EuroTEX 2012 & 6th ConTEXt Meeting

October 8–12, 2012, Breskens, The Netherlands

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

- Tweemaal per jaar een NTG-bijeenkomst.
- Het NTG-tijdschrift MAPS.
- De 'TEX Live'-distributie op DVD/CDROM inclusief de complete CTAN 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 $85.

Afmelding kan met ingang van het volgende kalenderjaar door opzegging per e-mail aan de penningmeester.

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 1.40.11 en luatex 0.70.1 draaiend onder Linux 2.6. De gebruikte fonts zijn Linux Libertine, het niet-proportionele font Inconsolata, schreefloze fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

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

Contents

Conference program

Monday, October 8

08:00		Breakfast
09:00		Conference opening
09:15	Kees van der Laan	Recreational TEX&Co - with a serious undertone
10:15	Jano Kula	Run for Fun
11:00		Coffee
11:30	Mari Voipio	TEXtile crafts
12:30		Lunch
14:00	Kees van der Laan	Julia fractals in PostSCript
14:45	Taco Hoekwater	MetaClock
15:30		Coffee
16:00	Patrick Gundlach	Database publishing with LuaTEX and the speedata Publisher
16:45	Patrick Gundlach	A journey to the land of LuaLATEX
17:30		Dante membership meeting
19:00		Dinner
20:30		CG membership meeting

Tuesday, October 9

08:00		Breakfast
09:00	Uwe Ziegenhagen	Professional Business Reports with LATEX
09:45	Leo Arnold	Many versions from one source - LATEX for lecturers and teachers
10:30		Coffee
11:00	Taco Hoekwater	MetaPost path resolution isolated
11:45	Taco Hoekwater	Parsing PDF content streams with LuaTEX
12:30		Lunch
14:00	Hans Hagen	context: the script
14:45	Hans Hagen	context: after the cleanup
15:30		Coffee
16:00	Luigi Scarso	MFLua
16:45	Mari Voipio	Metapost workshop
19:00		Dinner
20:30	Willi Egger	Conference folder workshop

Wednesday, October 10

08:00		Breakfast
09:00		Excursion
19:30		Conference Dinner

Thursday, October 11

08:00		Breakfast
09:00	Bogusław Jackowski	OTF math fonts: GUST e-foundry's workbench
09:45	Jerzy Ludwichowski	Present and future of the TG Math Project: the report and some questions
10:30		Coffee
11:00	Piotr Strzelczyk	Is backward compatibility of LM Math and CM math sensible?
11:45	Gust Foundry	BOF session: 'The future of math fonts'
12:30		Lunch
14:00	Hans Hagen	xml
14:45	Hans Hagen	a couple of styles
15:30		Coffee
16:00	Hans Hagen	lexing
16:45	Hans Hagen	(visual) debugging
17:30	– –	Glass blowing demonstration (registration required, costs 5 EURO)
19:00		Dinner

Friday, October 12

08:00		Breakfast
09:00	Yamamoto Munehiro	TEX Typesetting Circumstances for Japanese Publishing
09:45	Kitagawa Hironori	Japanese Typesetting with LuaTEX
10:30		Coffee
11:00	Hans Hagen	Tricks with the parbuilder (Arabic typesetting)
11:45	Ivo Geradts Kai Eigner	Typesetting Sanskrit with LuaTEX
12:30		Lunch
14:00	Hans Hagen	mixed columns
14:45	Tomás Hála	Differences in typesetting rules between Czech and Slovak languages (in the context of ConTEXt)
15:30		Coffee
16:00	Jean-Michel Hufflen	MlBibTEX and Its New Extensions
16:45	Jean-Michel Hufflen	Demonstration of the `mlbibcontext` Program
17:45	Sietse Brouwer	Bof session: Context Wiki
18:30		2013 Announcements
18:45		Conference closing
19:00		Dinner

Abstract
Recreational use of TEX&Co in my work is enumerated and elucidated. Examples from MetaFun, from Lancaster's Fonts for Free, from Jackowski&Ryćko metafont logo, and from Word have been borrowed. PostScript and let TEX insert mark-up, will be the main subjects of discussion. PostScript is not sufficient for graphics. Now and then MetaPost is used to specify a problem in a declarative way, or at the end Photoshop is used to enrich the graphics interactively by colour gradients. Moreover, for drawing emulations of 3D objects, projection techniques are indispensable. Emulations of Escher's impossible cube and of Gabo's objects are included as 3D-examples. All my pictures have a recreational flavour because none has been triggered by external practical need. Interesting is the combined use of Turtle Graphics and recursion. TEX codes and PostScript codes are compared, although they are like apples and pears intrinsically incomparable, but ... have been used for the same purpose. The most astonishing is that so much from `BLUe.tex` passed by unnoticed. `Pic.dat` for TEX-alone pictures has received its cousin library, `PSlib.eps`, for PostScript pictures. The TEX-MF-flow picture has been updated and included, next to a screen-shot of a nowadays IDE TEXworks. In this note I'll try to draw your interest, to persuade you, kind reader, to look at the contents, the paradigms, and the kernel and modules set-up of `BLUe.tex`. My sincere hope is that `BLUe.tex` will be saved from oblivion, that the paradigms used will be adhered.
The serious undertone in TEX is about minimal mark-up or better still the absence of user mark-up, where TEX will insert the mark-up. The serious undertone in PostScript is about printing along paths, especially for the special cases where the paths are implicit. Handy and convenient is the extended `PSlib.eps` to over 300 pictures.
Critics on TEX&Co and pdfTEX have been included, next to my wishes.
After the presentation `PSTricks` was shown to me, and my comment on it is included.

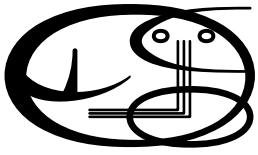
Keywords
Acrobat Pro, Adobe, art, automatic mark-up, backtracking, BLUe, Blue Sky research, bridge, Caroll, chess, Cohen, ConTEXt, crosswords, dancing text, Deubert, Ensor, EPSF, Escher, FIFO, font charts, function-grapher previewer, Gabo, Hagen, Henderson, IDE (Integrated Development Environment), impossible figures, Jackowski, Lancaster, Lauwerier, LIFO, Lindenmayer, magic square, Malevich, Margritte, MetaFun, MetaPost, Metafont, MetaType1, minimal encapsulated PostScript, minimal mark-up, minimal plain TeX, Mondriaan, Monte Carlo, musiX.tex, mppreviewer, Nolde, Photoshop, π-decimals, projection, PSlib, PSTricks, PSView, Pythagoras tree, Ryćko, Schrofer, Soto, Taupin, TEXworks, tic-tac-toe, Vasarely.

Introduction

In the late 80s I became aware of TEX and immediately realized the relevance for a university community. I started the 'Publiceren met LATEX' project, which resulted in the CWI-syllabus 19. We organized a LATEX course at Utrecht. University users found their way in how to use LATEX. I became 1st president of NTG.

In order to learn macro-writing I developed my bridge macros, which marks my start of Recreational use of TEX. My learning of macro-writing culminated in `BLUe.tex` and my 'Publishing with TEX' guide,[1] which concentrates on what can be done by TEX alone, without incorporating the results of graphics software.

My next project was typesetting tables by TEX, where I en-passant looked for a taxonomy of tables. The conclusion of this work was that tables are too varied, but one could discern a broad class of tables which have a border and in there the proper table data. This lead to my table macros, which I presented at the EuroTEX92 at Prague.[2] My Recreational use of TEX in the table area are amongst others bridge layouts, the crosswords table, a magic square, and the PASCAL's triangle of binomial coefficients.



At the Prague conference I was impressed by Karel Horák's graphical work in Metafont. At home I started to use Metafont for graphics, mainly recreational, which resulted in my 'cat'.[3] The incorporation of Metafont graphics in TEX via symbols of a font I experienced as inconvenient. Later I learned about `\psfig`, which easily lets you include PostScript pictures in TEX-documents. The use of `\psfig` marks my beginning of TEX&PostScript. My viewer was the Apple Laserwriter, and not `PSView` which was not available on my PowerMac. My use of PostScript as part of TEX&Co has a strong recreational flavour. Many pictures have been inspired by work of artists, such as Escher, Gabo, Malevich, Mondriaan, Soto, Schrofer, Vasarely, ... as can be witnessed in this paper. Pictures have been improved:

- □ PostScript pictures resulting from MetaPost with at least better BoundingBox values,
- □ gkp-pictures were done anew in PostScript now and then, and included in `PSlib.eps`.

All pictures come with better explanations. Inclusion in my pdfTEX-documents goes by the macros `\pdfximage... \pdfrefximage\pdflastximage`, or my less-verbose macros `\insertjpg`, c.q. `\insertpdf`.

My use of graphics with TEX marks five periods: 1st by LATEX's picture-environment, 2nd by TEX's gkp-macros, with the same functionality as the picture-environment, 3rd by Metafont, supported by projection techniques, 4th by MetaPost, 5th by PostScript, supported by Photoshop as post-processor, mainly for colour gradients.

This paper consists of examples from earlier MAPS papers, from Hagen's MetaFun, from Lancaster's Fonts for Free, from the 3D Jackowski&Ryćko metafont logo, from Word and from literature. The 1st appendix contains my balanced binary tree macros in TEX of old next to my superior PostScript variant, on occasion of the EuroTEX-ConTEXt2012. Another appendix contains determination of the BoundingBox values in 1-pass, on-the-fly. The 3rd appendix contains a LMR font table.

Select what you are interested in. If only you enjoy one picture, kind reader, I'm happy already ■.

1. Available from CTAN.
2. Knuth considers bordered matrices in the TEXbook but does not mention bordered tables.
3. Much later the cat was adapted to MP and the resulting data resulted in an EPSF.

One by one the guests arrive, MAPS 96.2 1996

This 1-page paper is best read with Cohen's song in the background.[4] It is a plea for serious — non-recreational — use of TEX.

