Saturday, \dayplus{2}
      \item[Examples classes] on Thursday \dayplus{7}
         in Computing Room, 2.00 -- 3.15\,p.m., \\3.30 -- 4.45\,p.m.,
         4.45 -- 6.00\,p.m.
      \item[Computer] Practical Classes  on
         Thursdays\\ \dayplus{14}  and \dayplus{21},
         2.00 -- 3.15\,p.m., 3.30 -- 4.45\,p.m., 4.45 -- 6.00\,p.m.
      \item[Assessed practical]
         Thursday \dayplus{28}, \\ 2.00 -- 3.15\,p.m.,
         3.30 -- 4.45\,p.m., \\4.45 -- 6.00\,p.m.
   \end{description}
\end{slide}
```

*Example of output.* This produces the output in Figure 1.



**Format of this component**

**7 Formal lectures** on Tuesdays and Thursdays *start at 9 o'clock*

**4 Informal Lectures** on Saturdays, including this Saturday, 14 January

**Examples classes** on Thursday 19 January in Computing Room, 2.00 – 3.15 p.m., 3.30 – 4.45 p.m., 4.45 – 6.00 p.m.

**Computer Practical Classes** on Thursdays 26 January and 2 February, 2.00 – 3.15 p.m., 3.30 – 4.45 p.m., 4.45 – 6.00 p.m.

**Assessed practical** Thursday 9 February, 2.00 – 3.15 p.m., 3.30 – 4.45 p.m., 4.45 – 6.00 p.m.

1

Figure 1. Example of dates on a timetable

## Calculating cross- and dot-products of vectors

*Definitions.* Exercises involving the products, be they cross or dot, of vectors are invariably with integer coefficients, and so it is easy to calculate them using integer arithmetic.

If two vectors **a** and **b** have Cartesian components $a_1, a_2, a_3$ (or $a_1\mathbf{i}+a_2\mathbf{j}+a_3\mathbf{k}$) and $b_1, b_2, b_3$ (or $b_1\mathbf{i}+b_2\mathbf{j}+b_3\mathbf{k}$) respectively, then the dot (or scalar) product is defined[1] by

$$\mathbf{a}\cdot\mathbf{b} = a_1b_1 + a_2b_2 + a_3b_3$$

and the cross (or vector) product is defined by

$$\mathbf{a}\times\mathbf{b} = (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}.$$

Although it is easy to do the arithmetic using the `calc` package, good style in mathematics dictates that $0\mathbf{i}+4\mathbf{j}+0\mathbf{k}$ is not as elegant as $4\mathbf{j}$ and that intermediate calculations that involve a negative number ought to have that negative number in brackets, e.g. $4\times -3 - -3\times 2$ should be replaced by $(4\times(-3)) - ((-3)\times 2)$.

So in this package, I have tried to avoid these lapses of style.

*Coding of the package.* The package coding is:

```
\RequirePackage{calc, ifthen}

\renewcommand{\i}{{\mathbf i}}
\renewcommand{\j}{{\mathbf j}}
\renewcommand{\k}{{\mathbf k}}

\newcommand{\bfcdot}%
    {\mathbin{\mbox{\normalfont \bf \raise 0.4ex \hbox{.}}}}

\newcommand{\bftimes}{\mathbin{\mbox{\normalfont \bf\textsf{x}}}}

%% Define the counters for each of the components and a temporary one.
\newcounter{@ai}\newcounter{@aj}\newcounter{@ak}
\newcounter{@bi}\newcounter{@bj}\newcounter{@bk}
\newcounter{@ci}\newcounter{@cj}\newcounter{@ck}
\newcounter{@temp}

%% A command to include brackets if the value is less than zero
\newcommand{\@negbrackets}[1]%
    {\setcounter{@temp}{#1}
    \ifthenelse{\value{@temp}<0}%
        {(\the@temp)}%
        {\the@temp}
    } % end \@negbrackets

%% to check whether a previous component exists
\newboolean{@previous}
\setboolean{@previous}{false}

%% to be used for the first non-zero component
\newcommand{\@firstcomponent}[2]%
    {\setcounter{@temp}{#1}
    \ifthenelse{\value{@temp}=0}%
        {\relax}%
        {\setboolean{@previous}{true}
        \ifthenelse{\value{@temp}=1}%
            {#2}%
            {\the@temp\,#2}
        }
    } % end \@firstcomponent

%% to be used if the previous component is not zero
\newcommand{\@subsequentcomponent}[2]%
    {\setcounter{@temp}{#1}
    \ifthenelse{\boolean{@previous}}%
        {\ifthenelse{\value{@temp}=0}%
            {\relax}%
            {\ifthenelse{\value{@temp}=1}%
                {+#2}%
                {\ifthenelse{\value{@temp}=-1}%
                    {-#2}%
                    {\ifthenelse{\value{@temp}<0}%
                        {\the@temp\,#2}%
                        {+\the@temp\,#2}
                    }
                }
            }
        }%
        {\@firstcomponent{#1}{#2}}%
    } % end \@subsequentcomponent

%% writes a zero vector if all previous components are zero,
%% and resets the switch
\newcommand{\@zerovector}%
    {\ifthenelse{\boolean{@previous}}%
        {\setboolean{@previous}{false}}%
        {\mathbf 0}
    } % ends \@zerovector

%% puts it all together and produces the output
\newcommand{\crossproduct}[6]%
    {\setcounter{@ai}{#1}\setcounter{@aj}{#2}\setcounter{@ak}{#3}
    \setcounter{@bi}{#4}\setcounter{@bj}{#5}\setcounter{@bk}{#6}
    \setcounter{@ci}{\value{@aj}*\value{@bk}-\value{@ak}*\value{@bj}}
    \setcounter{@cj}{\value{@ak}*\value{@bi}-\value{@ai}*\value{@bk}}
    \setcounter{@ck}{\value{@ai}*\value{@bj}-\value{@aj}*\value{@bi}}%
    (\@firstcomponent{#1}{\i}%
     \@subsequentcomponent{#2}{\j}\@subsequentcomponent{#3}{\k}
        \@zerovector) \bftimes
    (\@firstcomponent{#4}{\i}%
     \@subsequentcomponent{#5}{\j}\@subsequentcomponent{#6}{\k}
        \@zerovector)\\
    &=& \left(\@negbrackets{\the@aj}\times\@negbrackets{\the@bk}
        -\@negbrackets{\the@ak}\times\@negbrackets{\the@bj} \right)\i +
        \left(\@negbrackets{\the@ak}\times\@negbrackets{\the@bi}
        -\@negbrackets{\the@ai}\times\@negbrackets{\the@bk}\right)\j +
        \left(\@negbrackets{\the@ai}\times\@negbrackets{\the@bj}
        -\@negbrackets{\the@aj}\times\@negbrackets{\the@bi}\right)\k \\
    &=& \@firstcomponent{\the@ci}{\i}   \@subsequentcomponent{\the@cj}{\j}
          \@subsequentcomponent{\the@ck}{\k}\@zerovector
    } %ends \crossproduct

\newcommand{\dotproduct}[6]%
    {\setcounter{@ai}{#1}\setcounter{@aj}{#2}\setcounter{@ak}{#3}
    \setcounter{@bi}{#4}\setcounter{@bj}{#5}\setcounter{@bk}{#6}
    \setcounter{@ci}{\value{@ai}*\value{@bi}+\value{@aj}*\value{@bj}+
    \value{@ak}*\value{@bk}}%
    (\@firstcomponent{#1}{\i}%
     \@subsequentcomponent{#2}{\j}%
     \@subsequentcomponent{#3}{\k}
        \@zerovector) .
    (\@firstcomponent{#4}{\i}%
     \@subsequentcomponent{#5}{\j}\@subsequentcomponent{#6}{\k}
        \@zerovector)\\
    &=& \@negbrackets{\the@ai}\times\@negbrackets{\the@bi}
        +\@negbrackets{\the@aj}\times\@negbrackets{\the@bj}
        +\@negbrackets{\the@ak}\times\@negbrackets{\the@bk} \\
    &=& \the@ci
    } %ends \dotproduct
```

*Input to the package.* With this package, the following produces the output in Figure 2.

```
\begin{eqnarray*}
    \mathbf{p} \bftimes \mathbf{q}& = &\crossproduct{-3}{-1}{6}{2}{-2}{4}
\end{eqnarray*}

\begin{eqnarray*}
```

```
    \mathbf{p} \bftimes \mathbf{q}& = &\crossproduct{8}{-1}{6}{0}{0}{4}
\end{eqnarray*}

\begin{eqnarray*}
    \mathbf{q}\bftimes \mathbf{r} &=& \crossproduct{-3}{0}{4}{0}{0}{4}
\end{eqnarray*}

\begin{eqnarray*}
    \mathbf{q}\bftimes \mathbf{r} &=& \crossproduct{0}{0}{0}{0}{0}{0}
\end{eqnarray*}

\begin{eqnarray*}
    \mathbf{p} \bfcdot \mathbf{q}& = &\dotproduct{-3}{-1}{6}{2}{-2}{4}
\end{eqnarray*}

\begin{eqnarray*}
    \mathbf{p} \bfcdot \mathbf{q}& = &\dotproduct{8}{-1}{6}{0}{0}{4}
\end{eqnarray*}

\begin{eqnarray*}
    \mathbf{q}\bfcdot \mathbf{r} &=& \dotproduct{-3}{0}{4}{0}{0}{4}
\end{eqnarray*}

\begin{eqnarray*}
    \mathbf{q}\bfcdot \mathbf{r} &=& \dotproduct{0}{0}{0}{0}{0}{0}
\end{eqnarray*}
```

A very similar package is written for calculating determinants of $2 \times 2$ and $3 \times 3$ matrices.

## Solution of second-order constant coefficient homogeneous linear differential equations

*Definitions.* I shan't be solving differential equations; but *reverse engineer* them from the solution.

If the solutions to the auxiliary equation are real, say $\lambda_1$ and $\lambda_2$, then the differential equation is

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - (\lambda_1 + \lambda_2)\frac{\mathrm{d}y}{\mathrm{d}x} + \lambda_1\lambda_2 y = 0$$

and has solution

$$y = A\mathrm{e}^{\lambda_1 x} + B\mathrm{e}^{\lambda_2 x}.$$

So first choose $\lambda_1$ and $\lambda_2$ and then generate the question and solution from those two numbers.

If the solutions to the auxiliary equation are complex, say $\alpha \pm \beta\mathrm{i}$, then the differential equation is

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - 2\alpha\frac{\mathrm{d}y}{\mathrm{d}x} + \left(\alpha^2 + \beta^2\right) y = 0$$

and has solution

$$y = \mathrm{e}^{\alpha x}\left(A\cos(\beta x) + B\sin(\beta x)\right).$$

So first choose $\alpha$ and $\beta$ and generate the question and solution from those two numbers.

*Coding.*

```
\RequirePackage{ifthen, calc}
\parindent 0pt

\newboolean{@positive} % to cater for negative sign
```

$$\begin{aligned}
\mathbf{p}\times\mathbf{q} &= (-3\,\mathbf{i}-\mathbf{j}+6\,\mathbf{k})\times(2\,\mathbf{i}-2\,\mathbf{j}+4\,\mathbf{k})\\
&= ((-1)\times 4-6\times(-2))\,\mathbf{i}+(6\times 2-(-3)\times 4)\,\mathbf{j}+((-3)\times(-2)-(-1)\times 2)\,\mathbf{k}\\
&= 8\,\mathbf{i}+24\,\mathbf{j}+8\,\mathbf{k}
\end{aligned}$$

$$\begin{aligned}
\mathbf{p}\times\mathbf{q} &= (8\,\mathbf{i}-\mathbf{j}+6\,\mathbf{k})\times(4\,\mathbf{k})\\
&= ((-1)\times 4-6\times 0)\,\mathbf{i}+(6\times 0-8\times 4)\,\mathbf{j}+(8\times 0-(-1)\times 0)\,\mathbf{k}\\
&= -4\,\mathbf{i}-32\,\mathbf{j}
\end{aligned}$$

$$\begin{aligned}
\mathbf{q}\times\mathbf{r} &= (-3\,\mathbf{i}+4\,\mathbf{k})\times(4\,\mathbf{k})\\
&= (0\times 4-4\times 0)\,\mathbf{i}+(4\times 0-(-3)\times 4)\,\mathbf{j}+((-3)\times 0-0\times 0)\,\mathbf{k}\\
&= 12\,\mathbf{j}
\end{aligned}$$

$$\begin{aligned}
\mathbf{q}\times\mathbf{r} &= (\mathbf{0})\times(\mathbf{0})\\
&= (0\times 0-0\times 0)\,\mathbf{i}+(0\times 0-0\times 0)\,\mathbf{j}+(0\times 0-0\times 0)\,\mathbf{k}\\
&= \mathbf{0}
\end{aligned}$$

$$\begin{aligned}
\mathbf{p}\cdot\mathbf{q} &= (-3\,\mathbf{i}-\mathbf{j}+6\,\mathbf{k}).(2\,\mathbf{i}-2\,\mathbf{j}+4\,\mathbf{k})\\
&= (-3)\times 2+(-1)\times(-2)+6\times 4\\
&= 20
\end{aligned}$$

$$\begin{aligned}
\mathbf{p}\cdot\mathbf{q} &= (8\,\mathbf{i}-\mathbf{j}+6\,\mathbf{k}).(4\,\mathbf{k})\\
&= 8\times 0+(-1)\times 0+6\times 4\\
&= 24
\end{aligned}$$

$$\begin{aligned}
\mathbf{q}\cdot\mathbf{r} &= (-3\,\mathbf{i}+4\,\mathbf{k}).(4\,\mathbf{k})\\
&= (-3)\times 0+0\times 0+4\times 4\\
&= 16
\end{aligned}$$

$$\begin{aligned}
\mathbf{q}\cdot\mathbf{r} &= (\mathbf{0}).(\mathbf{0})\\
&= 0\times 0+0\times 0+0\times 0\\
&= 0
\end{aligned}$$

Figure 2. Example of dot and cross products