> "*...This plea, this shout, hopes to awaken the notion that we are all better off if we write macros in the lowest common set of all TEX-flavours, i.e. plain TEX. At least it might initiate a discussion whether to do so or otherwise, because I'm realistic enough that not all share my views ...*"

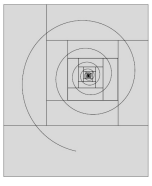
A little later the song continues

> "*And no one knows where the night is going*
> *And no one knows why the wine is flowing*
> *O love, I need you, I need you, I need you*
> *I need you now ...*"

The point I'm trying to make is that we are all better off when complex fundamental parts will be programmed in plain TEX, perhaps after it has proven to be worth it.[5]

To end Cohen's song

> "*The guests are coming through*
> *The open-hearted many*
> *The broken-hearted few*"



Looking back the TEX-community decided otherwise: LATEX-packages are contributed to CTAN; ConTEXt and LuaTEX were developed; the new Latin Modern Roman fonts are Adobe Type 1 . The GUST e-foundry TEX-Gyre OTF-project is under way, funded by several LUGs and TUG. But ... nevertheless I keep saying it.

Macros from `BLue.tex` and pictures from `PSlib.eps` can be reused even by ConTEXt, LaTEX, ...respectively MetaPost users, because they are written in the common plain TEX subset, respectively the underlying PostScript. But there is more than reuse ...MetaType 1 fonts are in Adobe Type 1, however ... Adobe has declared Adobe Type 1 obsolete, see Ludwichowski this proceedings.

The Life-cycle diagram of publications is one of my favourites. The invoke of `\halign` is straight-forward.

$$\begin{array}{ccccc} \text{Produce} & \rightarrow & \text{Distribute} & \rightarrow & \text{Consume} \\ \uparrow & & \uparrow & & \downarrow \\ \text{reuse} & \leftarrow & \text{retrieve} & \leftarrow & \text{store} \end{array}$$

```
\halign{&\enspace\hfil#\hfil\cr
   Produce&$\rightarrow$&Distribute&$\rightarrow$&Consume\cr
$\uparrow$&            &$\uparrow$&            &$\downarrow$\cr
    reuse&$\leftarrow$ &retrieve  &$\leftarrow $&store\cr}
```

In principle the above life-cycle is OK, but ... in practice the reuse aspect is hampered by changes, such as a different IDE

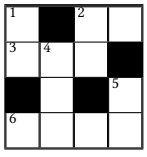
- ☐ or a new TEX engine, such as pdfTEX, which no longer supports for example `\psfig`
- ☐ or programs have become obsolete such as the picture environment
- ☐ or the gkp-macros have become outdated, as happened with the Happy Birthday cake picture.

Moreover, it is hard to maintain original data over time, over computer renewals. Nevertheless ...

4. During the presentation the tune was played by just pushing a button in my slide, multi-media, aha.
5. The same holds for pictures: we should create and adhere a library of PS pictures. Why not start with `PSlib.eps`?

Typesetting Crosswords via TEX, MAPS 8, 1992

The typesetting crosswords tool, as one of the tools in `tools.dat`, comes with `BLUe.tex`. The environment is `\begincrosswords` ... `\endcrosswords`. The example has been borrowed from the table chapter of PWT. The crosswords tool has been copied from BLUe's `tools.dat` and used stand-alone in this paper.



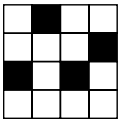
Across
2 Switch mode
3 Knuth
6 Prior to TEX

Down
1 Public domain
2 All right
4 All comes to it
5 Atari type


```
\begincrosswords
$\bdata
P*On
DEk*
*n*S
Edit
\edata
\crw
<Clues in 2 \vtop's, v-centered>
\sol
\endcrosswords
```

Interesting is the near-WYSIWYG-data specification of the puzzle. Minimal mark-up has been strived after, no `\cr`-s nor `&`-s have to be inserted by the user, TEX will do it for you. Mean-and-lean is that the solution or the puzzle can be toggled by `\sol` respectively `\crw`. Note the use of capitals and lower case. The capitals mark where a number for the clues has to be inserted, automagically ☺. Paradigm: let TEX insert mark-up.

A variant via PostScript inspired by David Byram–Wigfield,[6] who created a special font `QuadFont`, interesting in itself, for the black and white squares. But ... without numbers for the clues and no toggling of solution and puzzle. In my PS version I simplified, without creating `QuadFont`. TEX's version is superior.



```
/x{0 0 20 20 rectstroke 20 0 translate} def
/X{0 0 20 20 rectstroke 0 0 20 20 rectfill 20 0 translate} def
/crl{-80 -20 translate 0 0 moveto} def
x X x x crl              %X denotes black and x denotes white
x x x X crl
X x X x crl
x x x x crl
```

Typesetting Bridge via TEX, MAPS 7, 1991

My recreational use of (La)TEX started with writing LaTEX bridge macros in 1990.[7] In 1995 as part of `BLUe.tex` the plain TEX variants became a tool in BLUe's toolbox. The macros are available in `BLUe.tex`'s `tools.dat`. The environment `\beginbridge` ...`\endbridge` selectively loads, behind the scenes and OS-independently, the macros from the `tools.dat` into your BLUe job. They can also be copied from the toolbox, manually, and used as a independent part, without BLUe, as I did for this paper.

6. For a wealth of examples see Practical PostScript—A guide to Digital Typesetting. David Byram–Wigfield. `http://www.cappella.demon.co.uk`, or John Deubert's `http://www.acumentraining.com/acumenjournal.html`.

7. Bridge is a card game and played with 52 cards: A K Q J T 9 ...3 2, each in the suits: , , and . There are 4 players around a table called North, East, South and West. N and S form a team, so do E and W. The cards are dealt, each receives 13 cards and then the auction starts. After the auction the playing of the cards begins. A game takes 5-7min.

Puzzle KQ76 6NT
J98 by East
J942
65

AJ3 T9
K653 A2
AK3 T5
AQT KJ9xxxx

N
W E
S

8542
QT74
Q876
2

```
\beginbridge%loads bridge nacros
\def\LFTINF{Puzzle}
\def\RGTINF{\vtop{\hbox{6NT}
                  \hbox{by East}}}
\Ns={KQ76}\Es={T9}     \Ss={8542}\Ws={AJ3}
\Nh={J98} \Eh={A2}     \Sh={QT74}\Wh={K653}
\Nd={J942}\Ed={T5}     \Sd={Q876}\Wd={AK3}
\Nc={65}  \Ec={KJ9xxxx}\Sc={2}   \Wc={AQT}

\showgame
```

Trick					NS	EW
1	4!	K	8	2	–	1
2	A	5	x	2	–	2
3	Q	6	x	2	–	3
4	T	9	K	4	–	4
5	J	5	3	6	–	5
6	9	8	5	7	–	6
7	x	6	J	2	–	7

```
\LEADS\bplay
h4! & hK & h8 & h2 & --& 1\LEADW\cr
cA  & c5 & cx & c2 & --& 2\cr
cQ  & c6 & cx & s2 & --& 3\cr
cT  & h9 & cK & s4 & --& 4\LEADE\cr
cJ  & s5 & s3 & s6 & --& 5\cr
c9  & s8 & h5 & s7 & --& 6\cr
cx  & d6 & sJ & d2 & --& 7\cr
```

Puzzle KQ NS squeezed on
J continuation?
J94
–

A T9
63 A
AK3 T5
– x

–
QT7
Q87
–

8	x	7	6	J	–	8

et cetera

```
\bintermezzo
\def\RGTINF{\vtop{\hbox to 0pt{NS squeezed on\hss}
            \hbox to 0pt{\cl\ continuation?\hss}
            \hbox to 7ex{\hss}}} %phantom for alignment
\Ec={{\ooalign{\hfil\raise.07ex%
    \hbox{x}\hfil\crcr\mathhexbox20D}}}%mark lead
\showgame  \Ec={x}%restore E club
\eintermezzo
cx & h7 & h6 & hJ & --& 8\cr
\omit et cetera \hidewidth \cr
\eplay\endbridge
```

The `\bplay` ...`\eplay` table is interrupted by showing the status of the play, the remaining cards, between trick 7 and 8. Interesting is that data integrity has been strived after, because played cards have been removed from memory. Note the minimal mark-up: h just means hearts. At the time I did not know how TEX could include the `&`-s and `\cr`-s.

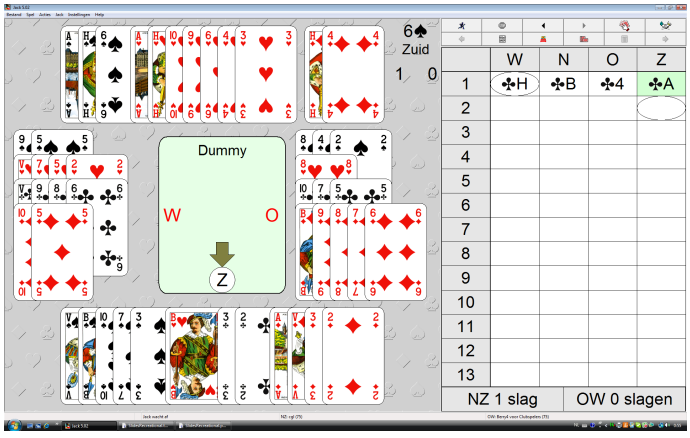
There is also an auction-environment. The example is borrowed from the table chapter of Publishing with TEX, PWT for short. In the LATEX Graphics Companion the LATEX bridge macros are mentioned, and some results have been shown.

From the macro-writing point of view, the dynamically declaration of token variables, as shown below, is interesting; a paradigm. The cards as sets and TEX-operations on sets is a paradigm too.

```
%\NT is alias of \newtoks without restricted use; \ea means \expandafter
\ea\let\ea\NT\csname newtoks\endcsname
\NT\Ns\NT\Es\NT\Ss\NT\Ws \NT\Nh\NT\Eh\NT\Sh\NT\Wh
\NT\Nd\NT\Ed\NT\Sd\NT\Wd \NT\Nc\NT\Ec\NT\Sc\NT\Wc \NT\hnd
```

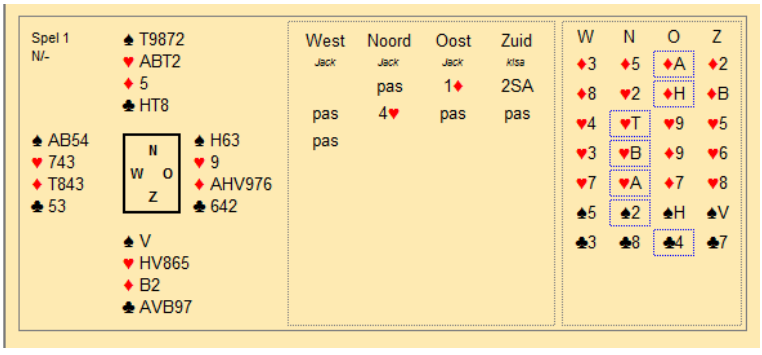
Computers and Bridge

In the past 20 years we have witnessed an enormous development, and increase in the use, of computers. In Bridge this has resulted in Bridge playing software such as the Dutch multiple Computer Bridge World-champion Jack.



Jack plays bridge

Characteristics: Data integrity, WYSIWYG input, Portable Bridge Notation standard, HTML export.



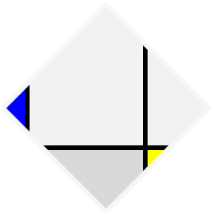
Jack bridge reporting

Do notice that typesetting is an aside for the developers of Jack.

The (recreational) play is assisted by Bridgemate (chipped-)boxes, which are used for the registration of the scores and are Wi-Fi coupled to the tournament directors computer for calculating the ranking. The results are put on the club's WWW page, made possible by the Nederlandse Bridge Bond, where the club members may find the scores and the ranking. All the hands played are usually also available, on the WWW and on an A4-sized paper.

Mondriaan inspired invitation

The Mondriaan (background) lozenge has been emulated in PostScript. The complete invitation after merging of the photograph and adding text, has been done in Photoshop. Emulated Mondriaan →

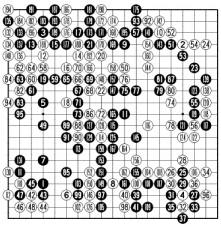


Other Games and TEX

Chapter 10 of the LaTEX Graphics Companion is devoted to Playing Games: Chess, Chinese Chess, Go, Backgammon, Card Games, Crosswords in various forms, and Sudokus.[8] In the sequel I'll mention what has been published in MAPS on the issue.

Hanna Kołodzieska has published in MAPS 7, p63–68, 1991, her 'Go diagrams with TEX.' She was inspired by Zalman Rubinstein, 'Chess printing via MetaFont and TEX,' TUGboat, 10, 2.
Piet Tutelaers has published in the (same) MAPS issue on the occasion of the NTG meeting about Games&TEX, 'A font and Style for Typesetting Chess using (La)TEX,' MAPS 7, p41-46.

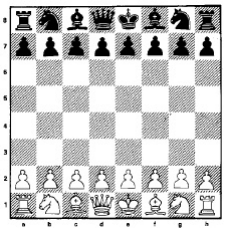
Go diagrams with TEX

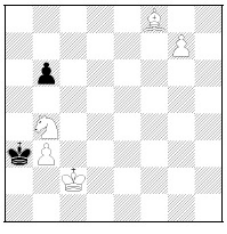


←Go situation

Chess board with pieces →

Chess position →→

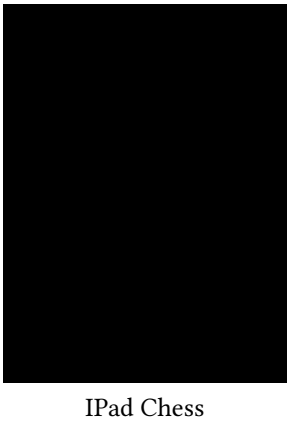




```
\board{ * * B * }
      { * * * P }
      { P * * * }
      { * * * * }
      { N * * * }
      {kP* * * }
      { *K* * * }
      { * * * * }
```

Computer chess is computer architecture encompassing hardware and software capable of playing chess autonomously without human guidance. Computer chess acts as solo entertainment (allowing players to practice and to better themselves when no human opponents are available), as aids to chess analysis, for computer chess competitions, and as research to provide insights into human cognition.

Chess-playing computers are now accessible to the average consumer. From the mid-70's to the present day, dedicated chess computers have been available for purchase. There are many chess engines such as Crafty, Fruit and GNU Chess that can be downloaded from the Internet for free. These engines are able to play a game that, when run on an up-to-date personal computer, can defeat most master players under tournament conditions. Top programs such as the Proprietary software programs Shredder or Fritz or the open source program Stockfish have surpassed even world champion calibre players at blitz and short time controls. In October 2008 Rybka was rated top in various rating lists and has won many recent official computer chess tournaments such as CCT 8 and 9, the 2006 Dutch Open Computer Championship, the 16th IPCCC, and the 15th World Computer Chess Championship. As of August 2012, Houdini is the top rated chess program on the IPON rating list with Rybka in 5th place.
Courtesy `http://en.wikipedia.org/wiki/Computer_Chess`.

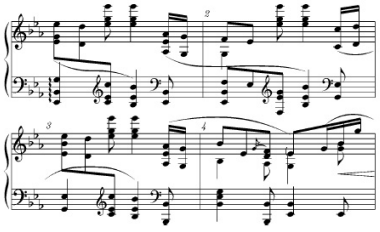


IPad Chess

8. Curiously Draughts is missing.

TEX and Music

Chapter 9 of the LaTEX Graphics Companion is devoted to Preparing Music Scores, and consists of 76p. It is hardly for fun. MusiXTEX of the late Daniel Taupin is leading. Too advanced and too difficult to be treated here, as can be witnessed from the various pre-processors to simplify the use.

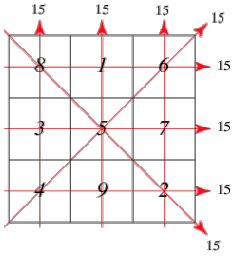


Graphics in Publishing with TEX, 1995

The graphics in PWT is limited, because the graphics is obtained by TEX alone. The gkp-macros have been used for PWT. These macros are limited due to the few discrete orientations of lines and there is no colouring.

Magic Squares recreational Math

A magic square of order n, is a square array of numbers consisting of the distinct positive integers 1, 2, ... n, arranged such that the sum of the numbers in any horizontal, vertical, or main diagonal line is always the same number, known as the magic constant $M_n = .5n(n^2+1)$.
Proof. Sum of all elements $\sum_{k=1}^{N} k = .5N(N+1),\ N = n^2$. One column, or row, sums up to $.5N(N+1)/n = .5n(n^2+1)$, the magic constant M_n.
In `http://en.wikipedia.org/wiki/Magic_square` curious, recreational Math algorithms are mentioned for squares of (double) even and odd n.



A 4x4 magic square puzzle is available at `http://www.dubster.com/math/`, where one can drag-and-drop the pieces; the magic constant is 30.

For odd squares the fun algorithm reads ... Starting from the central column of the first row with the number 1, the fundamental movement for filling the squares is diagonally up and right, one step at a time. If a filled square is encountered, one moves vertically down one square instead, then continuing as before. When a move would leave the square, it is wrapped around to the last row or first column, respectively.

```
- 1 -     - 1 -     - 1 -     - 1 -     - 1 -     - 1 6     - 1 6     8 1 6     8 1 6
- - - →   - - - →   3 - - →   3 - - →   3 5 - →   3 5 - →   3 5 7 →   3 5 7 →   3 5 7
- - -     - - 2     - - 2     4 - 2     4 - -     4 - 2     4 - 2     4 - 2     4 9 2
```

The permutation array algorithm, as mentioned above, is implemented as a PS-snippet for `ms3x3` as follows[9]

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Magic Square of order 3. %Permutation array algorithm as given in Wikipedia
%%Creator: Kees van der Laan, okt 2012
%%BoundingBox: 0 0 24 36
%%BeginSetup
%%EndSetup
/Times-Roman 12 selectfont
/p [0 8 1 6
      3 5 7
```

9. `ms3x3`, `ms4x4` and `ms5x5` are included in `PSlib.eps`.

```
      4 9 2] def%0th entry dummy
0 1 2{/i exch def 0 25 i 12 mul sub moveto
      1 1 3{/j exch def p i 3 mul j add get ( ) cvs show 2 0 rmoveto}for
     }for showpage
%%EOF
```

In PSlib I have included a branch-and-bound, backtracking algorithm for order 3.

8 1 6
3 5 7
4 9 2

The finding of symmetrical copies in the branch-and-bound algorithm is suppressed by fixing the middle above element on 1 and restricting the loop variables. The unrestricted code took 38sec and restricted 8sec in Acrobat Pro. PSview took 3sec for the restricted version. The number of magic squares for n=1 is 1, for n=2 there is no magic square and for n=3, 4 and 5 see the accompanying table. Programming the Magic square is as instructive as programming the 8-Queens problem. For the latter see Wirth, N(1976): Algorithms + Datastructures = Programs, p143. Programming Magic squares yields extra Math insight.

n	M_n	Number
3	15	1
4	34	880
5	65	275,305,224

For double-even squares the fun algorithm reads ... All the numbers are written in order from left to right across each row in turn, starting from the top left corner. Numbers are then either retained in the same place or interchanged with their diametrically opposite numbers. In the magic square of order four, the numbers in the four central squares and one square at each corner are retained in the same place and the others are interchanged with their diametrically opposite numbers.

$$\begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{matrix} \rightarrow \begin{matrix} 1 & 15 & 14 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 3 & 2 & 16 \end{matrix} \rightarrow \begin{matrix} 1 & 15 & 14 & 4 \\ 12 & 6 & 7 & 9 \\ 8 & 10 & 11 & 5 \\ 13 & 3 & 2 & 16 \end{matrix}$$

The Magic square of Dürer shows more than the usual properties: also the four quadrants add up to the magical constant 34. By adding up 2 to each cell the magic constant becomes 42, the answer to the question of 'Life, Universe, and Everything.'[10]

16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1

```
\oldstyle
\halign{\quad\hfil#\hfil&&\quad\hfil#\hfil\cr
        16& 3&  2& 13\cr
        5& 10& 11&  8\cr
        9&  6&  7& 12\cr
        4& 15& 14&  1\cr}
```

In PWT the \btable macro was used with flexibility with respect to the frame and the horizontal and vertical lines. Syntactic sugar?

Frans Goddijn calls \oldstyle numbers 'dartele cijfertjes.'[11] A nice Dutch word, dartel.

Knuth's most beautiful tables

Knuth's useful and most beautifully structured and parametrized mark-up of font tables is worth studying. Knuth's macros have been incorporated in BLUe.tex. In the Metafont book in App H a similar but interactive program testfont.tex is available and when TEXlive has been installed one can just say \inputtestfont.tex.

10. Adams, D(1982): The Hitchhiker's guide to the Galaxy. Pan Books.
11. Goddijn, F(1998): Dartele cijfers: poor man's oldstyle, MAPS 20.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	"0x
´01x	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl	
´02x	ı	ȷ	`	´	ˇ	˘	¯	°	"1x
´03x	¸	ß	æ	œ	ø	Æ	Œ	Ø	
´04x	`	!	”	#	$	%	&	’	"2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	"3x
´07x	8	9	:	;	¡	=	¿	?	
´10x	@	A	B	C	D	E	F	G	"4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	"5x
´13x	X	Y	Z	[	“	]	^	˙	
´14x	‘	a	b	c	d	e	f	g	"6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	"7x
´17x	x	y	z	–	—	”	~	¨	
	"8	"9	"A	"B	"C	"D	"E	"F	

For use with pdfTEX the mark-up reads as given below.
LaTEX and ConTEXt users are not aware of this mark-up, I presume, but they might benefit from it.