```
\setboolean{@positive}{true}

\newcounter{@a}\newcounter{@b}
\newcounter{@trace}\newcounter{@det}
\newcounter{diffcoefficient} % for external use
\newcounter{functioncoefficient} % for external use
\newcounter{@x}
\setcounter{@a}{3}
```

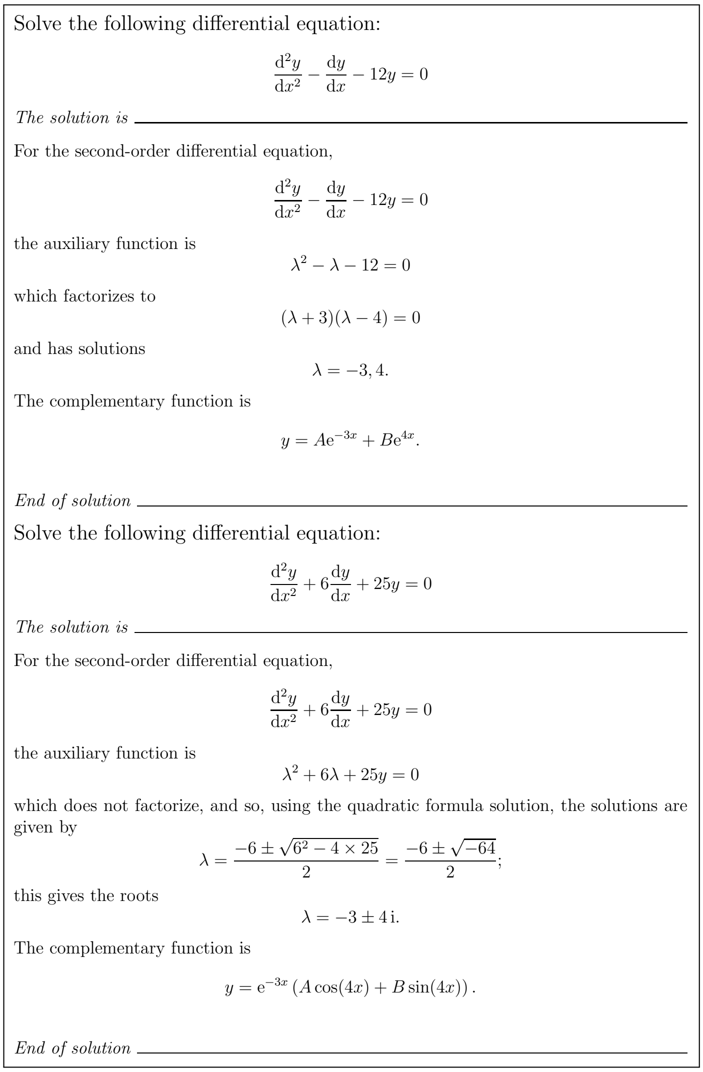
```
\setcounter{@b}{4}

\newcommand{\sign}{\ifthenelse{\boolean{@positive}}{+}{-}}

\newcommand{\dequreal}[2]%
        {\setcounter{@a}{#1}\setcounter{@b}{#2}
        \setcounter{@x}{\value{@a}+\value{@b}}
        \setcounter{diffcoefficient}{-\value{@a}-\value{@b}}
        \setcounter{functioncoefficient}{\value{@a}*\value{@b}}
        \frac{\mathrm{d}^2y}{\mathrm{d}x^2}
        \ifthenelse{\value{@x}=0}%
            {\relax}%
            {\ifthenelse{\value{@x}<0}%
                {\setcounter{@trace}{-\value{@x}}\setboolean{@positive}{true}}%
                {\setboolean{@positive}{false}\setcounter{@trace}{\value{@x}}}
            \sign
            \ifthenelse{\value{@trace}=1}%
                {\relax}%
                {\the@trace} %% to remove 1
            \frac{\mathrm{d}y}{\mathrm{d}x}
        }
        \setcounter{@x}{{\value{@a}*\value{@b}}}
        \ifthenelse{\value{@x}<0}%
            {\setcounter{@det}{-\value{@x}}\setboolean{@positive}{false}}%
            {\setcounter{@det}{\value{@x}}\setboolean{@positive}{true}}
        \sign
        \ifthenelse{\value{@det}=1}%
            {\relax}%
            {\the@det} %% to remove 1
        y=0 %\quad (\mbox{arguments are } #1,#2)
    } % ends dequreal

\newcommand{\auxiliaryreal}[2]%
        {\setcounter{@a}{#1}\setcounter{@b}{#2}
        \setcounter{@x}{\value{@a}+\value{@b}}
        \lambda^2
        \ifthenelse{\value{@x}=0}%
            {\relax}%
            {\ifthenelse{\value{@x}<0}%
                {\setcounter{@trace}{-\value{@x}}\setboolean{@positive}{true}}%
                {\setboolean{@positive}{false}\setcounter{@trace}{\value{@x}}}
            \sign
            \ifthenelse{\value{@trace}=1}{\relax}{\the@trace} %% to remove 1
            \lambda}
        \setcounter{@x}{{\value{@a}*\value{@b}}}
        \ifthenelse{\value{@x}<0}%
            {\setcounter{@det}{-\value{@x}}\setboolean{@positive}{false}}%
            {\setcounter{@det}{\value{@x}}\setboolean{@positive}{true}}
        \sign   \ifthenelse{\value{@det}=1}{\relax}{\the@det} %% to remove 1
        =0
    } % ends \auxiliaryreal

\newcommand{\auxiliaryfactors}[2]%
    {(\lambda \setcounter{@x}{-#1}
    \ifthenelse{\value{@x}<0}%
        {\relax}%
        {+}
    \the@x)
    (\lambda \setcounter{@x}{-#2}
    \ifthenelse{\value{@x}<0}%
```

```
        {\relax}%
        {+}
    \the@x) =0
    } % ends \auxiliaryfactors

\newcommand{\cfreal}[2]%
    {y=A\mathrm{e}^{#1 x}+B\mathrm{e}^{#2 x}
    } % ends \cfreal

\newcommand{\dequcomplex}[2]%
        {\setcounter{@a}{#1}\setcounter{@b}{#2}
        \setcounter{@x}{2*\value{@a}}
        \setcounter{diffcoefficient}{-2*\value{@a}}
        \setcounter{functioncoefficient}
                         {{\value{@a}*\value{@a}+\value{@b}*\value{@b}}}
        \frac{\mathrm{d}^2y}{\mathrm{d}x^2}
        \ifthenelse{\value{@x}=0}%
            {\relax}%
            {\ifthenelse{\value{@x}<0}%
                {\setcounter{@trace}{-\value{@x}}\setboolean{@positive}{true}}%
                {\setboolean{@positive}{false}\setcounter{@trace}{\value{@x}}}
            \sign
            \ifthenelse{\value{@trace}=1}{\relax}{\the@trace} %% to remove 1
        \frac{\mathrm{d}y}{\mathrm{d}x}}
        \setcounter{@x}{{\value{@a}*\value{@a}+\value{@b}*\value{@b}}}
        + \the@x    y=0 %\quad (\mbox{arguments are } #1,#2)
    } % ends \dequcomplex

\newcommand{\auxiliarycomplex}[2]%
    {\setcounter{@a}{#1}\setcounter{@b}{#2}
    \setcounter{@x}{2*\value{@a}}
    \lambda^2
    \ifthenelse{\value{@x}=0}%
        {\relax}%
        {\ifthenelse{\value{@x}<0}%
            {\setcounter{@trace}{-\value{@x}}\setboolean{@positive}{true}}%
            {\setboolean{@positive}{false}\setcounter{@trace}{\value{@x}}}
        \sign
        \ifthenelse{\value{@trace}=1}%
            {\relax}%
            {\the@trace} %% to remove 1
        \lambda
        }
    \setcounter{@x}{{\value{@a}*\value{@a}+\value{@b}*\value{@b}}}
    + \the@x y=0
    } % ends \auxiliarycomplex

\newcommand{\bsquared}{\relax}

\newcommand{\bsquaredd}[1]{\renewcommand{\bsquared}{#1}}

\newcommand{\auxiliaryquadratic}[2]%
    {\setcounter{@a}{#1}\setcounter{@b}{#2}
    \ifthenelse{\value{@b}<0}%
        {\setcounter{@b}{-\value{@b}}}%
        {\relax}
    \setcounter{@x}{2*\value{@a}}
    \ifthenelse{\value{@a}>0}%
        {\bsquaredd{(\setcounter{@trace}{-2*\value{@a}}\the@trace)^2}}%
        {\bsquaredd{\setcounter{@trace}{-2*\value{@a}}\the@trace^2}}%
    \setcounter{@det}{{\value{@a}*\value{@a}+\value{@b}*\value{@b}}}
    \lambda = \frac{\the@x\pm\sqrt{\bsquared-4\times \the@det}}{2}
    \setcounter{@det}{4*\value{@b}*\value{@b}}
    = \frac{\the@x\pm\sqrt{-\the@det}}{2}
    } % ends \auxiliaryquadratic

\newcommand{\desolreal}[2]%
    {For the second-order differential equation,
    \[\dequreal{#1}{#2}\] the auxiliary function is
    \[\auxiliaryreal{#1}{#2}\]
    which factorizes to  \[\auxiliaryfactors{#1}{#2}\]
    and has solutions \[\lambda= #1, #2.\]
    The complementary function is \[\cfreal{#1}{#2}.\]
    } % end \desolreal

\newcommand{\cfcomplex}[2]{%
    \setcounter{@b}{#2}
    \ifthenelse{\value{@b}<0}%
        {\setcounter{@b}{-\value{@b}}}%
        {\relax}
    y=\ifthenelse{#1=0}%
        {\relax}%
        {\mathrm{e}^{#1 x}\left(}A\cos(\the@b x) + B\sin(\the@b x)
        \ifthenelse{#1=0}{\relax}{\right)}
    } % end \cfcomplex

\newcommand{\desolcomplex}[2]%
    {For the second-order differential equation,
    \[\dequcomplex{#1}{#2}\]
    the auxiliary function is
    \[\auxiliarycomplex{#1}{#2}\]
    which does not factorize, and so, using the quadratic formula solution,
   the solutions are given by
    \[\auxiliaryquadratic{#1}{#2};\] this gives the roots
    \[\lambda= #1\pm
        \ifthenelse{#2<0}%
            {\setcounter{@x}{-#2}\the@x}%
            {#2}
        \,\mathrm{i}.
    \]
    The complementary function is \[\cfcomplex{#1}{#2}.\]
    } % end \desolcomplex
```

*The input.* As an example the following input (with horizontal rules to separate out the solutions from the questions):

```
\usepackage{second_order.des.sty}
\begin{document}
{\large Solve the following differential equation:}
\[\dequreal{-3}{4}\]
\emph{The solution is} \hrulefill\par
\desolreal{-3}{2}\\\emph{End of solution} \hrulefill\par
{\large Solve the following differential equation:}
\[\dequcomplex{-3}{4}\] \emph{The solution is}  \hrulefill\par
\desolcomplex{-3}{4}\\\emph{End of solution} \hrulefill
\end{document}
```

produces the output in Figure 3.



Solve the following differential equation:

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - \frac{\mathrm{d}y}{\mathrm{d}x} - 12y = 0$$

*The solution is*

For the second-order differential equation,

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - \frac{\mathrm{d}y}{\mathrm{d}x} - 12y = 0$$

the auxiliary function is

$$\lambda^2 - \lambda - 12 = 0$$

which factorizes to

$$(\lambda + 3)(\lambda - 4) = 0$$

and has solutions

$$\lambda = -3, 4.$$

The complementary function is

$$y = A\mathrm{e}^{-3x} + B\mathrm{e}^{4x}.$$

*End of solution*

Solve the following differential equation:

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + 6\frac{\mathrm{d}y}{\mathrm{d}x} + 25y = 0$$

*The solution is*

For the second-order differential equation,

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + 6\frac{\mathrm{d}y}{\mathrm{d}x} + 25y = 0$$

the auxiliary function is

$$\lambda^2 + 6\lambda + 25y = 0$$

which does not factorize, and so, using the quadratic formula solution, the solutions are given by

$$\lambda = \frac{-6 \pm \sqrt{6^2 - 4 \times 25}}{2} = \frac{-6 \pm \sqrt{-64}}{2};$$

this gives the roots

$$\lambda = -3 \pm 4\,\mathrm{i}.$$

The complementary function is

$$y = \mathrm{e}^{-3x}\left(A\cos(4x) + B\sin(4x)\right).$$

*End of solution*

Figure 3. Example of second-order differential equations and their solutions

## Cancelling fractions

Sometimes it is useful to write down the numerator and the denominator of a fraction, and for the common factors to be removed, so that the numerator and denominator are relatively prime.

In this package, the cancellation is done automatically, reducing the numerator and denominator to being relatively prime to each other. If the denominator is 1, then a whole number is printed; it also takes care of the signs.

The programming is semi-efficient; efficient in that it finds the minimum of the denominator and numerator, and finds factors up to the square root of this minimum; inefficient in that it tries every number between 2 and this minimum value, so there is some duplication (e.g. it tries 6 as a factor, after it has tried both 2 and 3).

*Coding.*

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]

\RequirePackage{ifthen,calc}
```

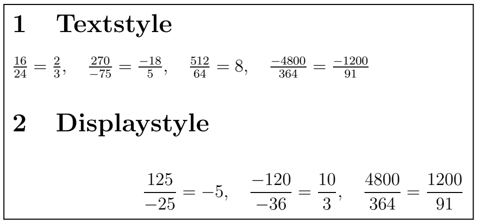
```
\newcounter{@a}\newcounter{@b}
\newcounter{@min}\newcounter{@i}
\newcounter{@c}\newcounter{@d}
\newcounter{@temp}

\newboolean{@minus}
\newboolean{@less}
\newboolean{@found}
\setboolean{@minus}{false}
\newboolean{@factor}
\setboolean{@factor}{true}
\newboolean{cancellation} % flag to tell user whether
                          % a fraction has been simplified

\newcommand{\Isfactor}[2]%
    {\setcounter{@c}{#1}\setcounter{@d}{#2}
    \setcounter{@temp}{(\value{@c}/\value{@d})*\value{@d}}
    \ifthenelse{\value{@temp}=\value{@c}}%
        {\setboolean{@factor}{true}}%
        {\setboolean{@factor}{false}}
    } %% ends \Isfactor


\newcommand{\Fraction}[2]%
    {\setcounter{@a}{#1}\setcounter{@b}{#2}
    \setboolean{cancellation}{false}
    \setboolean{@found}{true}
    \setboolean{@less}{true}
    \setcounter{@i}{2} %% to start the first factor
    \Findabsmin{\value{@a}}{\value{@b}}
    \whiledo{\boolean{@less}}%
        {\whiledo{\boolean{@found}}%
            {\Isfactor{\value{@a}}{\value{@i}}
            \ifthenelse{\boolean{@factor}}%
                {\Isfactor{\value{@b}}{\value{@i}}
                \ifthenelse{\boolean{@factor}}%
                    {\setcounter{@a}{\value{@a}/\value{@i}}
                    \setcounter{@b}{\value{@b}/\value{@i}}%
                    \setboolean{cancellation}{true}}%
                    {\setboolean{@found}{false}}
                }
                {\setboolean{@found}{false}}
            }
        \addtocounter{@i}{1}\setboolean{@found}{true}
        \setcounter{@temp}% %% check if less than square
            {\value{@i}*\value{@i}-\value{@min}-1}
        \ifthenelse{\value{@temp}<0}%
            {\relax}%
            {\setboolean{@less}{false}}
        }
    \ifthenelse{\boolean{@minus}}
               {\setcounter{@a}{-\value{@a}}}{\relax} %%  minus sign
    \ifthenelse{\value{@b}=1}{\the@a}{\frac{\the@a}{\the@b}}
    }  % ends \Fraction

\newcommand{\Findabsmin}[2]%% to find the minimum absolute value
                           %% of two numbers
    {\setcounter{@a}{#1}\setcounter{@b}{#2}
    \ifthenelse{\value{@a}<0}%
```

```
        {\setcounter{@a}{-\value{@a}}\setboolean{@minus}{true}}%
        {\setboolean{@minus}{false}}
    \ifthenelse{\value{@b}<0}%
        {\setcounter{@b}{-\value{@b}}
        \ifthenelse{\boolean{@minus}}%
            {\setboolean{@minus}{false}}
            {\setboolean{@minus}{true}}}%
        {\relax}
    \ifthenelse{\value{@a}<\value{@b}}%
        {\setcounter{@min}{\value{@a}}}%
        {\setcounter{@min}{\value{@b}}}
    }  %% ends \Findabsmin
```

*Example of input and output.* For this input:

```
\usepackage{fractioncancellation}
\begin{document}
\section{Textstyle}
    $\frac{16}{24}=\Fraction{16}{24},
     \quad \frac{270}{-75}=\Fraction{270}{-75},
     \quad \frac{512}{64}=\Fraction{512}{64},
     \quad \frac{-4800}{364}= \Fraction{-4800}{364}
    $
\section{Displaystyle}
    \[\frac{125}{-25}= \Fraction{125}{-25},
      \quad \frac{-120}{-36}= \Fraction{-120}{-36},
      \quad \frac{4800}{364}= \Fraction{4800}{364}
    \]
\end{document}
```

the output is shown in Figure 4.



1 Textstyle

$\frac{16}{24}=\frac{2}{3}, \quad \frac{270}{-75}=\frac{-18}{5}, \quad \frac{512}{64}=8, \quad \frac{-4800}{364}=\frac{-1200}{91}$

2 Displaystyle

$$\frac{125}{-25}=-5, \quad \frac{-120}{-36}=\frac{10}{3}, \quad \frac{4800}{364}=\frac{1200}{91}$$

Figure 4. Example of fraction cancellation

## Giving marks and comments on a student's script

I wanted to create a style sheet for giving comments and marks on a student's script. The marks for each section are known in advance, so the LaTeX package checks that the marks given do not exceed the maximum for that part, gives a comment if no marks have been given for a part, and adds up the marks for each section, and ultimately for the whole assignment.

*Coding.*

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{msxr209tma02}[2007/12/08 v1.1 (John Trapp)]
\LoadClass[fleqn,12pt]{article}
\RequirePackage{fancyhdr}
\RequirePackage{graphicx}
\RequirePackage{amsmath}
\RequirePackage[a4paper]{geometry}%
```

```
\RequirePackage{calc}
\RequirePackage[pageshow]{supertabular}
\RequirePackage{varioref}

\def\lastpage@putlabel{\addtocounter{page}{-1}%
   % borrowed from the lastpage package
   \immediate\write\@auxout{\string
   \newlabel{LastPage}{{}{\thepage}}}%
   \addtocounter{page}{1}}

\AtEndDocument{\comments%
   \clearpage\lastpage@putlabel}%


\newcommand{\@firstname}{\relax}
\newcommand{\@surname}{\relax}
\newcommand{\@studentpin}{\relax}
\newcommand{\@tutor}{\relax}
\newcommand{\@signoff}{\relax}
%%
\addtolength{\parskip}{1ex}

\setlength{\headheight}{15pt}

\newcommand{\firstnameis}[1]{\renewcommand{\@firstname}{#1}}
\newcommand{\surnameis}[1]{\renewcommand{\@surname}{#1}}
\newcommand{\studentpinis}[1]{\renewcommand{\@studentpin}{#1}}
\newcommand{\tutoris}[1]{\renewcommand{\@tutor}{#1}}
\newcommand{\signoff}[1]{\renewcommand{\@signoff}{#1}}

%% set up the header and footer

\pagestyle{fancy}
\lfoot{}\rfoot{}                 %Empty header and footer boxes
\fancyheadoffset[L,R]{\marginparsep}
\fancyhead{} \fancyfoot{}
\lhead{\@firstname\ \@surname\ \@studentpin}
\rfoot{\today}
\lfoot{\@tutor}
\rhead{\textrm{Page\ \thepage \ of\ \pageref{LastPage}}}
%\cfoot{{\sc\name\space\pin}}
\renewcommand{\footrulewidth}{0.2pt}
\setcounter{secnumdepth}{3}

%% Set up text width and height.

\geometry{textwidth=160mm}
\geometry{textheight=250mm}
\geometry{marginparwidth=00mm}
\geometry{lmargin=20mm,marginparsep=0mm}
\geometry{includehead,reversemp}%%,includemp

\raggedbottom
\setlength{\parindent}{0pt}

\renewcommand\today{\number\day\space \ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space \number\year} %% to give the date in UK format
```

```
% Set up default comment text,  zero marks, description label and mark total

\def\commentdefault #1#2#3{\begingroup \expandafter \endgroup%
\expandafter \commentdefault@next
  \csname #1@text\expandafter\endcsname
    \csname #1@marktotal\expandafter\endcsname
    \csname #1@mark\expandafter\endcsname
    \csname #1@error\expandafter\endcsname
    \csname #1@line\endcsname
    {#2}{#3}{#1@@mark}}

\def\commentdefault@next #1#2#3#4#5#6#7#8{%
    \def #1{No comment}% set comments for subsection from supplied parameter
    \def #2{#6}% set total mark for subsection using parameter supplied
    \def #3{0}% set default mark as 0
    \def #4{No mark}% set default comment if no mark has been entered
    \def #5{#7  &  \noexpand{#1} &  \noexpand{#3}/#2 & \noexpand{#4}\\}
    \newcounter{#8}\setcounter{#8}{0}% set default mark as 0
    }

\commentdefault{definition}{3}{Description}
\commentdefault{aspect}{2}{Aspect}
\commentdefault{outline}{5}{Outline}
\commentdefault{assumptions}{7}{Assumptions}
\commentdefault{variables}{5}{Variables}
\commentdefault{formulation}{13}{Formulation}
\commentdefault{solve}{6}{Solve the model}
\commentdefault{drawgraphs}{2}{Draw graphs}
\commentdefault{deriveresults}{2}{Derive results}
\commentdefault{collectdata}{3}{Collect data}
\commentdefault{describeinwords}{7}{Describe}
\commentdefault{decisioncompare}{3}{Compare}
\commentdefault{comparereality}{6}{Compare to reality}
\commentdefault{criticizemodel}{11}{Criticize model}
\commentdefault{describerevision}{3}{Description}
\commentdefault{revision}{7}{Formulation}
\commentdefault{conclusions}{5}{Conclusions}
\commentdefault{basicpresentation}{5}{Basic}
\commentdefault{discretionary}{5}{Discretionary}

% Set up construct for comments
\def\comment #1#2#3%
{\begingroup \expandafter \endgroup%
\expandafter \comment@next
    \csname #1@text\expandafter\endcsname
    \csname #1@marktotal\expandafter\endcsname
    \csname #1@mark\expandafter\endcsname
    \csname #1@error\endcsname
    {#2}{#1@@mark}{#3}}

    \def\comment@next #1#2#3#4#5#6#7{%
    \def #1{#5}% \wibble@text
    \def #3{#7}% \wibble@mark
    \def #4{\ifnum #7 > #2 {Error} \else \relax \fi}% \wibble@error
    \setcounter{#6}{#7}%
}

\parindent 0pt

\newcounter{@onemark}
```

```
\newcounter{@twomark}
\newcounter{@threemark}
\newcounter{@fourmark}
\newcounter{@fivemark}
\newcounter{@sixmark}
\newcounter{@sevenmark}
\newcounter{@eightmark}

\newcounter{@total}
\setcounter{@total}{100}
\renewcommand{\arraystretch}{1.5}
%\shrinkheight{-1cm}

\newcommand{\@overallcomments}{No overall comments}
\newcommand{\overallcomments}[1]{\renewcommand{\@overallcomments}{#1}}

\newcommand{\@postscript}{\relax}
\newcommand{\postscript}[1]{\renewcommand{\@postscript}{#1}}

\newcommand{\subheading}[1]{\multicolumn{3}{l}{\bf #1}& \\}
\newcommand{\subtotal}[1]{%
             &\multicolumn{2}{r}{Sub-total}&{\large \bf #1}\\ \hline}

\tabletail{\multicolumn{4}{r}{\em continued on next page}\\}
%% to show continuation at bottom of each page
\tablelasttail{&&&\\} % to avoid continuation message at end of table

\newcommand{\comments}%% The main command to assemble all the material.
{Dear \@firstname,\par \@overallcomments
\begin{center}
\@signoff\\[1in]
\end{center}
\begin{flushright}
 \@tutor\\[1cm]
\end{flushright}
\textbf{\large Comments and marks on each section}\par%

%% set up the counters for the marks at each subsection, and for the total

\setcounter{@onemark}{\value{definition@@mark}+\value{aspect@@mark}}
\setcounter{@twomark}{\value{outline@@mark}+\value{assumptions@@mark}+
 \value{variables@@mark}+\value{formulation@@mark}}
\setcounter{@threemark}{\value{solve@@mark}+\value{drawgraphs@@mark}+
 \value{deriveresults@@mark}}
\setcounter{@fourmark}
  {\value{collectdata@@mark}+\value{describeinwords@@mark}+
  \value{decisioncompare@@mark}}
\setcounter{@fivemark}
  {\value{comparereality@@mark}+\value{criticizemodel@@mark}}
\setcounter{@sixmark}
  {\value{describerevision@@mark}+\value{revision@@mark}}
\setcounter{@sevenmark}{\value{conclusions@@mark}}
\setcounter{@eightmark}
  {\value{basicpresentation@@mark}+\value{discretionary@@mark}}
\setcounter{@total}{\value{@onemark}+\value{@twomark}+
 \value{@threemark}+\value{@fourmark}+\value{@fivemark}+
 \value{@sixmark}+\value{@sevenmark}+\value{@eightmark}}


%% construct the table of comments and marks
\begin{supertabular*}{\linewidth}{lp{4.5in}rr}\hline
\subheading{Specify the purpose of the mathematical model}
    \definition@line \\
    \aspect@line\\
 \subtotal{\the@onemark}
\subheading{Create the  model}
    \outline@line \\
    \assumptions@line \\
    \variables@line \\
    \formulation@line \\
    \subtotal{\the@twomark}
\subheading{Do the mathematics}
        \solve@line \\
        \drawgraphs@line \\
        \deriveresults@line \\
    \subtotal{\the@threemark}
\subheading{Interpret the results}
        \collectdata@line \\
        \describeinwords@line \\
        \decisioncompare@line \\
    \subtotal{\the@fourmark}
\subheading{Evaluate the model}
        \comparereality@line \\
        \criticizemodel@line \\
    \subtotal{\the@fivemark}
\subheading{Revise the model}
        \describerevision@line \\
        \revision@line \\
    \subtotal{\the@sixmark}
\subheading{Conclusions}
        \conclusions@line \\
    \subtotal{\the@sevenmark}
\subheading{Presentation}
    \basicpresentation@line \\
  \discretionary@line\\
 \subtotal{\the@eightmark}
    &\multicolumn{2}{r}{\textbf{\large Total}} &
    \textbf{\large \the@total/100}\\ \cline{3-4}
  \end{supertabular*}
\@postscript}
```

*The template to be populated by the tutor.*

```
\documentclass{msxr209tma02}

\signoff{Best wishes,} %% Whatever closing (centred)
% statement that one wants
\tutoris{John Trapp\\00953618} %% In a flushright environment,
% so can have multiple lines

\firstnameis{} %% student first name
\surnameis{} %% student surname
\studentpinis{} %% student PIN

\begin{document}

\overallcomments{} %% overall comments before detailed comments and marks,
% closed by \signoff and tutor’s name

%% Write the comments for the subsection as the second parameter,
% and the marks as the third parameter; the first parameter
% being the subheading to which it refers. The total marks for
% that section are commented out at the end, as a reminder.

\comment{definition}{}{}%{3}
\comment{aspect}{}{}%{2}
\comment{outline}{}{}%{5}
\comment{assumptions}{}{}%{7}
\comment{variables}{}{}%{5}
\comment{formulation}{}{}%{13}
\comment{solve}{}{}%{6}
\comment{drawgraphs}{}{}%{2}
\comment{deriveresults}{}{}%{2}
\comment{collectdata}{}{}%{3}
\comment{describeinwords}{}{}%{7}
\comment{decisioncompare}{}{}%{3}
\comment{comparereality}{}{}%{6}
\comment{criticizemodel}{}{}%{11}
\comment{describerevision}{}{}%{3}
\comment{revision}{}{}%{7}
\comment{conclusions}{}{}%{5}
\comment{basicpresentation}{}{}%{5}
\comment{discretionary}{}{}%{5}

\postscript{} %% To add anything after the table of comments and marks.