```
\input blue.tex
\beginchart{\postdisplaypenalty=0
\tenrm}
%or \tenit ... \tenlmr?
\normalchart
\endchart
\bye
```

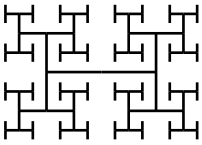
```
%or simply
\input testfont.tex
%a prompt for font name
%appears: type cmr10 f.e.
\table
\bye
```

Font tables have been supplied in the TEXbook, Appendix F. In Appendix H of the Metafont book.`testfont.tex` is discussed[12]

I was curious how I could obtain a font table for Latin Modern Roman. Hans Hagen prompted `\starttext\showfont[lmroman10regular][all]\stoptext`, which I processed under Context(LuaTeX) in TEXworks. Pane 1 of the table is supplied in the 3rd Appendix.

H-fractal from PWT

Earlier I remarked that the binary tree, the H-fractal and Adobe's `FractArrow`, Bluebook, p.74, are closely related, one just has to adapt the invoke by the appropriate angle.



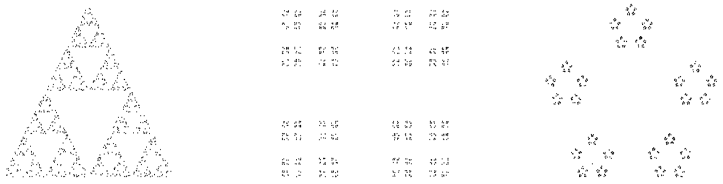
In `BLUe.tex` I implemented the Turtle Graphics approach. The H-fractal was programmed recursively and supplied as exercise 5.3 in PWT. Apart from pictures generated on-the-fly, pictures are provided in `pic.dat`, the picture-base of TEX-alone pictures which comes with `BLUe.tex`.

Compared with programming in PostScript the coding of a TEX-alone picture is cripple, without the possibility to crop the result, to include BoundingBox values for pdfTEX. There is no need to include the H-fractal `gkp`-codes and PS-codes here; they have been included in the EuroTEX-ConTEXt2009 proceedings. I also mentioned there the notches, the absence of appropriate line-endings in TEX. TEX is the wrong tool for graphics, definitely. But ... in cooperation with Metafont artistic effects can be obtained, as was done by Jackowski&Ryćko in the early 90s. For simple, quick-and-dirty, line-drawings TEX might do.

Iterated Function System fractals from PWT

In 1989 I attended the TUG conference where Alan Hoenig showed some iterated function system fractals,[13] which I reproduced in PWT. The idea is that the points within an n-gon are created by: the mean of a random point within the n-gon and one of its corners at random, à la Monte Carlo. A random number generator for plain TEX had to be written. The representation of the corner points is tricky via `\newdimen`-variables, in order to perform the arithmetic. Too much details in order to be presented here.

12. Lueking, D(2010): How to use `fntproof.tex` and `testfont.tex` (from the WWW).
13. His paper has been published in TUGboat 1989. For iterated function systems and fractals, see Peitgen c.s. (2004 2nd ed): Chaos and Fractals. Springer. No sophisticated Math is required for reading the book.



Pascal triangle from PWT

The table chapter of PWT contains the Pascal triangle. The triangle shows the binomial coefficients $\binom{n}{k}$.
If the values of $\binom{n}{k}$ are available, the typesetting is a trifle via the use of `\displaylines`, TEXbook, p362.
The values $\binom{n}{k}$ can be generated on-the-fly by the recursion

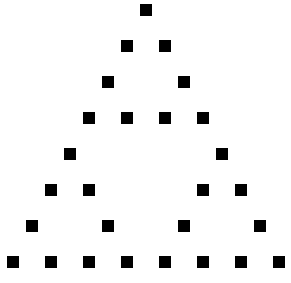
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \quad n = 2, ..., \quad k = 1, ..., n-1, \quad \binom{n}{0} = \binom{n}{n} = 1$$

which has been used in the code as shown in the verbatim text at right below. The intriguing macros make use of recursion for calculating each element in a row. Each row is overwritten in `\1,\2,...` which also entails that each 'row' is extended dynamically. This reminds me of the dynamic array functionality.
Paradigm: a counter-value becomes a control sequence name to denote the position in the row with as value the binomial coefficient.

```
$$\displaylines{1\cr
1 \quad 1\cr
...
1 \quad 9 ... 9 \quad 1\cr}$$
```

```
\newcount\n \newcount\rcnt \newcount\ccnt \newcount\tableentry \newcount\prev
%
\def\pascal#1{\n#1 \def\0{1}                                    %presets
   \ccnt1 \loop\expandafter\xdef\csname\the\ccnt\endcsname{0}
   \ifnum\ccnt<\n \advance\ccnt1
   \repeat  \rcnt0 \ccnt0 \displaylines{\rows}}
%
\def\rows{\global\advance\rcnt1 \ifnum\rcnt>\n \swor\fi \nxtrow\rows}
\def\swor#1\rows{\fi}
%
\def\nxtrow{1 \ccnt1 \prev1
 \loop\ifnum\ccnt<\rcnt \tableentry\prev \prev\csname\the\ccnt\endcsname
 \advance\tableentry\prev
 \expandafter\xdef\csname\the\ccnt\endcsname{\the\tableentry} %the new entry
  \quad\the\tableentry \advance\ccnt1                        %show the entry
 \repeat\cr}
$$\pascal{8}$$
```

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1 \\
 & 5 & & 10 & & 10 & & 5 & \\
1 & & 6 & & 15 & & 20 & & 15 \\
 & & & & \cdots & & & &
\end{array}$$



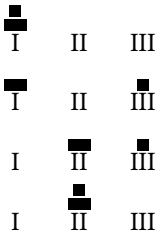
In the picture at right a variant of the PACAL triangle has been shown, where the odd-valued entries are coloured black and the even-valued entries are left blank, which reminds me of the Sierpiński triangle. The macros and pictures have been submitted to GUST's Programming Pearls 2012.

Towers of Hanoi play from PWT

In general BLUe has as top level minimal one-part macros and tries to circumvent the curly braces mania: no curly braces around arguments! Invoke: `\Hanoi\I\II\III\n`,

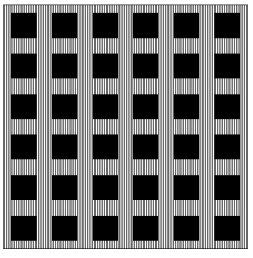
where the capital Roman numerals denote the towers. The one-part macro invokes the two-part macros, the environment. The Hanoi macros are available within the `\beginhanoi...\endhanoi` environment.

The process of replacement of the disks will also be printed by the shortened invoke of the one-part macro `\sethanoi<n>`, `<n>` an integer, the height of the initial tower. The intermediate stages will be shown, no user mark-up is needed.
The Hanoi-tool has been copied from BLUe's `tools.dat` and is used stand-alone in this paper to reproduce the results.
Paradigm: the use of a hidden loop counter. The loop counter is dynamically created in `\preloop`; the user is not bothered by it.



Soto's Op Art from PWT

A verbose version of Soto's Op Art[14] emulation was written originally in Metafont in 1995. For the EuroTEX-ConTEXt2009 the picture was redone in concise plain TEX, with the use of TEX's `\leaders`, `\xleaders` and the reuse of `\setbox`-es. A gkp-macro version appeared earlier in PWT. On occasion of this conference a simpler, mean-and-lean PS-variant has been written.



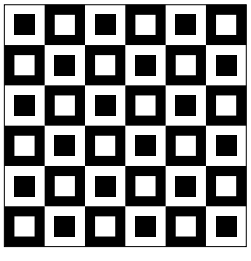
```
\def\boxit#1{\vbox{\hrule
           \hbox{\vrule#1\vrule}\hrule}}
\newbox\cb  \newdimen\ul \ul=6ex
\newdimen\size \size12\ul
\setbox\cb\vbox to2\ul{\vss
    \hbox to2\ul{\hss\vrule height1.2\ul width1.2\ul
                 \hss}%
                      \vss}%
$$\boxit{\vbox{\offinterlineskip
\hbox{\xleaders\hbox to.5ex{\hss\vrule height\size
                           \hss}\hskip\size}%
\kern-\size%setback
\leaders\hbox{\leaders\copy\cb\hskip\size}\vskip\size}}$$
```

```
%!PS-Adobe
...
gsave .25 setlinewidth
57{1 0 translate
   0 0 moveto 0 57 lineto}repeat
stroke
grestore
0 0 57 57 rectstroke
3 3 translate
6{gsave
  6{0 0 6 6 rectfill
    9 0 translate}repeat
  grestore
  0 9 translate}repeat
```

In TEX we strive after efficiency by using `\setbox`-es, such that repetitive material is only typeset once and reused by copying; in PS there is the ucache concept. My PS-graphics is small and fast enough.

Jiggling squares from PWT

An old example, which I did in TEX, see EuroTEX-ConTEXt2009, and on occasion of EuroTEX-ConTEXt2012 in PostScript, in a split-second.



```
%!PS-Adobe
...
gsave .1 setlinewidth 0 0 120 120 rectstroke grestore
/Oc{0 0 moveto 20 0 rlineto 0 20 rlineto -20 0 rlineto closepath}def%outer contour
/Ic{5 5 moveto 0 10 rlineto 10 0 rlineto 0 -10 rlineto closepath}def%inner contour
3{gsave 3{Oc Ic fill 20 0 translate
          Ic    fill 20 0 translate}repeat
 grestore
 gsave 0 20 translate
        3{Ic fill    20 0 translate
          Oc Ic fill 20 0 translate}repeat
 grestore 0 40 translate}repeat
showpage
```

14. Jesús Rafael Soto, 1923–2005, was a Venezuelan Op and Kinetic artist, a sculptor and a painter. Soto has created penetrables, interactive sculptures which consist of square arrays of thin, dangling tubes through which observers can walk. It has been said of Soto's art that it is inseparable from the viewer; it can only stand completed in the illusion perceived by the mind as a result of observing the piece. http://www.wikipedia.org/wiki/Jesus-Rafael_Soto.

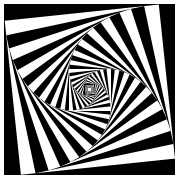
Do realize the use of integers only in the model, otherwise rounding errors spoil the strict regularity. No lines are drawn, except for the border, just `fill`s filled by the non-zero winding rule. Pitfalls for the unwary.
One element of the tile is a square O, programmed by a `0c 1c fill`. How is the O programmed in the Metafont book? From p303

```
beginlogochar("O",15); x1=x4=.5w; top y1=h+o; bot y4=-o; x2=w-x3=good.x(1.5u+s);  y2=y3=barheight;
                       super_half(2,1,3); super_half(2,4,3); labels(1,2,3,4); endchar;
```

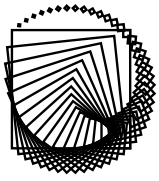
Remarkable is that 2 halves of a superellipse have been used, not just the complete inner and outer contours, apparently for consistency with the upper half of the letter A. On p32 the character O has been drawn by the use of `penstroke`.

Shrinking squares

At last I traced the origin of the left ubiquitous illustration, which is about drawing just squares in transformed user space: Barnsley, M.F(1988): Fractals Everywhere. It is used in Ch. 3.6 to illustrate the idea of a contractive transformation on a compact metric space.



```
%!PS-Adobe-3.0 EPSF-3.0
...
/r 50 def /2r {r r add} def /-r {r neg} def
/alpha 6 def /c 1 alpha cos alpha sin add div def
/square{-r -r 2r 2r rectfill}def
0 1 45{2 mod setgray square
       /r r c mul def alpha rotate}for
```



The right picture is an intriguing variant; nearly the same code.
Paradigm: change of black and white in traversals of the loop.

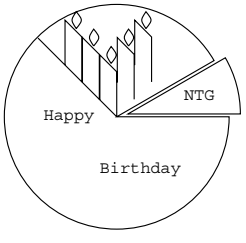
Malevich suppremátism

Malevich Suprematism:
White cross on a White background
Emulation →

I have included a picture, and its emulation, of Malevich's[15] 'White Cross on a white background,' because he is the father of suprematism, which deletes the superfluous, which I associate with Minimal Mark-up. But ... sometimes redundancy is beneficial.

Happy birthday NTG from PWT

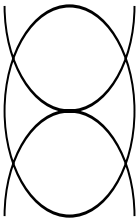
This cake I produced on occasion of the first lustrum of NTG. The original version made use of LATEX's picture environment and does no longer work on my system since I abandoned LATEX. The 2nd version was done with the gkp-macros.[16] At the time I experienced drawing a circle by splines as difficult.[17] The 3rd version of the picture is in PostScript, shown at right, is a trifle and took me a couple of minutes.



15. Kazimir Malevich, 1878–1935. Russian painter, born in Kiev.
16. I should make it priority 1 to get `BLUe.tex` running again as a format in TEXworks.
17. For the solution à la Knuth, see the Metafont book p263, or Appendix 1 in Gabo's Torsion, MAPS 42, 2011.

Logo from PWT and a logo from MetaFun

The logo was created by the gkp-macros. The current version in PostScript took me just a minute. The proportions obey the golden ratio, realized by scaling. At right a logo borrowed from MetaFun.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: cgllogo, 2012
%%BoundingBox: -63 -101 63 101
%%BeginSetup
%%EndSetup
/r 100 def /-r r neg def 3 setlinewidth .618 1 scale
r 0 moveto  0 0 r 0 360 arc
2{r -r moveto 0 -r r 0  180 arc 1 -1 scale }repeat
stroke
```

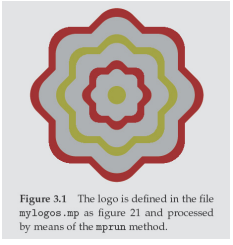
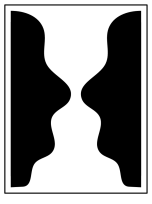


Figure 3.1 The logo is defined in the file `mylogos.mp` as figure 21 and processed by means of the `mprun` method.

Profiles or a candle?



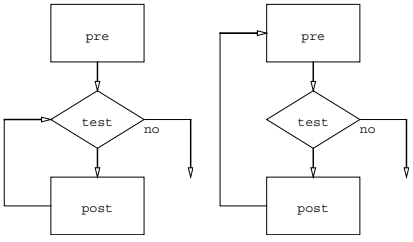
```
size:=210;path p[];
p1=(-1.1size,-1.5size){right}---(-.9size,-1.49size)
    ..(-.7size,-1.1size)..(-.4size,-.9size)..
   (-.4size,-.3size)..(-.1size,0)..(-.5size,.6size)..(-.55size,1.2size)...
   {left}(-1.1size,1.5size)---cycle;
p2= p1 reflectedabout ((0,-size),(0,size));
fill p1; fill p2;
draw (-1.2size,-1.6size)---(1.2size,-1.6size)---(1.2size,1.6size)---
     (-1.2size,1.6size)---cycle;
```

The Metafont code of 1996 was converted into MetaPost, well simply stripped from Metafont's necessities such as screen settings, `cullit`, `show`, ..., and dropped onto Troy Henderson's `mppreviewer` to yield `.png`, on occasion of EuroTEX-ConTEXt2012.[18] Happily, I saved `profile` from disappearance.

Flowchart from PWT

In PWT the flowchart of TEX's loop was made within LATEX's picture-environment and later converted into plain TEX where the picture was created by the gkp-macros. On the occasion of EuroTEX-ConTEXt2009 the flowchart was created in MetaPost by means of Hobby's `boxes` macros, and resulted in a PostScript program created by MP. The right picture mimics TEX's loop.[19] At right my PostScript code on occasion of EuroTEX-ConTEXt2012. No `boxes` macros are needed; the coding is equally simple, or equally difficult, depending on your expertise, as the MetaPost code.
The recent picture, with golden ratio proportions, took me 45min to create in PostScript, which is too long for a production tool. The creation of the previous loop-pictures took me at least as long, if not longer ☹. Use is made of `rectstroke`, `centershow` and Adobe's Bluebook `arrow`. Ça va sans dire that the direct PostScript program is much shorter than the PostScript code which resulted from MetaPost.

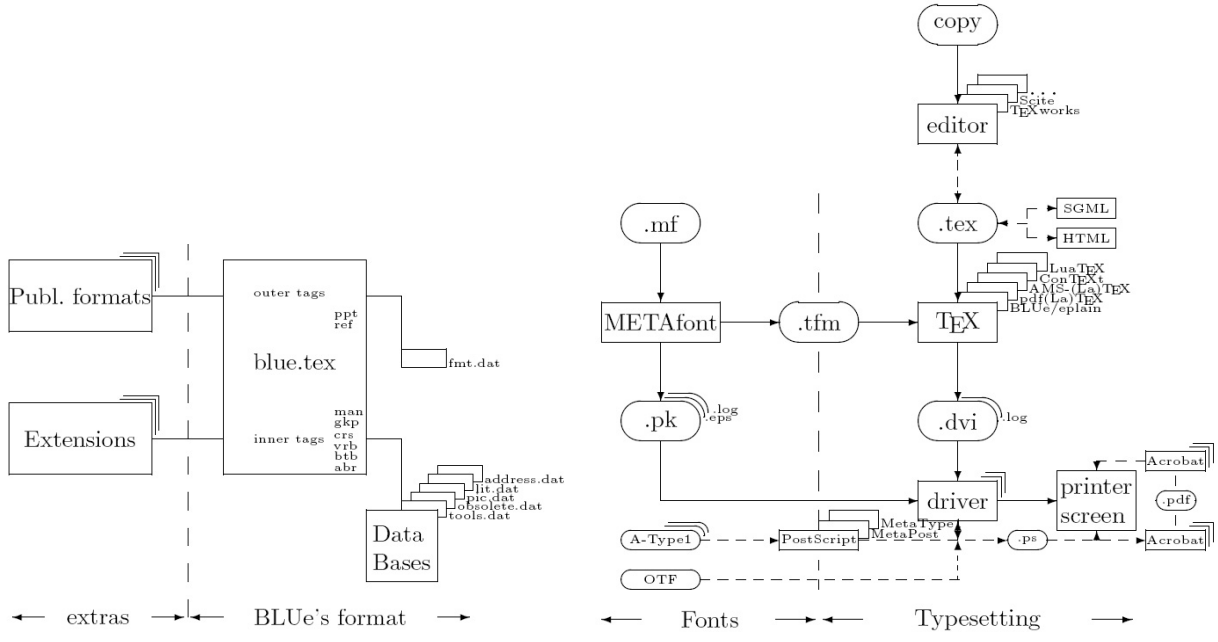
18. I used Hans Hagen's pair `\startuseMPgraphic{dummy}`, `\stopuseMPgraphic` and `\useMPgraphic{dummy}` and got results by ConTEXt(LuaTEX) in TEXworks. I no longer need Henderson's MPpreviewer. Troy has also provided a LaTEX and a function-grapher previewer.
19. In contrast with the usual implementations, `while` respectively `repeat` ... `until`, where the test is performed at the beginning respectively at the end, the test for termination in TEX's loop can be placed at a place at will within the loop. The same is possible in PostScript within the `loop` procedure, where termination goes via the invoke of `exit` (for the inner loop). I consider the implementation of TEX's loop ingenious.




```
/Courier 14 selectfont
/s 50 def  /t s .61803 mul def %half width and halh height
/2s s s add def/2t t t add def/3t 2t t add def /6t 6 t mul def
/-s s neg def /-t t neg def /-2s 2s neg def /-2t 2t neg def
%bottom rectangle
-s -t 2s 2t rectstroke
0 -7 moveto (post) centershow
-s 0 moveto -2s 0 lineto 0 6t rlineto
-2s 6t -s 6t .5 5 10 arrow stroke                              %top-down arrow
%diamond
0 3t translate
-s 0 moveto 0 t lineto s 0 lineto 0 -t lineto closepath stroke%diamond
0 -7 moveto (test) centershow
0 2t 0 t .5 5 10 arrow stroke                                  %top-down arrow
0 -t 0 -2t .5 5 10 arrow stroke                                %arrow on the
left
s 0 moveto 2s 0 lineto 2s 0 2s -2t .5 5 10 arrow  stroke       %arrow on the
right
s -14 moveto (no) show
%top rectangle
0 3t translate -s -t 2s 2t rectstroke 0 -7 moveto (pre) centershow
```