%% All these comments and marks are assembled outwith this file.
\end{document}
```

*An example in use.* To reduce the length, the comments are minimal; I have also included examples of marks not being given for a section, and too many marks in a sub-part.

With this input

```
\documentclass{msxr209tma02}

\signoff{Best wishes,} %% Whatever closing statement (centred) that one wants
\tutoris{John Trapp\\00953618} %% In a flushright environment,
                              %% so can have multiple lines

\begin{document}

\firstnameis{Andrew}
\surnameis{Aardvark}
\studentpinis{12345678}

\overallcomments{You have written an excellent modelling report,
  and you seem to have a good idea of how to model.
  I particularly liked your presentation of the graphs and figures.}

\comment{definition}{You have stated the problem very well\ldots}{2}%{3}

\comment{aspect}{You had some good ideas,
  but I thought that you should have mentioned \ldots
  }{4}%{2}

\comment{outline}{A very good outline of the problem,
  and gives me a good idea of what you will be doing}{5}%{5}

\comment{assumptions}{You seem to have covered all
  the assumptions}{6}%{7}
```

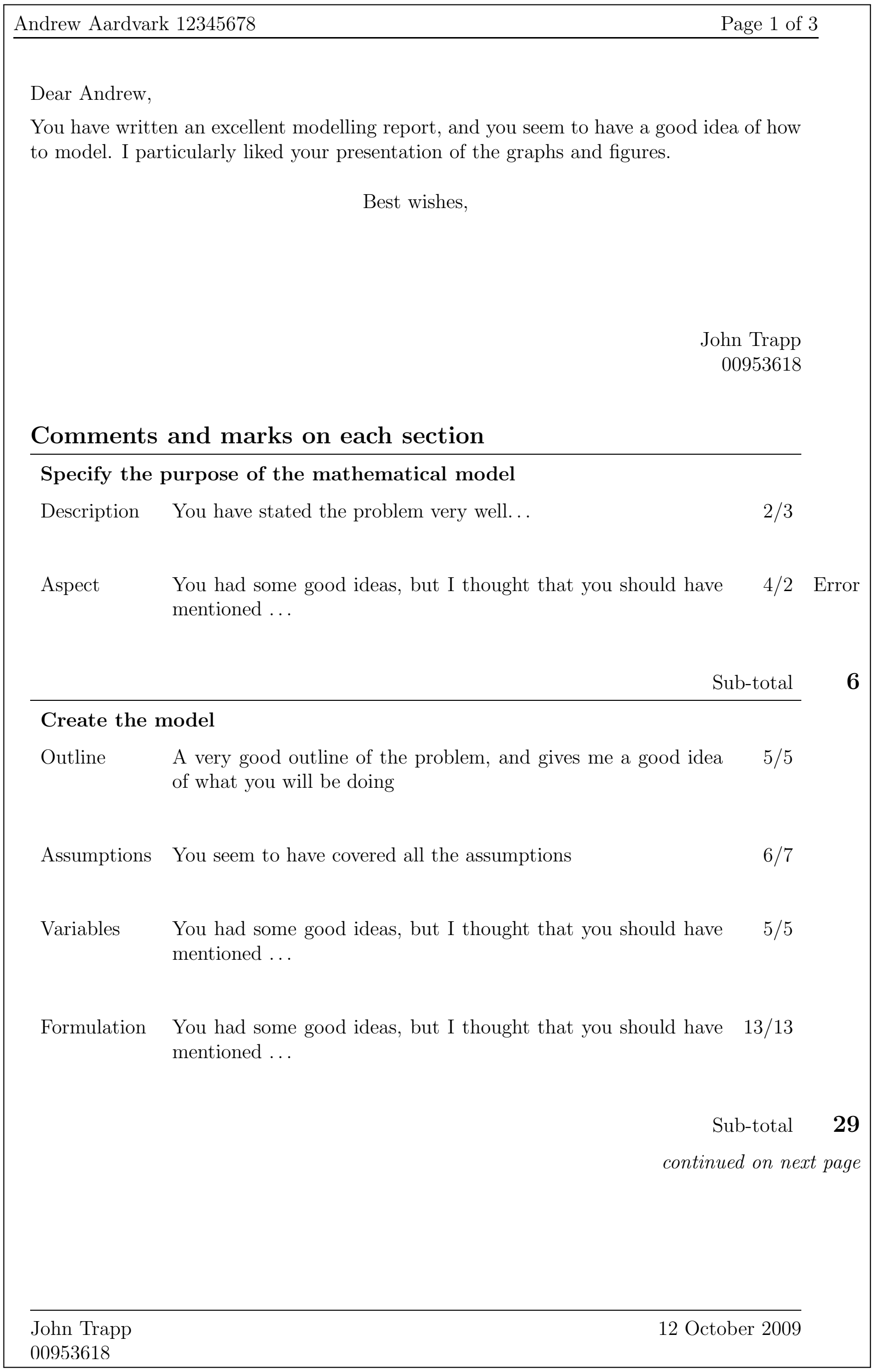
```
\comment{variables}{You had some good ideas,
  but I thought that you should have mentioned \ldots
  }{5}%{6}

\comment{formulation}{You had some good ideas,
 but I thought that you should have mentioned \ldots }{13}%{13}
\comment{solve}{Ooooh}{4}
\comment{drawgraphs}{Oh good}{2}%{2}
\comment{deriveresults}{Good picture}{2}%{2}
\comment{collectdata}{Useful data that you have collected}{2}%{3}
\comment{describeinwords}{Good description of the solution }{6}%{7}
\comment{decisioncompare}{Not really come to grips with this}{1}%{3}
\comment{comparereality}{Sort of on the right lines}{4}%{6}
\comment{criticizemodel}{Very good criticism,
  and useful comments; however, you should refer
  to your evaluation and try to assess which
  revision will help you to obtain a better model}{9}%{11}

\comment{conclusions}{Uggh}{4}
\comment{basicpresentation}{Very good presentation;
  you have been very clear in your description,
  and your use of Mathcad is exemplary}{7}%{5}

\comment{discretionary}{You had some good ideas,
  but I thought that you should have mentioned \ldots }{4}%{5}

\postscript{This is a section just in case one wants
  to add something after the detailed marks and comments}