The `flowchartloop def` is included in `PSlib.eps`. Do compare the three generations of code: based on the picture environment, MetaPost (both listed in the EuroTEX-ConTEXt2009 paper), and the PostScript code given here. My conclusion is, and was, that PostScript can equally-well be used directly in a 1-pass job, most of the times.

Below my most complex TEX-made flowcharts of old. The TEX-MF picture has been adapted for this conference.



The above TEX-flow is nowadays practically integrated in simple to use IDE's, such as TEXworks, where the various processing modes, such as pdfTEX, can be chosen from pull-down menus.[20]

TEXworks shows 3 panels: the edit panel with the source `.tex`, the result panel with `.pdf`, and the processing window with the process report and the error messages,

20. Blue Sky provided 20 years ago similar functionalities in its TEXtures for the Macintosh.

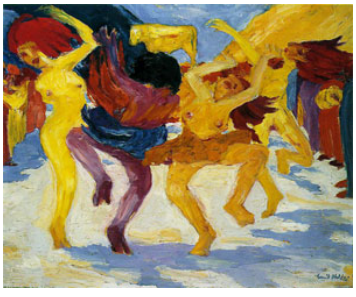
eventually. A form of WYSIWYG. Another pull-down menu in the edit panel lets you choose your font and the use of spelling checkers. There is also a script option. In the help menu there is an option for the 'Short manual for TEXworks' by Alan Delmotte, Stefan Löffler, and others.

Font Fun in TEX&Co

Though TEX's CM-fonts are bitmapped and rigged, occasionally recreational effects have been obtained.

Dancing texts by PostScript

Hans Hagen in his MetaFun inspired me to think about dancing texts. In PostScript the effect can be obtained by the use of kshow, where the procedure as argument of kshow takes care of (slightly) rotating user space for each character. Nrand delivers a random number $\in [0,1)$, and unirand a random number $\in (-1,1)$, both from PSlib.eps. The colours are composed randomly. In Photoshop dancing-like texts can be obtained by typesetting along a sine-curve. The picture at right is by Emil Nolde.[21]
Avoid in PostScript the trap to create a font variant.
A nice application for children's party invitation cards.



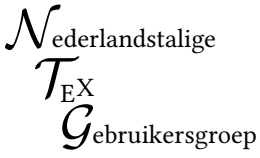
Kees van der Laan

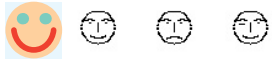
```
07101951 srand
/Helvetica 35 selectfont 0 0 moveto 1 0 0 setrgbcolor
{pop pop unirand 4 mul rotate nrand nrand nrand setrgbcolor}
(Kees van der Laan) kshow %a paradigm
```

21. Emil Nolde, 7 August 1867 Near Nolde (Denmark) – 13 April 1956 Seebrüke, was a German painter and printmaker. He was one of the first Expressionists, a member of Die Brücke, and is considered to be one of the great oil painting and watercolour painters of the 20th century. He is known for his vigorous brushwork and expressive choice of colours. Golden yellows and deep reds appear frequently in his work, giving a luminous quality to otherwise somber tones. His watercolors include vivid, brooding storm-scapes and brilliant florals. *'There is silver blue, sky blue and thunder blue. Every colour holds within it a soul, which makes me happy or repels me, and which acts as a stimulus. To a person who has no art in him, colours are colours, tones tones...and that is all.'*

Font Fun in TEX

The classical example is the word TEX, with dropped E. Another classic is XETEX which can't be done in TEX-alone. With dvips the mirroring can be done at the PS-level, but alas pdfTEX does not allow for PS. NTG's first logo was 'Nederlandse TEX Gebruikersgroep,' which was soon changed, on the way to the TUG meeting at Karlsruhe in discussion with Johannes Braams, into 'Nederlandstalige TEX Gebruikersgroep,' meaning Dutch-language based.



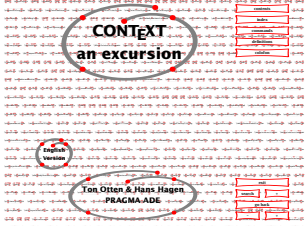


The coloured smiley is in PostScript; the others are done in TEX by dots; very cripple TEX programming.

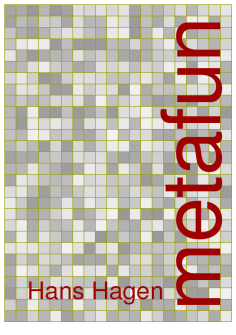
←EuroTEXConTEXt2012 logo

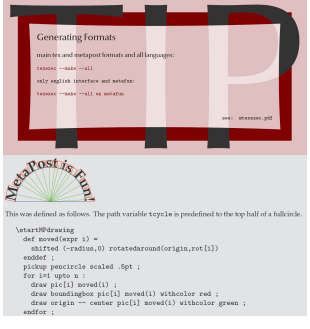
Toruń98 logo, by B. Jackowski→

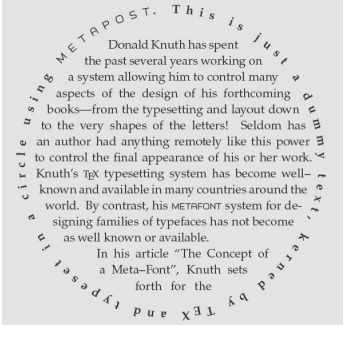
The TEX-lion and the MF-cat, by Duane Bibby, the running illustrations in the TEXbook and the MetaFontbook, made the books a pleasure to read. The EuroTEX-ConTEXt2012 logo is nice and fun.



Hans Hagen's MetaFun







Don Lancaster's Font Fun

In the late 1970s TEX appeared with rigid, bitmap CM font families. In the mid 1980s Adobe developed scalable, and adaptable PostScript fonts, thanks to the font matrix concept.[22] Lancaster played 20 years ago with PostScript's font variability. In his `pssecrets`[23] he showed various font variations. These modified fonts can be used in PostScript, which is fair enough. Adobe Type 1, with `afm2tfm` for conversion of the metrics, can be used in TEX, although this route becomes more and more outdated in view of that Adobe has Adobe Type 1 declared obsolete and in view of unicode and the TEX-Gyre OTF-project, next to the incorporation of Open Type Fonts in the new TEX-engines LuaTEX, or XƎTEX.[24]

reverse

emboss

shiny

3D

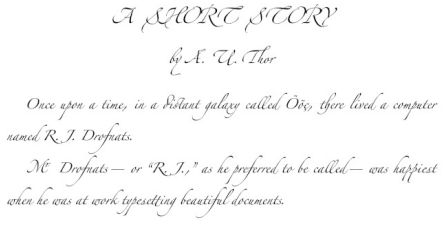
The tiny PostScript program for the shadowfont is not at all difficult and demonstrates the use of the font matrix, TFM for short.[25] The reverse font is straightforward too with TFM: e.g. `[-40 0 0 40 0 0]`.

rotated

shadow

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Shadow font, Don Lancaster, 1990
%%BoundingBox: -1 -25 180 30
%%BeginSetup
%%EndSetup
.8 setgray  /msg (Free font) def
/Palatino-Bold findfont [40 0 32 -30 0 0] makefont
setfont
 0 0 moveto msg show%shadow
 0 setgray /Palatino-Bold 40 selectfont
 0 0 moveto msg show
```

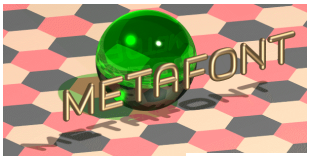
From the XƎTEX showcase




```
\font\body="Zapfino" at 10pt \body
\font\title="Zapfino:Stylistic Variants=First variant glyph set" at 12pt
\font\author="Zapfino:Stylistic Variants=Second variant glyph set" at 10pt
\centerline{\title A \ SHORT \ STORY}
\vskip 6pt
\centerline{\author by A. U. Thor}
\vskip .5cm
Once upon a time, in a distant galaxy called Ööç,
there lived a computer named R.~J. Drofnats.

Mr.~Drofnats---or ``R. J.,'' as he preferred to be called---was happiest
when he was at work typesetting beautiful documents.
\bye
```

Metafont&TEX can be used to create beautiful artistic results with fonts as has been shown in the 90s by Bogusław Jackowski and Marek Ryćko. Non-scalability is not relevant for pieces of art.

22. Making outline fonts from TEX's CM fonts is not simple, while outline variants of PS fonts are a trifle. Using font outlines for clipping is fun in PS. In TEX I don't know how to do it.
23. `http://wwwtinaja.com/glib/pssecrets.pdf`. The layout of his tiny programs is horrible. The ones I copied I have simplified.
24. Veith, U, M. Miklavec(2012): Another incarnation of Lucida: Towards Lucide OpenType. BachoTEX2012 proceedings, 5–13. Ludwichowski, this proceedings.
25. The font matrix is specified by 6 digits between square brackets, similar to the general transformation matrix, TFM for short, of PostScript.

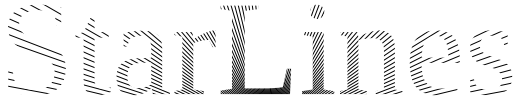


Word Art in MS-Word

My Word 7 comes with `Word Art` options and the user can play with the appearance of texts.
My wife Svetlana Morozova tipped me about the font fun in Word. In Photoshop similar effects can be obtained. TEX's bitmap CM fonts are too rigid for fun.

Font outlines

In TEX and MetaPost the creation, and the clipping use, of an outline of a glyph is not possible. In PostScript it is part of the orthogonal philosophy: any path, including the character path left by a `charpath` operator, can be used as a clipping outline boundary.[26] The `def o(utline)show`, with on the stack a `(string)`, reads as follows `/oshow{true charpath stroke}def`.
`Clipping` of an outline path may yield interesting effects. The example is borrowed from the Bluebook p103.




```
/Times-Roman 50 selectfont .25 setlinewidth
/rays{120{0 0 moveto 108 0 lineto
      1.5 rotate}repeat stroke}def
0 0  moveto (StarLines) true charpath clip
newpath 100 -15 translate rays
```

Carving

Another nice example in Hans' `MetaFun` is the tallying of data. I imitated his ConTEXt–MetaPost table example. My tallying is done in PostScript, see the code below at right, and the table is set via `\halign`.

System	%	Users
Atari	10.4	𝍸 𝍸 |
MS-DOS	49.1	𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 ||
OS/2	9.4	𝍸 𝍸
MacOS	5.7	𝍸 |
UNIX	51.9	𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸
Windows	64.2	𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 |||

```
/tally{/n exch def 0 0 moveto
1 1 n{dup 5 mod 0 eq{-8 0 rmoveto /d 10 nrand sub def
                      7 d rlineto  4 d neg  rmoveto}
                     {/r unirand 3 mul def
                       r rotate 0 10 rlineto
                       r neg rotate
                       2 -10 rmoveto} ifelse
}bind for
}def
```

Hans' 1-pass job has much in favour. I like, of course, my cooperating tools approach. I don't have to remember the philosophy and details of ConTEXt, MetaPost, nor Metafun; just good old plain TEX and PostScript. The tallying macro and the dancing text were written on occasion of EuroTEX-ConTEXt2012.

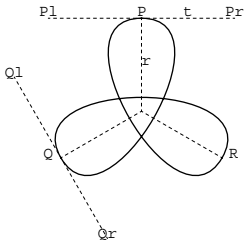
26. This works only for characters which are defined as outlines.

Escher knot

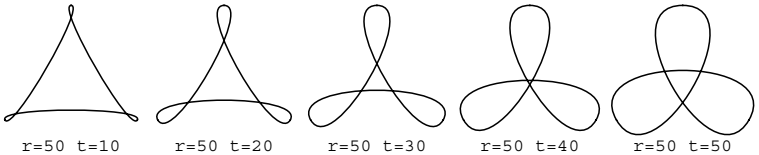
The Escher knot was programmed in Metafont and MetaPost. It marks my beginning of using Metafont/Post as declarative graphical languages. From the latter program the spline data were distilled and inserted in the tiny PostScript program below, with the number of fractional decimals rounded to 2. The gradient colouring has been done in Photoshop by my wife Svetlana Morozova on occasion of the EuroTEX-ConTEXt2009.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -40 -31 40 43
%%Creator: MetaPost and JJW, CGL, June 1996
%%BeginSetup
%%EndSetup
3{-21.65 12.5 moveto
-21.65 27.75 -13.78 42.50  0 42. 50    curveto
 13.78 42.50  21.65 27.75 21.65  12.5  curveto
 21.65 -0.24 16.05 -12.19  6.58 -20.33 curveto
-14.32 15.87 moveto
-12.43 23.59 -7.45 29.75  0   29.75 curveto
  9.65 29.75 15.16 19.42 15.16 8.75 curveto
 15.16 -2.08 9.38 -12.09  0  -17.5  curveto
120 rotate
}repeat stroke
```

While pondering about the Escher knot another solution came to mind for the single knot in PostScript.




```
/Courier 12 selectfont
/r 70 def /t 70 def
/P{0 r}def /Q{P 120 rot}def /R{P -120 rot}def
/Pr{t r}def /Pl{t neg r}def /Qr{Pl 120 rot}def
gsave .1 setlinewidth [2 3] 11 setdash
   3{0 0 moveto P lineto 120 rotate}repeat stroke grestore
P 2 add moveto (P) centershow
Q exch 13 sub exch moveto (Q) show
R exch 5 add exch moveto (R) show
3{P moveto Pr Qr Q curveto 120 rotate} repeat stroke
```



An alike of the third figure in MP reads

```
beginfig(59)
draw (0,u)      {right}   .. tension 4..
     (u*dir-150){dir 120} .. tension 4..
     (u*dir-30) {dir/120} .. tension 4..
     cycle
endfig;
```

The variability by `r` and `t` seems sufficient.[27] The 'tube' version is complicated by hidden lines, which were gracefully handled in MetaPost in the EuroTEX-ConTEXt2009 paper, by use of `cutbefore` and `cutafter`. The single knot version has less graceful curves. The shape can be adapted by changing `r` and/or `t`. It looks like that Metafont's `tension` functionality is not needed. BTW, I much prefer for a curve in Metafont's lingo `z0..controls z1 and z2..z3`, more in accordance with PS's `curveto` and the Math formula $\sum_{i=0}^{3}(1-t)^{3-i}t^i z_i$, without the strange unusual notions `tension` and `curl`.

In PostScript there is no path data-structure and no `def` for calculating the intersection point of 2 B-cubics. It is curious that PostScript does not contain an evaluation procedure for points on a spline. The 'de Casteljau' algorithm for evaluation is nothing more than fixing precedence of operations by parentheses

27. Dennis Roegel has published many articles on MetaPost.

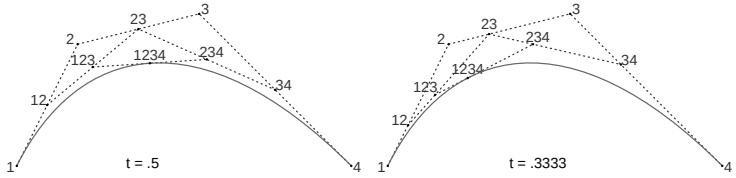
$$z(t) = \sum_{i=0}^{3} z_i(1-t)^{3-i}t^i = (1-t)\Big((1-t)\Big((1-t)z_0 + t\,z_1)\Big) + t\,\Big((1-t)z_1 + t\,z_2)\Big)\Big) + \\ + t\Big((1-t)\Big((1-t)z_1 + t\,z_2)\Big) + t\,\Big((1-t)z_2 + t\,z_3)\Big)\Big).$$

Together with my `solveit` it should not be difficult to write a `def splineintersection`.

Not recreational, pretty serious. Maybe, someday, sometime ...
Knuth in the MetaFont book, p13, gives a graphical representation of the evaluation, mediation algorithm

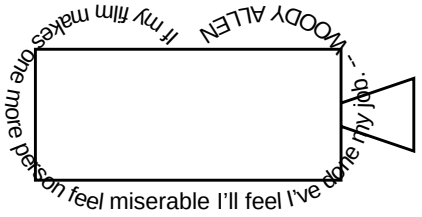
$$\begin{matrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{matrix} \begin{matrix} \\ \rightarrow \\ \\ \end{matrix} \begin{matrix} z_{12} = \frac{1}{2}[z_1, z_2] \\ z_{23} = \frac{1}{2}[z_2, z_3] \\ z_{34} = \frac{1}{2}[z_3, z_4] \end{matrix} \begin{matrix} \\ \rightarrow \\ \\ \end{matrix} \begin{matrix} z_{123} = \frac{1}{2}[z_{12}, z_{23}] \\ z_{234} = \frac{1}{2}[z_{23}, z_{34}] \end{matrix} \rightarrow z_{1234} = \frac{1}{2}[z_{123}, z_{234}]$$

where $\frac{1}{2}[z_1, z_2]$ means the midpoint of the line through z_1 and z_2.
To get the remaining points of the curve determined by z_1, z_2, z_3, z_4 repeat the same construction on $z_1, z_{12}, z_{123}, z_{1234}$ and $z_{1234}, z_{234}, z_{34}, z_4$, ad infinitum.



The fixing of the precedence of the operators by parentheses is the simplest way to describe the algorithm.

Text along paths, Adobe's pathtext, Bluebook p167




```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Blue Book Program 11, page 171
%%BoundingBox: 126 285 412 435
%%BeginSetup
%%EndSetup
(C:\\PSlib\\Bluebook.eps) run
/Helvetica 16 selectfont 2 setlinewidth
150 310 210 90 rectstroke
360 347 moveto 410 330 lineto 410 380 lineto 360 363 lineto stroke
200 360 70 0 270 arc  200 110 add 360 70 270 180 arc%path for text
(If my film makes one more person feel miserable\
 I'll feel I've done my job.-- WOODY ALLEN) 55 pathtext
```

I would not dream of trying to do this picture in TEX. Adobe treated us on a nice, powerful PostScript `def pathtext`. But ... sometimes we can do without it.

8 March, MAPS 42, 2011

There were several problems which had to be solved in order to achieve the present. First, typesetting along a lemniscate, which was done by Adobe's `pathtext` BLUebook p168. Next the typesetting of Cyrillic in PostScript. This was done by Adobe's `ReEncodeSmall`, Bluebook p207, after I found a font with Cyrillic glyphs. Third, inclusion of `.jpg` photographs in an EPSF program, which was done after the `.jpg` was saved as EPSF in Photoshop.

Powerful `pathtext` is, but ... we can do without `pathtext` when the path is implicit.