%% All these comments and marks are assembled outwith this file.
\end{document}
```

the output is given in Figures 5, 6 and 7.

## Notes

[1] Note to non-mathematicians: these definitions are useful, and not arbitrarily plucked out of the ether.

John Trapp



Andrew Aardvark 12345678 Page 1 of 3

Dear Andrew,

You have written an excellent modelling report, and you seem to have a good idea of how to model. I particularly liked your presentation of the graphs and figures.

Best wishes,

John Trapp
00953618

## Comments and marks on each section

**Specify the purpose of the mathematical model**

| | | | |
|---|---|---|---|
| Description | You have stated the problem very well. . . | 2/3 | |
| Aspect | You had some good ideas, but I thought that you should have mentioned . . . | 4/2 | Error |
| | | Sub-total | **6** |

**Create the model**

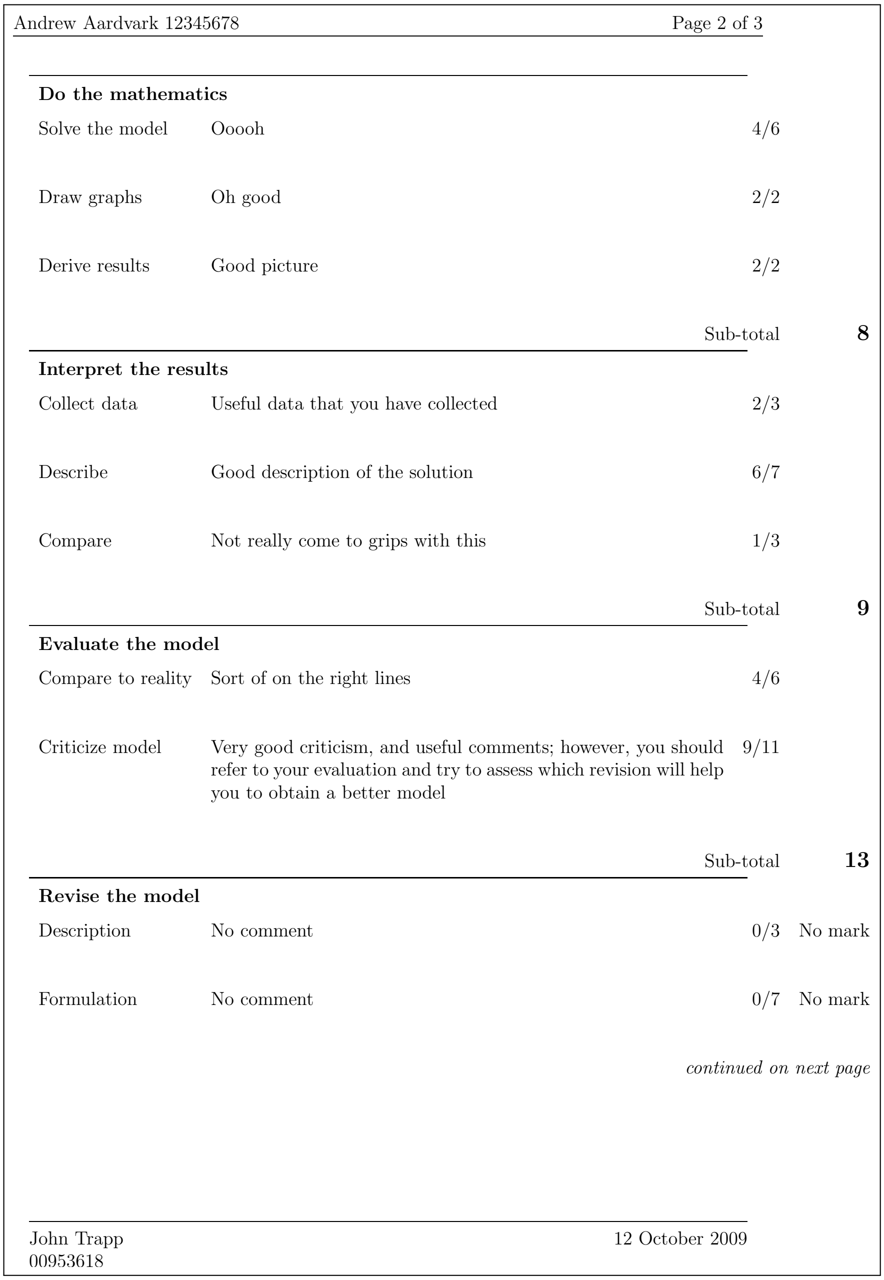
| | | | |
|---|---|---|---|
| Outline | A very good outline of the problem, and gives me a good idea of what you will be doing | 5/5 | |
| Assumptions | You seem to have covered all the assumptions | 6/7 | |
| Variables | You had some good ideas, but I thought that you should have mentioned . . . | 5/5 | |
| Formulation | You had some good ideas, but I thought that you should have mentioned . . . | 13/13 | |
| | | Sub-total | **29** |

*continued on next page*

John Trapp
00953618

12 October 2009

Figure 5. First page of marking example



Andrew Aardvark 12345678 Page 2 of 3

**Do the mathematics**

| | | | |
|---|---|---|---|
| Solve the model | Ooooh | 4/6 | |
| Draw graphs | Oh good | 2/2 | |
| Derive results | Good picture | 2/2 | |
| | Sub-total | | **8** |

**Interpret the results**

| | | | |
|---|---|---|---|
| Collect data | Useful data that you have collected | 2/3 | |
| Describe | Good description of the solution | 6/7 | |
| Compare | Not really come to grips with this | 1/3 | |
| | Sub-total | | **9** |

**Evaluate the model**

| | | | |
|---|---|---|---|
| Compare to reality | Sort of on the right lines | 4/6 | |
| Criticize model | Very good criticism, and useful comments; however, you should refer to your evaluation and try to assess which revision will help you to obtain a better model | 9/11 | |
| | Sub-total | | **13** |

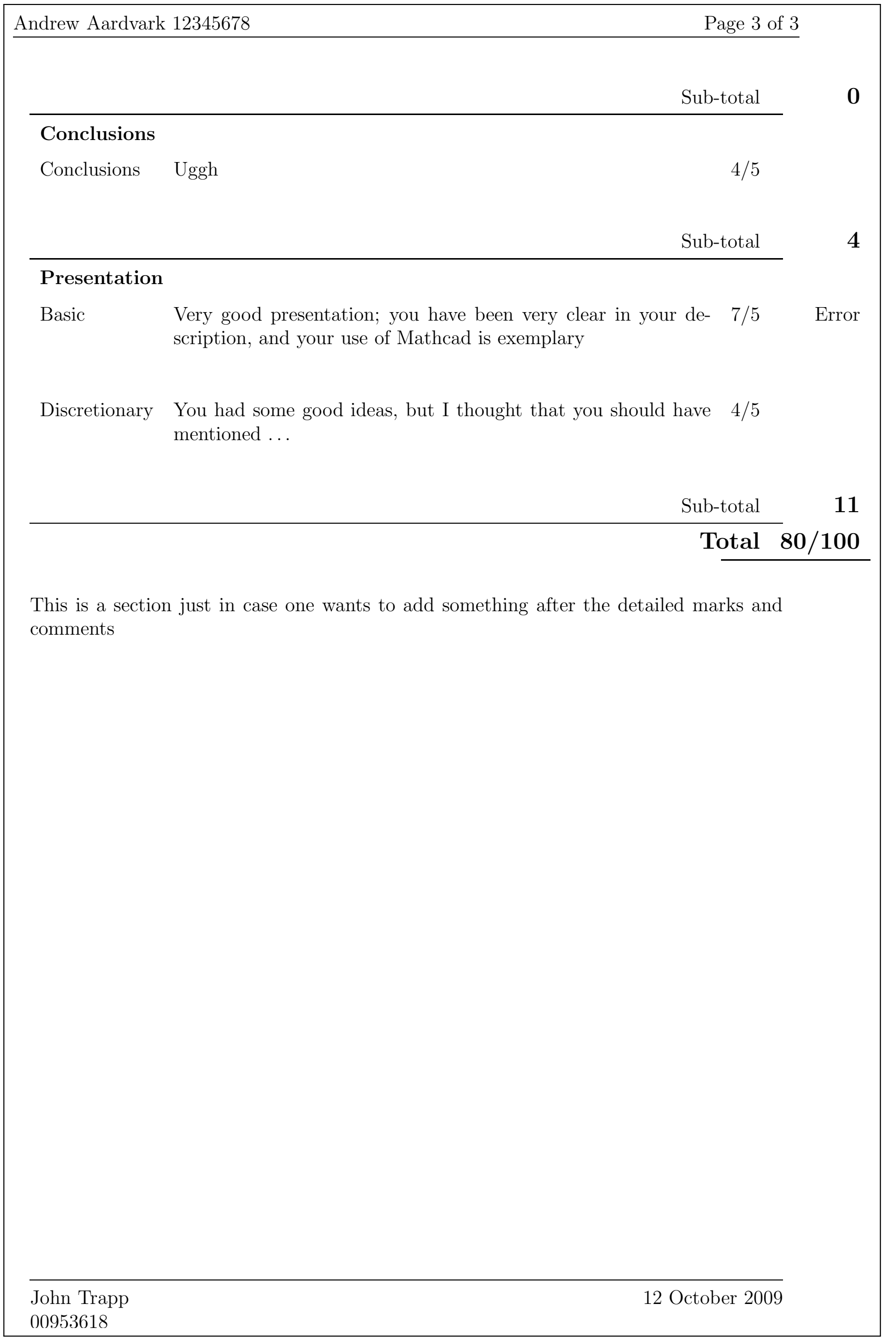
**Revise the model**

| | | | |
|---|---|---|---|
| Description | No comment | 0/3 | No mark |
| Formulation | No comment | 0/7 | No mark |

*continued on next page*

John Trapp 12 October 2009
00953618

Figure 6. Second page of marking example



Andrew Aardvark 12345678 Page 3 of 3

| | | | |
|---|---|---|---|
| | | Sub-total | **0** |
| **Conclusions** | | | |
| Conclusions | Uggh | 4/5 | |
| | | Sub-total | **4** |
| **Presentation** | | | |
| Basic | Very good presentation; you have been very clear in your description, and your use of Mathcad is exemplary | 7/5 | Error |
| Discretionary | You had some good ideas, but I thought that you should have mentioned . . . | 4/5 | |
| | | Sub-total | **11** |
| | | **Total** | **80/100** |

This is a section just in case one wants to add something after the detailed marks and comments

John Trapp
00953618

12 October 2009

Figure 7. Last page of marking example

# Experiences typesetting mathematical physics

**Abstract**
Twenty years ago, the author was just about to start his university studies in math and physics. A year or so later, he not only discovered a fascinating program called TEX, but he also got involved in a project of typesetting a series of lecture notes which eventually became book manuscripts for a complete course in theoretical physics. In the end, he spent about seven years working on typing, editing, revising, and formatting more than 2500 book pages containing a large amount of math.
While there are many experiences from such a project one could talk about, ranging from issues of project management to document design and layout, this talk will focus on two specific topics: adapting LATEX to deal with the specific requirements of mathematical notation in physics and fine-tuning the appearance of math formulas.

**Keywords**
math typesetting, physics, notation

Given the conference motto of educational uses of TEX, this paper is based on the author's personal experiences of typesetting a series of lecture notes which eventually became a series of textbooks for a complete course in theoretical physics [1, 2, 3, 4, 5, 6].

## Introduction: How I got started typesetting physics

When I started my university studies in math and physics in 1989, I did not know anything about TEX or about typesetting in general. However, I quickly noticed that there was a vast difference in quality of typeset material when it came to math formulas. As it turned out, TEX was already being used by several of the math and physics departments for routine typesetting tasks (such as the weekly handouts of homework exercises to students), while other departments were still using a mathematical typewriter.

I first got to know TEX in the summer of 1990 through some of my friends, who had gotten a TEX distribution from the staff of the university computer center. Unfortunately, I did not have a decent computer to run it until a few months later, so instead of jumping into using TEX right away, I started by reading several books about it, including *The TEXbook* itself. When I eventually got started some time later that year, I soon began looking under the hood, trying to understand how the various bits and pieces of a TEX distribution fit together.

I got started with typesetting physics in early 1991, when the professor who held the theoretical physics course asked for volunteers to help him typeset his lecture notes, so they could be printed and handed out to students. After working on this project for several months on a voluntary basis, I was eventually employed as a student assistant from 1991 to 1995, getting paid to work on editing and typesetting lecture notes with TEX. After finishing my diploma and staying on to work on a PhD project at the same department, I continued to work on book manuscripts until the end of 1998.

From 1991 to 1993 the first editions were prepared for the principal courses in theoretical physics, consisting of the volumes on mechanics, electrodynamics, quantum mechanics, thermodynamics, and quantum field theory.

From 1994 to 1996 some additional volumes were added for special courses in general relativity, cosmology, and elementary particle theory.

Over time, as I learned more about typography and about typesetting rules for math, I revised the layout and the macros several times and began to write my own document classes and macro packages, switching from LaTeX 2.09 in 1991 to NFSS2 in 1993 and to LaTeX $2_\varepsilon$ in 1994.

Using a full-page layout on A4 paper and Computer Modern fonts at 11 pt type size, the series amounted to 2000 printed pages in total, consisting of six volumes of 300 to 350 pages each and two volumes of 100 pages each.

From 1996 to 1999 the second editions of several volumes were prepared, when the courses were held again for the next generation of students. By this time, a publisher had been found who was interested in publishing the lecture notes as textbooks, but since they already had another series of theoretical physics lectures in their program, it was decided to run several volumes together as an omnibus edition of two large volumes of 1200 to 1300 pages each.

For the publisher edition, the layout was revised using Times and MathTime fonts at 10 pt type size on a smaller page size and using a more compact layout, arriving at about 2500 book pages in total. At the same time the macros for math typesetting were revised once more, taking advantage of some of the extra fonts in the MathTime Plus distribution such as an upright Greek font.

The first volume of the omnibus edition finally appeared in 1999 [1], shortly after I had left university, followed by the second volume in 2004 [2], shortly after my professor retired. (In the meantime, second editions of the remaining volumes were prepared from 1999 to 2003, when the courses were held again.)

After the series was finally completed and deemed successful by the publisher, individual volumes [3, 4, 5, 6] were also published separately in recent years.

In summary, working on this project for seven years from 1991 to 1998 was an interesting experience in a wide range of topics, ranging from project organization to support and maintenance of TeX installations, and from high-level document design of layout to low-level details of math typesetting.

Regarding the technical progress, there are some interesting stories to be told:

In 1991, running a complete book of 300 to 350 pages on a 16 MHz 386 PC required taking a lengthy break and occasionally resulted in crashing TeX, if you forgot to use a BigTeX and had too many labels and references. Usually, running chapters separately with `\includeonly` was the preferred way of working, but this still took several minutes per chapter for each run.

In 1998, running a combined edition of 1200 pages on a 100 MHz 486 PC was already much quicker, but it also required enlarging some parameters in `texmf.cnf` to make it work without running out of memory.

Nowadays, we have GHZ computers with GBytes of memory, and modern TeX distributions have become big enough by default, so speed and size are no longer an issue, although it still takes time to process 1200 pages.

On the other hand, getting the fine points of math typesetting right is still far from trivial, so we will concentrate on these topics in the following sections. In the next section of this paper, we will look at the difficulties of how to set up (LA)TeX for properly typesetting physics according to the standards of the field. In the final section, we will look at some examples of how to improve the appearance of math formulas and how to deal with special requirements of notation.

## Adapting (La)TeX for typesetting physics

While TeX, in general, does a very good job of typesetting math, different fields of sciences have slightly different conventions how math should be typeset, and TeX does not support all of them equally well.

By default, TeX's built-in rules for typesetting math-mode material are geared towards the conventions applicable for an American style of typesetting math, as specified by style manuals such as *The Chicago Manual of Style* [7] or the style guides of well-respected math publishers [8]. However, this is only one particular style and not everyone favors the same style. For example, the French tradition may suggest a different style and the Russian tradition yet another one.

In physics and related fields, the established conventions are specified by the handbooks of professional societies, such as the IUPAP red book [9, 10] in physics or the IUPAC green book [11, 12, 13] in physical chemistry, or by international standards such as ISO 31-11 or ISO 80000-2 [14, 15].

In essence, the most important points can be summarized as follows:

□ Symbols for physical quantities should be typeset in math italic.

□ Symbols for vectors should be typeset in bold math italic.

□ Symbols for tensors should be typeset in bold sans serif italic.

□ Symbols for physical units should be typeset in upright roman.

□ Symbols for chemical elements should be typeset in upright roman.

□ Symbols for elementary particles should be typeset in upright roman.

□ Mathematical constants (such as $\mathrm{e}$, $\mathrm{i}$, $\mathrm{\pi}$) should be upright roman.

□ Mathematical operators (such as $\mathrm{d}$, $\partial$, $\mathrm{\delta}$, $\Delta$) should be upright roman.

In theory, these rules should apply universally to typesetting physics and they should apply to all symbols without exceptions, regardless of whether they are Latin or Greek, uppercase or lowercase. In practice, however, many science publishers insist on their own style guides, which follow the official standards only to a greater or lesser degree [16, 17].

For example, upright bold might be used for vectors instead of bold italic and upright bold sans serif might be used for tensors instead of bold sans serif italic. In addition, the rules for typesetting mathematical constants and mathematical operators are often neglected.

While it is easy to summarize the rules in a few sentences, it is far from easy to implement them using a standard TeX system with a standard set of Computer Modern fonts. In fact, even some recent editions of the guidebooks typeset with TeX suffer from these limitations, so they have become less reliable than some of the earlier editions typeset with traditional methods.

Given the default setup of (La)TeX (for an American style of mathematics), we have to deal with the following inconsistencies:

□ Symbols from the Latin alphabet are typeset in math italic by default, and they can be switched to upright roman (or other alphabets).

□ Symbols from the uppercase Greek alphabet are in roman by default, and they can be switched to math italic (or other alphabets).

□ Symbols from the lowercase Greek alphabet are in math italic by default, but cannot switch families, because they are not available otherwise.

In addition, we have to deal with the following limitations of fonts:

- □ A bold math italic font (suitable for vectors) does exist in the standard set of Computer Modern fonts, but is not available as a math alphabet by default.
- □ A bold sans serif italic font (suitable for tensors) does not exist in the standard set of Computer Modern fonts, but could be substituted using non-standard fonts.
- □ An upright Greek font (suitable for elementary particles) is not available in the Computer Modern fonts and cannot easily be substituted, unless you switch to a different set of fonts.

To develop a setup suitable for typesetting physics according to the rules, the following steps must be taken:

- □ Load additional math families for bold math italic (suitable for vectors) and bold sans serif italic (suitable for tensors).
- □ Redefine the math codes of uppercase Greek letters, so that they are typeset in math italic instead of upright roman by default.
- □ Redefine the math codes of lowercase Greek letters, so that they can switch families where possible (in a controlled way).
- □ Define font switching macros for vectors, tensors, particles, and units, which prevent lowercase Greek letters from switching to math alphabets where they do not exist.
- □ Define macros for the markup of mathematical constants and operators to be set in upright roman where possible.
- □ Define macros for additional math operators and specific notations needed in physics.

## Loading math families and math alphabets

In the early years of the project, when LaTeX 2.09 was still being used, defining additional math families or font switches was not really supported by the LaTeX format, so it was impractical to do for authors of macro packages. Instead, clumsy workarounds were being used to get a bold math italic font by standard means, which involved switching to `\boldmath` from inside an `\hbox`.

It was only after the introduction of NFSS2 in 1993 and LaTeX 2ε in 1994 that it became feasible for authors of math support packages to redefine math fonts as needed. Given the LaTeX 2ε interface, loading additional math families for vectors or tensors is relatively straight-forward:

```
\DeclareSymbolFont{vectors}{OML}{cmm}{b}{it}
\DeclareSymbolFont{tensors}{OT1}{cmss}{bx}{it}
```

Similarly, font switching commands for math alphabets can be defined as follows:

```
\DeclareSymbolFontAlphabet{\mathvec} {vectors}
\DeclareSymbolFontAlphabet{\mathtens}{tensors}
```

The only problem with this setup might be that the font shape `cmss/bx/it` does not exist in CM fonts and is silently substituted by `cmss/bx/n`, which results in using upright bold sans serif for tensors as often done by publishers. (This would work better with LM fonts nowadays, but we did not have them in the 1990s.) Once these fonts and alphabets are loaded, it becomes a matter of redefining the math codes to get the symbols to appear in the proper shape by default.

## Redefining math codes

Given the default setup of (LA)TEX, the math codes for uppercase Greek letters are defined to be of type `\mathalpha` (which means: allowed to switch families), using the `operators` symbol font (upright roman) by default. To get them to use the `letters` symbol font (math italic) by default, we have to redefine the math codes as follows:

```
\DeclareMathSymbol{\Gamma}  {\mathalpha}{letters}{"00}
\DeclareMathSymbol{\Delta}  {\mathalpha}{letters}{"01}
...
\DeclareMathSymbol{\Omega}  {\mathalpha}{letters}{"0A}
```

In a more sophisticated setup, we could even make the default font configurable by a package option, using a macro to select the appropriate font. Nowadays, many support packages for math fonts tend to provide such switching mechanisms (and sometimes they also define extra macros such as `\upGamma` or `\varGamma`), but in the mid-1990s this was not the case.

For lowercase Greek letters, the situation is somewhat different. By default, the math codes are defined to be of type `\mathord` (which means: *not* allowed to switch families), using the `letters` symbol font (math italic) by default. To get them to switch to the `vectors` symbol font (bold math italic), we have to redefine the math codes using the type `\mathalpha`:

```
\DeclareMathSymbol{\alpha}  {\mathalpha}{letters}{"0B}
\DeclareMathSymbol{\beta}   {\mathalpha}{letters}{"0C}
...
\DeclareMathSymbol{\varphi} {\mathalpha}{letters}{"27}
```

## Defining font switching macros

Unfortunately, allowing lowercase Greek letters to switch families not only allows them to switch to families that exist (such as bold math italic), but also allows them to switch to families which do not exist (such as upright roman), potentially causing unexpected results.

In order to prevent such effects, we found it necessary to add some intelligence to font switching macros to check whether the argument of the font switch is a lowercase Greek letter or something else which does not require special treatment. The solution we came up with consists of the following code:

```
\newif\if@lowgreek  \newbox\@lowgreekbox

\def\@lowgreektest#1{\setbox\@lowgreekbox=\hbox{$%
  \global\@lowgreekfalse
  \ifnum\alpha>#1\else\ifnum\varphi<#1\else
  \global\@lowgreektrue\fi\fi
$}}
\def\@lowgreekswitch#1#2#3{\@lowgreektest{#1}%
  \if@lowgreek\def\next{#3}\else\def\next{#2}\fi\next{#1}}
```

In essence, we use a numeric comparison of math codes in `\@lowgreektest` to check if the argument is lowercase Greek (between `\alpha` and `\varphi`) and we use the result of this check in `\@lowgreekswitch` to choose between two alternative font commands to use for lowercase Greek letters or everything else. Given these macros, we can define font switching macros as follows:

```
\DeclareRobustCommand{\particle}[1]{%
  \@lowgreekswitch{#1}{\mathrm}{\mathnormal}}
\DeclareRobustCommand{\tens}[1]{%
  \@lowgreekswitch{#1}{\mathtens}{\mathvec}}
```

The effect should be that \particle will switch to \mathnormal (math italic) for lowercase Greek and to \mathrm (upright roman) for everything else. Similarly, \tens will switch to \mathvec (bold math italic) for lowercase Greek and to \mathtens (bold sans serif italic or upright) for everything else.

For vectors, no special font switching macros are needed, if the default style of bold math italic is used. However, if a publisher style prefers to use \mathbf (upright bold) for vectors, a similar switch would also be needed:

```
% if publisher insists on \mathbf (upright bold) for vectors
\DeclareRobustCommand{\vec}[1]{%
  \@lowgreekswitch{#1}{\mathbf}{\mathvec}}

% otherwise, if using \mathvec (bold math italic) for vectors
\let\vec=\mathvec
```

As will be obvious from this discussion, setting up the appropriate font families to satisfy the requirements of physics could be relatively straight-forward if all the fonts were math fonts providing the full range of Greek and Latin alphabets and not just text fonts providing only a subset. However, given the traditional setup of (LA)TEX and its limitations, such kinds of workarounds seem to be unavoidable to provide for font substitutions when symbols are not available.

## Defining logical markup for physics

As shown above, we have redefined \vec as a font switch, thereby overwriting the default definition as a math accent in (LA)TEX. Following the principle of logical markup, we have chosen to use \vec and a number of similar macros to mark up the meaning of entities in physical notation consistently, regardless of how they will be represented depending on package options of a macro package.

In summary, we have defined macros such as \vec (vectors), \tens (tensors), \text (textual indices), \units (physical units), \chem (chemical elements), or \particle (elementary particles), as summarized in the following table.

| markup | purpose | font | scope |
|---|---|---|---|
| none (default) | physical quantities | \mathnormal | Latin and Greek |
| \units | physical units | \mathrm | Latin mostly |
| \text | textual material | \mathrm | Latin only |
| \chem | chemical elements | \mathrm | Latin only |
| \particle | elementary particles | see above | Latin and Greek |
| \vec | vector quantities | see above | Latin and Greek |
| \tens | tensor quantities | see above | Latin and Greek |

In many cases, these macros are simply implemented as font switches if no special provisions for font substitutions are needed. In other cases, they could be either defined as font switches or as math accents with different representations depending on the preferred style of the author or publisher.

When using such markup, it is important to ensure a certain level of discipline and consistency when typing formulas. For example, there may be a difference between typing \dot\vec{x}} and \vec{\dot{x}}, especially if font substitutions involving \@lowgreektest and \@lowgreekswitch are used.

In general, macros which are known to be math accents (such as \dot or \hat) should be applied on the outer level to macros which could be implemented either as math accents or font switches (such as \vec or \tens). In addition, such macros should usually be applied to individual symbols only, so you should make sure to type \vec{E} \times \vec{B} instead of taking shortcuts such as \vec{E \times B}.

## Defining markup for mathematical constants and operators

Besides the requirements for physical notation discussed so far, there are also requirements for mathematical constants (such as $\mathrm{e}$, $\mathrm{i}$, $\pi$) and for mathematical operators (such as $\mathrm{d}$, $\partial$, $\delta$, $\Delta$), which are supposed to be typeset in upright roman, if the rules for typesetting physics are to be implemented to the full extent.

In practice, however, these rules are often neglected or only partially implemented for various reasons. One obvious problem is the lack of suitable fonts, making it difficult or impossible to typeset lowercase Greek letters (such as $\delta$) and certain other symbols (such as $\partial$) in upright roman.

Another problem is the question of markup, which requires careful attention by authors or editors to make sure that individual letters (such as $\mathrm{d}$, $\mathrm{e}$, $\mathrm{i}$) are typeset in the proper font depending on the context where they are used.

In particular, not every occurrence of these letters represents a mathematical constant or operator which should be typeset in upright roman. It always depends on the context: $d$ could also be used as a distance, $e$ could be used as the charge of an electron, and $i$ could be used as a component index. In such cases, the letters would be considered physical quantities and typeset in math italic.

It is only when $\mathrm{d}x$ is used as a differential operator, $\mathrm{e}^x$ is used as an exponential term, and $\mathrm{i}$ is used as the imaginary unit, that these letters would be considered mathematical constants or operators and typeset in upright roman.

Concerning suitable markup, there does not seem to be any agreement between authors of macro packages how these entities should be represented. Obviously, it would be inconvenient for authors to type `\mathrm` all over the place and this would also violate the principle of logical markup. On the other hand, defining the shortest possible macros (such as `\d`, `\e`, `\i`) conflict with standard TeX macros for the under-dot and the dotless-i in text mode. (Alternatively, one might prefer to use `\dd`, `\ee`, `\ii` for these macros, which are still easy enough to type, but will not cause conflicts with existing macros.)

In our case, we have indeed chosen the shortest possible macros, but using a slightly more sophisticated setup to limit the redefinitions to math mode while retaining the original definitions in text mode (assuming that `\@@d` and `\@@i` are used to capture the original meanings):

```
\DeclareRobustCommand{\d}{%
  \relax\ifmmode\mathrm{d}\else\expandafter\@@d\fi}
\DeclareRobustCommand{\e}{%
  \relax\ifmmode\mathrm{e}\else\error\fi}
\DeclareRobustCommand{\i}{%
  \relax\ifmmode\mathrm{i}\else\expandafter\@@i\fi}
```

Yet another approach might be to use explicit markup only for some letters which will be used in different contexts (such as $e$, $i$), while using a special setup of math codes to achieve global changes for other letters which will always be used in the same context (such as $\mathrm{d}$). For example, if the latter is only used as a mathematical operator, its math codes might just as well be redefined globally to set it in upright roman by default (using the `operators` font), thereby eliminating the need for authors to apply any special markup:

```
\DeclareMathSymbol{d}{\mathalpha}{operators}{‘d}
```

Using this approach, authors could simply type `$dx$` without being aware of the fine details, while getting the proper rendering $\mathrm{d}x$ instead of $dx$ automatically. Obviously, the same approach could also be used to set the preferred shape of other symbols which are almost always used as math operators (such as $\partial$, $\delta$, $\Delta$), provided that suitable fonts are available.

## Summary and conclusions on setting up math fonts

In the previous sections, we have discussed a low-level approach to set up (La)TeX for typesetting physics, based on the state of the art of the mid-1990s. At that time, there were relatively few choices of math fonts available besides Computer Modern, Concrete, or Euler; and math font support packages for LaTeX usually provided little else but the essential setup. When we later switched to the commercial MathTime font set, relatively little changed in the basic setup, except that we created a custom virtual font to take advantage of the upright Greek font in the MathTime Plus edition.

In recent years, many more choices of math fonts have become available (such as txfonts, pxfonts, fourier, mathdesign), often providing useful additions such as upright or bold fonts; and several math font support packages for LaTeX have started to provide various switching options (such as slantedgreek or uprightgreek) to configure the preferred style of rendering uppercase and lowercase Greek letters. Unfortunately, there is no universal interface for all such packages, and the details of what is available and/or configurable still vary quite a lot.

On another front of development, several packages have appeared which deal with the specifics of notations, including a recently released isomath package, which acts as a meta-package on top of other packages. Unfortunately, none of these packages covers all the details we encountered in our projects, and there is still no comprehensive package for a physics environment.

Looking towards the future, many recent developments of new TeX engines have concentrated on providing support for Unicode and OpenType font technology, including support for OpenType math fonts. Given these developments, the traditional concept of alphabetic symbols, which may switch between different font families, has to be reconsidered fundamentally.

In a traditional 8-bit TeX engine, multiple math fonts are loaded into different font families, each using the same slots for Latin or Greek alphabets to be rendered in different styles (such as roman, italic, bold, bold italic, script, etc.). In a Unicode setup, however, there will be only a single math font, using a different range of slots for each style of alphabets, so the font switching commands would actually have to switch between different slots, as shown in the following table:

| font style | Latin alphabet slots | Greek alphabet slots |
|---|---|---|
| upright roman | U+0041 ... U+005A | U+0391 ... U+03A9 |
| | U+0061 ... U+007A | U+03B1 ... U+03C9 |
| math italic | U+1D400 ... U+1D433 | U+1D6A8 ... U+1D6E1 |
| upright bold | U+1D434 ... U+1D467 | U+1D6E2 ... U+1D71B |
| bold math italic | U+1D468 ... U+1D49B | U+1D71C ... U+1D755 |
| bold sans upright | U+1D5D4 ... U+1D607 | U+1D756 ... U+1D79F |
| bold sans italic | U+1D63C ... U+1D66F | U+1D790 ... U+1D7C9 |

On the technical side, this method of switching alphabetic symbols between different Unicode slots will cause a lot of overhead to the implementation of font switching macros in support packages such as unicode-math, but fortunately these macros will have to be implemented only once and for all, since the same layout will be applicable to all forthcoming OpenType math fonts.

Regarding the range of available alphabets, Unicode math provides not only the above-mentioned alphabets, but also several more (including script, bold script, fraktur, bold fraktur, or blackboard bold). Moreover, the range of symbols includes not only the full uppercase and lowercase Latin and Greek alphabets, but also some letter-like operators such as $\boldsymbol{\nabla}$ and $\partial$. Given all these provisions, developing a full setup for typesetting physics should finally become much easier.

## Improving the appearance of math formulas

Many users of TeX in academic research have spent little time on learning TeX, relying only on introductory (La)TeX books or popular online guides such as lkurz or lshort. While these guides are useful to get started quickly, they usually do not spend much room on explaining the fine points of typesetting math. By contrast, *The TeXbook* devotes four chapters on the topic of math typesetting including a full chapter on the fine points of typing math formulas.

Before making any attempts to improve the appearance of math formulas, it is important to make sure that formulas are properly coded, and it may be worthwhile to refresh your reading of *The TeXbook* for this purpose.

### Avoiding common mistakes

Some common mistakes by inexperienced users include forgetting to type the proper symbol (such as using <= or << instead of `\le` or `\ll` for $\le$ or $\ll$) or failing to note the difference between similar symbols (such as using < and > instead of `\langle` and `\rangle` as delimiters for $\langle x \rangle$).

Another common mistake is forgetting to define suitable macros when special notations are needed. For example, in vector analysis, operators for the gradient, divergence, and rotation (curl) of a vector field may be defined as follows:

```
\def\grad{\mathop{\operator@font grad}\nolimits}
\def\div{\mathop{\operator@font div}\nolimits}
\def\rot{\mathop{\operator@font rot}\nolimits}
```

Simply typing `\mathrm{div}` or even `\mbox{div}` instead of using a `\mathop` may appear to give similar results, but will not produce the proper spacing.

### Alignment of indices

By default, TeX uses a different shift amount for the placement of subscripts when a subscript appears by itself (such as in $x_0$) or when a subscript appears together with a superscript (such as in $x'_0$). While each case may be perfectly fine by itself, an inconsistency becomes apparent when both cases appear in the same formula, as in the example of a simple coordinate transform:

$$x(t) = x_0 + v_0 t\,. \qquad x'(t) = x'_0 + v'_0 t\,.$$

To avoid this kind of inconsistency there are two possible solutions (besides hacking the fontdimen parameters of math fonts, which may be a questionable solution). One solution consists of adding empty groups as phantom superscripts by typing `x_{0}^{}` to get $x_{0}^{}$. In this case, all subscripts would be lowered a little bit more, as if a superscript was always present:

$$x(t) = x_{0}^{} + v_{0}^{} t\,, \qquad x'(t) = x'_0 + v'_0 t\,.$$

Another solution consists of inserting empty groups to avoid a build-up of superscripts and subscripts by typing `x'{}_{0}` to get $x'{}_{0}$. In this case, all subscripts would be lowered a little bit less, as if each appeared by itself without a superscript. Unfortunately, some backspacing will be needed to close the visual gaps, so you would have to type `x'{}_{\!\!\!0}` to get the following result:

$$x(t) = x_0 + v_0 t\,, \qquad x'(t) = x'{}_{\!\!\!0} + v'{}_{\!\!\!0} t\,.$$

In general, the first solution may be easier to use in displays, but the second one may be useful if you want to prevent an expression from becoming too big.

## Sizes of delimiters

By default, TEX automatically determines the required size of big delimiters if `\left` and `\right` are used around a subformula. However, there are situations where `\left` and `\right` cannot be used, and there are also some situations where you may want to adjust the size of delimiters manually. In a typical setup, TEX provides the macros `\big`, `\Big`, `\bigg`, and `\Bigg`, which can be used to select specific sizes of delimiters, such as 12 pt, 18 pt, 24 pt, and 30 pt.

One example of using explicit font sizes is to avoid inconsistencies which may occur using the default settings. Consider the following equation:

$$\left(\frac{\cos\alpha}{\alpha}\right)^2 + \left(\frac{\sin\beta}{\beta}\right)^2$$

As it turns out, the fraction involving $\cos\alpha$ is surrounded by 18 pt (Big) delimiters whereas the fraction involving $\sin\beta$ requires 24 pt (bigg) delimiters. The reason is simply that $\cos\alpha$ is smaller because it has no ascenders or descenders, whereas $\sin\beta$ is bigger because it has both. If you want to ensure a consistent size of delimiters, it may be preferable to use `\bigg` (24 pt) on both expressions:

$$\biggl(\frac{\cos\alpha}{\alpha}\biggr)^2 + \biggl(\frac{\sin\beta}{\beta}\biggr)^2$$

Another example of using explicit sizes is to prevent delimiters from becoming too big. Consider the following equation:

$$\boldsymbol{R} = \left(\sum_{i=1}^{N} m_i \boldsymbol{r}_i\right) / M \,, \qquad M = \sum_{i=1}^{N} m_i \,.$$

Whenever you have delimiters around a summation the size calculation incorporates the upper and lower limits as well. If you just want to cover the summation sign without limits, it may be preferable to use `\Big` (18 pt) in this case:

$$\boldsymbol{R} = \Bigl(\sum_{i=1}^{N} m_i \boldsymbol{r}_i\Bigr) / M \,, \qquad M = \sum_{i=1}^{N} m_i \,.$$

Yet another example of using explicit sizes is to make delimiters bigger than they would normally appear. Consider the following equation:

$$\delta \int_{t_0}^{t_1} F\left(x(t), \dot{x}(t), t\right) = 0 \,.$$

Here, the outer level of delimiters is exactly the same size as the inner level, despite using `\left` and `\right` to get big delimiters. If you want the outer level to be more visible, it may be preferable to use `\big` (12 pt) in this case:

$$\delta \int_{t_0}^{t_1} F\bigl(x(t), \dot{x}(t), t\bigr) = 0 \,.$$

The same principle also applies if you have different kinds of nested delimiters and want the outer level to stand out, such as in this example:

$$\frac{\mathrm{d}\langle\widehat{\boldsymbol{p}}\rangle}{\mathrm{d}t} = \frac{1}{\mathrm{i}\hbar}\bigl\langle[\widehat{\boldsymbol{p}}, \widehat{H}]\bigr\rangle \,.$$

## Sizes of radicals

Unlike the size of delimiters which can be influenced manually, the size of radicals is always determined automatically based on the size of the subformula under the radical. In unfortunate circumstances, TeX may happen to choose a bigger size than you would like, such as in the following example:

$$\sqrt{p^2c^2+m_0^2c^4}\,, \qquad \frac{1}{\sqrt{p^2c^2+m_0^2c^4}}\,.$$

A tricky way to avoid this problem is to use staggered indices (as discussed earlier) to prevent subscripts from being lowered too much:

$$\sqrt{p^2c^2+m_0{}^2c^4}\,, \qquad \frac{1}{\sqrt{p^2c^2+m_0{}^2c^4}}\,.$$

Whether or not this problem occurs strongly depends on the choice of fonts and the relative size of indices used with these fonts.

When using a font family which provides optical design sizes, a typical setup will use font sizes of 10 pt / 7 pt / 5 pt for the text font and the first and second level indices. In this case, the 7 pt indices are usually small enough, so that a build-up of superscripts and subscripts does not exceed the body size.

When using a font without optical design sizes, the second level indices may become unreadable if a scaled-down version of the base font is used at 5 pt, so a different progression of sizes is used, such as 10 pt / 7.5 pt / 6 pt. In this case, the 7.5 pt indices may turn out a little too big, so that a build-up of superscripts and subscripts may require the next larger sizes of roots or delimiters.

## Spacing and backspacing

By default, TeX does a good job of spacing in math mode based on the various classes of math symbols such as ordinary symbols, binary operators, relations, openings, closings, or punctuation. However, in some cases, it may be desirable or even required to insert space manually for better results.

The most common example of manual spacing occurs in integrals, where a thinspace (\,) is usually inserted before each differential term, such as in:

$$\int F(x,y,z)\,\mathrm{d}^3V\,, \qquad \int F(r,\vartheta,\varphi)\,r^2\mathrm{d}r\,\sin\vartheta\,\mathrm{d}\vartheta\,\mathrm{d}\varphi\,.$$

(There is no need for a thinspace after an index or exponent (such as in $r^2\mathrm{d}r$) where a visual gap may not be needed or before an operator (such as before $\sin\vartheta$) where space is inserted automatically. However, there is a need for a thinspace before $\mathrm{d}\vartheta$ and $\mathrm{d}\varphi$ and also before $r^2\mathrm{d}r$.)

In addition, it is also a good idea to insert manual spacing before punctuation in displayed equations (as already shown in many examples), and in places where you want to emphasize the logical structure, such as after a prefix term or before an exponential term, as shown in the following example:

$$\psi(x,t)=\int A(k)\,\mathrm{e}^{\mathrm{i}(kx-\omega(k)t)}\,\mathrm{d}k\,.$$

Besides such examples of manual spacing used to emphasize the logical structure, there are also situations where manual spacing is needed to avoid visual collisions or where manual backspacing is needed to avoid visual gaps (such as in $v^2/c^2$). Some typical examples are explained in *The TEXbook*.

Another typical example of manual backspacing occurs when exponents are attached to big delimiters, as shown in the following example:

$$\mathrm{i}\hbar\,\frac{\partial\psi}{\partial t} = \frac{1}{2m}\left(\frac{\hbar}{\mathrm{i}}\boldsymbol{\nabla} - q\boldsymbol{A}\right)^{2}\psi\,.$$

Since TEX considers font metrics in terms of rectangular boxes and does not use an italic correction in this situation, the placement of superscripts is based only on the size of the box and does not take into account the rounded shape. To improve the appearance, it may be helpful to insert manual backspacing in the exponent by typing `\right)^{\!\!2}`, arriving at this result:

$$\mathrm{i}\hbar\,\frac{\partial\psi}{\partial t} = \frac{1}{2m}\left(\frac{\hbar}{\mathrm{i}}\boldsymbol{\nabla} - q\boldsymbol{A}\right)^{\!\!2}\psi\,.$$

Obviously, such corrections only apply to rounded or angular shapes, whereas square brackets do not need such corrections.

Finally, another interesting example of manual spacing or backspacing is the use of staggered indices in tensor analysis, such as in:

$$g^{\lambda\mu}g_{\mu\nu} = \delta^{\lambda}{}_{\nu}\,, \qquad g_{\lambda\mu}g^{\mu\nu} = \delta_{\lambda}{}^{\nu}\,.$$

Just inserting an empty group between superscripts and subscripts may create visual gaps, which may be compensated by a little bit of backspacing:

$$g^{\lambda\mu}g_{\mu\nu} = \delta^{\lambda}{}_{\!\nu}\,, \qquad g_{\lambda\mu}g^{\mu\nu} = \delta_{\lambda}{}^{\!\nu}\,.$$

Howe much backspacing is needed may depend on the individual letters involved, as the diagonal shape of $\lambda$ and $\nu$ happens to coincide in this particular example.

### Summary and conclusions on fine-tuning math formulas

In the previous sections, we have presented several examples of math formulas, illustrating various kinds of problems that may benefit from manual adjustments in order to improve the appearance of math formulas. Drawing from 2500 pages of material and seven years of experience, lots of examples and variety of notations have been seen, which are impossible to cover in this paper, but hopefully our examples may give helpful advice to authors involved in similar projects.

## References

[1] Eckhard Rebhan: *Theoretische Physik I: Mechanik, Elektrodynamik, Spezielle und Allgemeine Relativitititätstheorie, Kosmologie.*
Spektrum Akademischer Verlag, Heidelberg, 1999. xxvi + 1222 pp., ISBN 3-8274-0246-8

[2] Eckhard Rebhan: *Theoretische Physik II: Quantenmechanik, Quantenfeldtheorie, Elementarteilchen, Thermodynamik und Statistik.*
Spektrum Akademischer Verlag, Heidelberg, 2004. xxii + 1354 pp., ISBN 3-8274-0247-6

[3] Eckhard Rebhan: *Theoretische Physik: Mechanik.*
Spektrum Akademischer Verlag, Heidelberg, 2006. xiv + 390 pp., ISBN 3-8274-1716-3

[4] Eckhard Rebhan: *Theoretische Physik: Elektrodynamik.*
Spektrum Akademischer Verlag, Heidelberg, 2007. xii + 406 pp., ISBN 3-8274-1717-1

[5] Eckhard Rebhan: *Theoretische Physik: Quantenmechanik.*
Spektrum Akademischer Verlag, Heidelberg, 2008. xii + 536 pp., ISBN 3-8274-1718-3

[ 6 ] Eckhard Rebhan: *Theoretische Physik: Thermodynamik.*
Spektrum Akademischer Verlag, Heidelberg, 2009 (to appear). ISBN 3-8274-1719-X

[ 7 ] University of Chicago Press: *The Chicago Manual of Style.*
University of Chicago Press, 15th revised edition, 2003. xvii + 956 pp., ISBN 0-226-10403-6
`http://www.chicagomanualofstyle.org/`

[ 8 ] Ellen Swanson: *Mathematics into Type.*
American Mathematical Society, updated edition, 1999. x + 102 pp., ISBN 0-8218-1961-5

[ 9 ] International Union of Pure and Applied Physics, S.U.N. Commission: *Symbols, Units, and Nomenclature in Physics.* Document U.I.P. 20 (1978), also published in *Physica A*, 93:1–60, 1978.

[ 10 ] International Union of Pure and Applied Physics, SUNAMCO Commission: *Symbols, Units, Nomenclature and Fundamental Constants in Physics.* Document IUPAP 25 (1987), SUNAMCO 87-1, also published in *Physica A*, 146:1–67, 1987.

[ 11 ] International Union of Pure and Applied Chemistry, Physical Chemistry Division: *Quantities, Units, and Symbols in Physical Chemistry.*
Blackwell Science, 1st edition, 1988.

[ 12 ] International Union of Pure and Applied Chemistry, Physical Chemistry Division: *Quantities, Units, and Symbols in Physical Chemistry.*
Blackwell Science, 2nd edition, 1993.
`http://old.iupac.org/publications/books/gbook/green_book_2ed.pdf`

[ 13 ] International Union of Pure and Applied Chemistry, Physical Chemistry Division: *Quantities, Units, and Symbols in Physical Chemistry.*
Royal Society of Chemistry, 3rd edition, 2007.

[ 14 ] International Organization for Standardization: *Quantities and Units — Part 11: Mathematical signs and symbols for use in the physical sciences and technology.* Technical Report ISO 31-11:1992.
International Organization for Standardization, Geneva, 1992.

[ 15 ] International Organization for Standardization: *Quantities and Units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology.* Technical Report ISO 80000-2:2009.
International Organization for Standardization, Geneva, 2009.

[ 16 ] Claudio Beccari: *Typesetting mathematics for science and technology according to ISO 31-11.*
*TUGboat*, 18(1):39–48, March 1997.
`http://www.tug.org/TUGboat/Articles/tb18-1/tb54becc.pdf`

[ 17 ] Ulrik Vieth: *Requirements for Typesetting Physics.*
Math Font Group discussion document. Unpublished. October 1997.
`http://www.tug.org/twg/mfg/papers/ulrik/physreq.pdf`

Dr. Ulrik Vieth
Vaihinger Straße 69
70567 Stuttgart
Germany
ulrik dot vieth (at) arcor dot de

# Abstracts without papers

## Using TEX For Educational Publishing

Hans Hagen

It is no secret that we started using TEX to typeset educational materials: it shaped ConTEXt. Currently we still use TEX for projects that relate to education. Although we see a shift to more web based education we also see that print is in high demand for proper reading as well as examination. In this presentation I will show some of our current usage of TEX, for instance in XML to PDF workflows and PDF assembling workflows.

## TEX and the art of course maintenance

Lucien Lemmens

A course can be considered as a collection of of elements that are connected and depend on each other. If you change something in element A, it has consequences for element B. References to material outside the notes – software-distributions, wiki-books, websites – generate changes that are outside the control of the maintainer of the course. Others are made by the author: correction of typos, rephrasing, using other examples etc. If not well organized these changes can be very time consuming and have unexpected consequences – adding a few lines in a text can spoil the layout.

Using for instance a 'statistics course' as an example, the structure of the elements in the course: data, math → algorithm → figure, implies a connection between elements in the notes: table, formula → code → graphics. A change in the elements of the course implies a change in the notes.

For the statistical analysis the open source realization of the S-language R is used. Change in the data, keeping the math unchanged, results in changes in the code and the graphics. Giving the changed files the same name after one has archived the files that will be changed, LATEX and ConTEXt have respectively the packages and the commands to deal with these changes smoothly. When tables are involved LATEX has an advantage due to the exportation properties of R. A little bit of coding in R allows that the exported table can be read by ConTEXt directly.

A change in the references requires a better organization. The `log-file` gives the indication that something is missing but leads almost always to a time consuming procedure to get the references correct. A hidden file that shows the connections requires discipline to make but once it exists, it improves the maintenance of the notes considerably. Two examples will be shown and discussed: the first one is a report, made by a student, in LATEX and R, by a student, on a simulation. The second one is a subsection of my notes, typeset using ConTEXt on a course on data-analysis.

## LATEX for students, teachers and researchers — some remarks

Zofia Walczak

As we all know TEX was designed more than 30 years ago. Throughout the years TEX spread over three different groups of people – scientists, school teachers and, as a consequence, students.

The major group of TEX users is mathematicians as it was created specially for them. At present there is a wide range of TEX users. I will show some examples how LATEX can be useful for different purposes in different domains.

## TEX at The Open University

Jonathan Fine

The Open University is the UK's leading provider of distance learning. Since 1992 it has used TEX for the production of mathematics and upper-level physics courses. It is used to produce custom authored course-book, assignment booklets, exams and supplementary materials. Most of these materials are printed commercially, in two or four colours.

This talk will give a survey of the past, present and future use of TEX at the OU.

## TEXworks: Lowering the entry barrier to the TEX world

Jonathan Kew

A brief presentation of the TEXworks project to tell and show what it is, and a report on the current status.

## Typesetting lyrics with ConTEXt

Vyatcheslav Yatskovsky

My workflow includes getting lyrics from the Web, printing them on separate sheets of paper, inscribe guitar chords (harmony) and binding into a "songbook". While most artists prefer MS Word or plain text formats, I found ConTEXt best suitable for my task for the following reasons:

□ output formatting is consistent and rich, and this comes with almost no efforts comparing to WYSIWYG editors;
□ marking a lyrics with keywords like verse, chorus, solo, etc, clarifies its structure and makes the song much easier to memorize and perform.

I will be glad to explain the environment internals, show my workflow, and demonstrate the collection of 150+ neatly formatted pdfs.

## Rejoining the mainstream

Jonathan Fine

Much has changed in the world of publishing and communication since the release of TEX in 1982. TEX was rapidly adopted by mathematicians and physicists as a much-loved document preparation system, although with a steep learning curve. It is also used in other specialist areas, such as technical documentation and database publishing.

Since the 1990s computer power and networking has grown many times, as has use of computers. IBM has been replaced by Microsoft as the dominant commercial force in computing, which is now in turn challenged by Google. People are looking to the internet, particularly the Web, for information, services and solutions.

Although TEX remains mainstream for mathematical content, as open-source software it has slipped, and its Web presence is weak. In 2009 TUG was rejected by Google as a mentoring organisation for their Summer of Code. TEX-related websites are somewhat dated and developer communities isolated, compared to the organisations that were accepted.

This talk presents recent work and proposals aimed at helping TEX and related software return to the mainstream of document processing.

1. On-line documentation for TEX, LATEX and ConTEXt
2. Social networking :
   1. Mathematical content
   2. TEX development and support
3. Typesetting:
   – Web service
   – Shared objects and callable functions
4. Standards:
   1. LATEX syntax and XML
   2. Mathematical content and MathML
   3. Unicode (and XƎTEX)

## A comparison of free PDF libraries

Martin Schröder

One of the reasons for the success of pdfTEX is the quality of the PDF inclusion, which uses code from XPDF. Over the last years a number of (free) PDF libraries and tools have been developed. I will show some of these and compare them.

## Playing with font features

Hans Hagen

In this presentation we will explore what OpenType features are, what they do, or don't do, where they succeed or fail. I will use an interactive tool that ships with the ConTEXt distribution. One of the objectives of this presentation is to make users aware that OpenType is great but that one needs to be aware of limitations, potential side effects and that while installation and usage has become easier a somewhat greater knowledge is expected with respect to what they make possible.

## Dynamic font features

Hans Hagen

In ConTEXt MkIV we have several ways to enable OpenType features. In base mode we use TEX's core methods, while in node mode we do everything in Lua. Both are static methods in the sense that the set of features to be applied is associated with a font instance. Instead of these (or on top of these) you can use dynamic features. This method has several variants and these will be discussed in this presentation cq. tutorial. I will show what they do as well as present the user interface to them. When time permits I will also give a demonstration of yet another mechanism, tagged font strategies. This method is used in the Oriental TEX project.

## SVG support in MetaPost 1.200

Taco Hoekwater

Since version 1.110, Metapost has an alternative backend. Besides the ability to create Encapsulated PostScript, it is now also possible to create Scalable Vector Graphics output. This talk shows some examples of this new backend and also highlights a few related extensions to Metapost that have been added in version 1.200.

## Licensing of the TEX Gyre family of fonts

Jerzy Ludwichowski

URW++ Design and Development, the well-known font foundry, which in 1996 donated the so-called basic 35 PostScript Type 1 fonts to the public under both the GNU Public License (GPL) and Aladdin Free Public License (AFPL), has on 22nd June 2009 agreed to release the same fonts under the LATEX Project Public License (LPPL).

This presentation will explain the significance of URW's decision for the TEX community, with special emphasis on the TEX Gyre font family.

## Math fonts: Notes from the trenches

Bogusław Jackowski

About a year ago, a math fonts expedition was organised by TEX LUGs. After a brave beginning, however, the offensive is now stuck in its trenches. Nonetheless, optimistic signs of the future victory are more and more apparent.

The necessary background information and the available technical data will be given along with the layout of the plans for the imminent math fonts offensive.

## Handwriting fonts, METAFONT and OpenType

Karel Píška

The fonts cover handwriting scripts used in Czech, Armenian and Georgian schools. METAFONT, Type 1 and OpenType solutions are presented. Different techniques applied in METAFONT and OpenType, especially for inclusion of numerous connections between letters or various glyph modifications, will be compared.

The original METAFONT Czech font slabikar was created by Petr Olšák; other fonts have been produced by the author. The fonts could be used for educational purposes.

## Secrets of a TEX distribution: ConTEXt minimals

Mojca Miklavec & Arthur Reutenauer

What does it take for a packaging system to follow the fast pace of the ever-improving ConTEXt? The 'new' ConTEXt minimals are an attempt at an answer.

Now the successor to the first 'minimal distribution' that was available as zip files from the Pragma web site, it ships all the necessary files in a single structure: the ConTEXt core of course, with its TEX and Lua code and its support scripts, but also third-party modules that can be retrieved upon desire since we aim at modularity; and, more importantly, the distribution also includes all the necessary binaries for the most popular architecture, in a sufficiently new version: Mark IV always needs a very recent version.

This latter point was one of the major incentives to create a new distribution; another one was the desire to avoid downloading big archive files when only a few source files were modified: in order to achieve that, we now use the rsync protocol, together with a minimal setup script on Unix systems, including Mac OS, and an install wizard for Windows, written by Vyacheslav Yatskovsky.

## PPCHTEX revisited

Hans Hagen

About 15 years ago I wrote a module for typesetting chemical structure formulas: PPCHTEX. The next few years it evolved and stabilized pretty well. The only extension till now has been that MetaPost replaced the PicTEX graphics but still PicTEX was used to position the lot. Although not commonly known, the fact that at that point ConTEXt had a Dutch user interface while PPCHTEX was kind of generic is one of the reasons why ConTEXt now has multiple user interfaces and became useable for those who didn't like Dutch.

Triggered by a question at the ConTEXt mailing list I decided to freeze the MkII version and played a bit with the code. I quickly concluded that it was about time to reprogram the lot for MkIV and get rid of the dependency on PicTEX. In this workshop for ConTEXt users I will demonstrate how we can combine the power of TEX, MetaPost and Lua to make quite efficient and compact modules that otherwise demand quite some auxiliary code.

## New structures in ConTEXt

Hans Hagen

Most of the structure related code in ConTEXt has been rewritten and uses Lua extensively for housekeeping. This step finalized the move of all multipass data to Lua. As we carry more state information around, we can also more conveniently support for instance multipass XML. In this presentation I will show where information ends up and in what way future versions will provide users with access to additional information.

## Upcoming spacing mechanisms

Hans Hagen

One of the complications with vertical spacing is that the more a macro package provides, the more interference between structural components is possible. There is only so much one can do about it, especially because TEX is not that good at looking back. In MkIV we will have a revisited vertical spacing model, one that eventually will replace the existing model.

# Participant list

Frans Absil, The Netherlands
  Netherlands Defence Academy
Nino Bašić, Slovenia
  FMF, University of Ljubljana
Gyöngyi Bujdosó, Hungary
  University of Debrecen
Wybo Dekker, The Netherlands
Alain Delmotte, Belgium
  Le Liseron éditions
Willi Egger, The Netherlands
  BOEDE
Jonathan Fine, United Kingdom
  Open University
Frans Goddijn, The Netherlands
  Stichting Johan Polak
Michel Goossens, Switzerland
  CERN
Steve Grathwohl, USA
Patrick Gundlach, Germany
Michael Guravage, The Netherlands
  Literate Solutions
Hans Hagen, The Netherlands
  PRAGMA ADE
Hartmut Henkel, Germany
  von Hoerner & Sulger GmbH
Taco Hoekwater, The Netherlands
  Bittext
Klaus Höppner, Germany
  DANTE e.V.
Jean-Michel Hufflen, France
  University of Franche-Comté – LIFC
Jelle Huisman, United Kingdom
  SIL International
Bogusław Jackowski, Poland
  GUST
K. M. Jeary, United Kingdom
  University of Cambridge
Jonathan Kew, United Kingdom
Rukhsar Khan, Germany
  Airnet Technologie- und Bildungszentrum GmbH
Harald König, Germany
Reinhard Kotucha, Germany
Siep Kroonenberg, The Netherlands
  Rijksuniversiteit Groningen, Faculteit economie en bedrijfskunde
Johannes Küster, Germany
  typoma GmbH
Kees van der Laan, The Netherlands
Lucien Lemmens, Belgium
  Universiteit Antwerpen
Manfred Lotz, Germany
Jerzy Ludwichowski, Poland
  GUST
Gisela Mannigel, Germany
Niall Mansfield, United Kingdom
  UIT
Bernd Militzer, Germany
Wolfgang Murth, Austria
  WMS Modell & Technik
Mahdi Omidali, Iran
  Islamic Azad University (Ghazvin Branch)
Karel Píška, Czech Republic
  Institute of Physics, Academy of Sciences
John Plaice, Australia
  The University of New South Wales
Bernd Raichle, Germany
  DANTE e.V.
Arthur Reutenauer, France
  GUTenberg
Stanislav Jan Šarman, Germany
  Rechenzentrum der TU Clausthal
Luigi Scarso, Italy
  logosrl
Martin Schröder, Germany
  QuinScape GmbH
Martin Sievers, Germany
  Einfach schöner publizieren
Péter Szabó, Switzerland
  Google
Philip Taylor, United Kingdom
John Trapp, United Kingdom
  Open University
Eszter Urbán, Hungary
Eva Van Deventer, South Africa
  University of South Africa
Ulrik Vieth, Germany
Mari Voipio, Finland
  K-Patents Oy
Jan de Vries, The Netherlands
  Netherlands Defence Academy
Zofia Walczak, Poland
  University of Łódź
Kevin Warnock, USA
  gOffice.com
Steffen Wolfrum, Germany
Vyatcheslav Yatskovsky, Ukraine
  National Aviation University

# TeX Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at `http://tug.org/consultants.html`. If you'd like to be listed, please see that web page.
To place a larger ad in *TUGboat*, please see `http://tug.org/TUGboat/advertising.html`.

**Dangerous Curve**
PO Box 532281
Los Angeles, CA 90053
+1 213-617-8483
Email: `typesetting (at) dangerouscurve.org`
Web: `http://dangerouscurve.org/tex.html`

We are your macro specialists for TeX or LaTeX fine typography specs beyond those of the average LaTeX macro package. If you use XeTeX, we are your microtypography specialists. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical TeX and LaTeX typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a TeX book.

**Martinez, Mercè Aicart**
Tarragona 102 4$^o$ 2$^a$
08015 Barcelona, Spain
+34 932267827
Email: `m.aicart (at) ono.com`
Web: `http://www.edilatex.com`

We provide, at reasonable low cost, TeX and LaTeX typesetting services to authors or publishers world-wide. We have been in business since the beginning of 1990. For more information visit our web site.

**Peter, Steve**
New Jersey, USA
+1 732 287-5392
Email: `speter (at) mac.com`

Specializing in foreign language, linguistic, and technical typesetting using TeX, LaTeX, and ConTeXt, I have typeset books for Oxford University Press, Routledge, and Kluwer, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. I have extensive experience in editing, proofreading, and writing documentation. I also tweak and design fonts. I have an MA in Linguistics from Harvard University and live in the New York metro area.

**Shanmugam, R.**
No. 38/1 (New No. 65), Veerapandian Nagar, Ist St.
Choolaimedu, Chennai-600094, Tamilnadu, India
+91 9841061058
Email: `rshanmugam92 (at) yahoo.com`

As a Consultant, I provide consultation, training, and full service support to individuals, authors, typesetters, publishers, organizations, institutions, etc. I support leading BPO/KPO/ITES/Publishing companies in implementing latest technologies with high level automation in the field of Typesetting/Prepress, ePublishing, XML2PAGE, WEBTechnology, DataConversion, Digitization, Cross-media publishing, etc., with highly competitive prices.
I provide consultation in building business models & technology to develop your customer base and community, streamlining processes in getting ROI on our workflow, New business opportunities through improved workflow, Developing eMarketing/E-Business Strategy, etc. I have been in the field BPO/KPO/ITES, Typesetting, and ePublishing for 16 years, handled various projects. I am a software consultant with Master's Degree. I have sound knowledge in TeX, LaTeX2$\varepsilon$, XMLTeX, Quark, InDesign, XML, MathML, DTD, XSLT, XSL-FO, Schema, ebooks, OeB, etc.

**Sievers, Martin**
Im Treff 8, 54296 Trier, Germany
+49 651 49 651 4936567-0
Email: `info (at) schoenerpublizieren.com`
Web: `http://www.schoenerpublizieren.com`

As a mathematician with ten years of typesetting experience I offer TeX and LaTeX services and consulting for the whole academic sector (individuals, universities, publishers) and everybody looking for a high-quality output of his documents.

From setting up entire book projects to last-minute help, from creating individual templates, packages and citation styles (BibTeX, biblatex) to typesetting your math, tables or graphics — just contact me with information on your project.

**Veytsman, Boris**
46871 Antioch Pl.
Sterling, VA 20164
+1 703 915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`

TeX and LaTeX consulting, training and seminars. Integration with databases, automated document preparation, custom LaTeX packages, conversions and much more. I have about fourteen years of experience in TeX and twenty-seven years of experience in teaching & training. I have authored several packages on CTAN, published papers in TeX related journals, and conducted several workshops on TeX and related subjects.