CD-DVD lables, MAPS 43, 2012

Adobe's 85 Bluebook

CD label

⟶

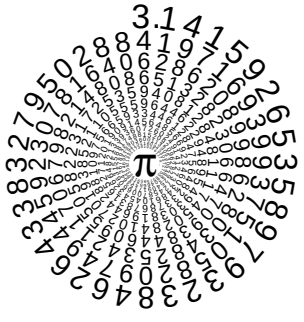
Adobe's 85 Bluebook CD label enriched with notes background

Essentially, it is Adobe's example program from the Bluebook p163, about printing along circular arcs. I have enriched the CD-label by a background, where the `.jpg` picture has been converted into EPSF.[28]

π-decimals

A nice printing along an (infinite) implicit spiral is π-decimals. Special is that it has been done without using the page-builder, as Pawel Jackowski used to say. The spiral path is implicit, no explicit path has been built up nor is `pathtext` invoked. It has been published as a GUST programming Pearl 2010. Below a slightly adapted version because `PSView` and `Acrobat` yielded different results on the BachoTEX Pearl version.[29]

The `pop pop` in the procedure are there because `kshow` pushes 2 neighbouring values of the string on the stack each time, which we don't use in the procedure. The backslash allows breaking a long string over lines. The picture was borrowed from the CWI-calendar of 1972.[30]



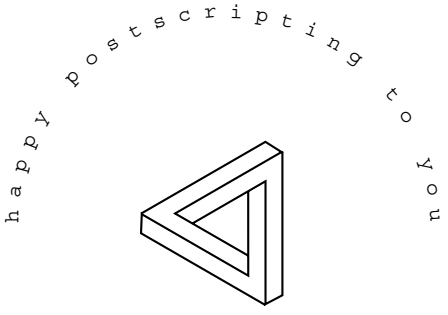
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Pi-decimals along a  Spiral, cgl 2010/2012
%%BoundingBox: -80 -100 100 90
%%BeginSetup
%%EndSetup
/Symbol 26 selectfont 1 -18 moveto (p) show%p denotes pi in the symbol font
/Helvetica 20 selectfont
0 70 moveto (3) show 1 0 rmoveto (.) show -2 0 rmoveto%back space
-10 rotate .995 dup scale
{pop pop -10 rotate 3 0 rmoveto .995 dup scale}
(14159265358979323846264338327950288419716939937510582097494459230781640628\
 62089986280348253421170679821480865132823066470938446095505822317253594081\
 28481117450284102701938521105559644622948954930381964428810975665933446128\
 47564823378678316527120190914564856692346034861045432664821339360726024914\
 12737245870066063155881748815209209628292540917153643678925903600113305305\
 48820466521384146951941511609433057270365759591953092186117381932611 79...)
kshow
```

The calculation of the digits of π is a different matter. For a historical survey see paragraph 3.3 in Peitgen c.s.(2004): Chaos and Fractals, or Beukers, F(2000): Pi, de Geschiedenis en de Wiskunde van het getal π. Epsilon. (In Dutch). It is no surprise that millions of digits could only be calculated because of computers.

28. Willi Egger explained how to use ConTEXt's layers to add a picture as background, EuroTEX-ConTEXt2009.
29. `PSView` and `Acrobat Pro` give sometimes different rotated results on pictures where use is made of rotated User Space. Apparently there is some confusion in implementing rotated user space. Moreover, a rotation over 89.9 degrees and a rotation over 90 degrees yielded significantly different results. This is all circumvented in the new version.
30. Frans Goddijn suggested that it would make a nice poster.

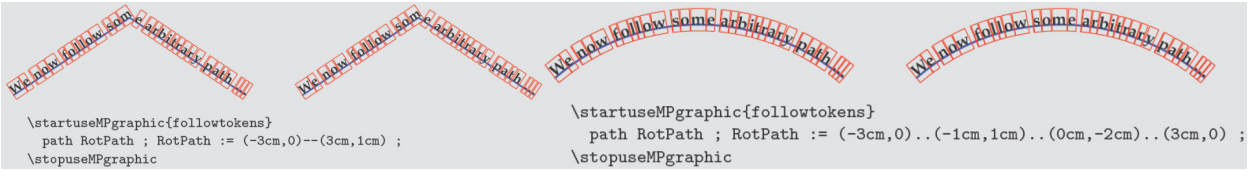
Seal: text along circular arc

The following seal, or text along a circular arc, illustrates the use of kshow, not pathtext. The circular path is implicit, no explicit path has been built up nor is pathtext invoked.[31]
The included, impossible Escher triangle is intriguing. Once the symmetry has been revealed the programming is a trifle. This time the PostScript def, as included in PSlib.eps, is given in the verbatim below. All 40 pictures of the 'Paradigm: Just a little bit of PostScript'- article have been included in PSlib.eps.




```
/textalongarc{%jalxx
/Courier 10 selectfont
/text (happy postscripting to you) def
/r 100 def
gsave
3{25  34 moveto%3 rotated copies of broken line
  25 -34 lineto
  17 -38.2 lineto
  17  20 lineto
 -17.6 0 lineto
120 rotate
}repeat stroke
  90 rotate  %begin orientation
  0 r moveto %begin point
{pop pop -7.04 rotate 6 0 rmoveto} text kshow
grestore
}def
```

Texts along arbitrary paths in ConTEXt interfaced with MetaPost




```
\startuseMPgraphic{followtokens}
  path RotPath ; RotPath := (-3cm,0)--(3cm,1cm) ;
\stopuseMPgraphic
```

```
\startuseMPgraphic{followtokens}
  path RotPath ; RotPath := (-3cm,0)..(-1cm,1cm)..(0cm,-2cm)..(3cm,0) ;
\stopuseMPgraphic
```

I tried a 1-liner MP-interfacing program from the MetaFun manual in TEXwork's ConTEXt(LuaTeX):

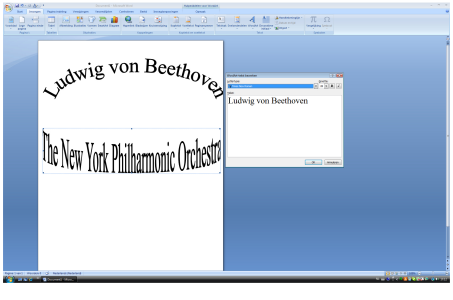
```
\starttext\startuseMPgraphic{dummy} fill fullcircle scaled 5cm withcolor red;\stopuseMPgraphic
\useMPgraphic{dummy} \stoptext
```

My first ConTEXt run! There is still hope in angry days for BLUe ... .

Professional Circular Text by Photoshop and Word

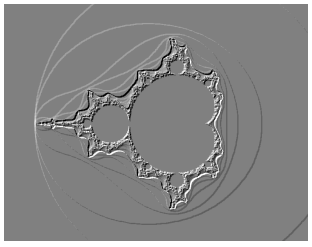
← Photoshop

Word→



31. If you want to do this in TEX&Metafont alone consult Hoenig, A(1989): Circular Reasoning: Typesetting along a circle and related issues, TUGBoat11, or easier consult the digital 24hrs library http://www.tug.org/TUGboat/tb11-2/tb28hoenig.pdf.

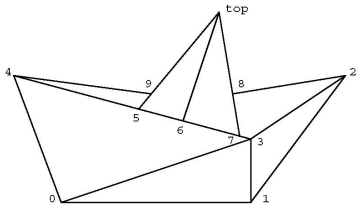
Stars around I — PostScript straight away, MAPS 18, 1997



The stars around notes, I & II, were written after Jacko's Metafont course in Holland, where he taught us among others the OK font. See also Adobe's Bluebook, programs 16–21. In Adobe's Redbook p101, there is an example of a user-defined font of two characters, a filled square and a filled triangle. Another example is given by David Byram–Wigfield who creates a special font `QuadFont` for crosswords. Don Lancaster advocates his Fonts for Free modifications, such as embossed variants.

GUST battleship

The GUST EuroTEX1994 logo — The Battleship — I rewrote at the time in PostScript. In order to obtain the intersection points of 2 straight lines a stack-oriented 2x2 linear equation solver was written in PostScript. In the specification of the points, `\p0` ... `\p9`, `intersect` has been invoked, which delivers the intersection point of 2 lines. The `mean` invoke delivers the midpoint of 2 points. The equation solver in PostScript and the `def`'s `intersect` and `mean` are included in `PSlib.eps`.[32] This is an example where the a priori projection of the drawing and working in 2D throughout is handy. No 3D data.



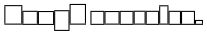

```
/p0{0 0}def
/p1{3 s mul 0}def
/p2{4.5 s mul 2 s mul}def
/p3{3 s mul s}def
/p4{-.75 s mul 2 s mul}def
/top{2.5 s mul 3 s mul}def
/p5{p0 top p3 p4 intersect}def
/p6{p0 p1 mean top p3 p4 intersect}def
/p7{top p1 p3 p4 intersect}def
/p8{p2 p5 top p1 intersect}def
/p9{p8 dup 0 exch top p0 intersect}def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

Paradigms: Loops, MAPS 96.2, 1996

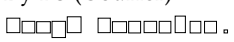
Outlines

Borrowed from the TEXbook p65, but rewritten with the use of the FIFO paradigm, and in PostScript, but ... alas, there is no `stringsize` operator in PS. PS' `stringwidth` delivers only the x-size of the string. The kludge of rotating a character and measuring the 'height' did not work. `Pathbb` was needed. A nice example of the use of outlines is GUST's logo.

By TEX



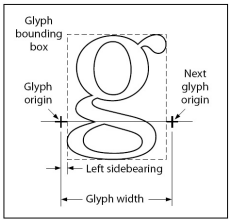
By PS (Courier)



```
\leavevmode\fifow Tough exercise. \wofif{ }%{ }=sentinel
%with
\def\fifo#1{\ifx\ofif#1\ofif\fi\process#1\fifo}%
\def\ofif#1\fifo{\fi}%
\def\fifow#1 {\ifx\wofif#1\wofif\fi\processw{#1}\fifow}%
\def\wofif#1\fifow{\fi}%
\def\processw#1{\fifo#1\ofif\ }%
\def\process#1{\boxit#1}%
\def\boxit#1{\setbox0=\hbox{#1}%
  \hbox{\lower\dp0\vbox{\hrule
   \hbox{\vrule\phantom#1\vrule}\hrule}}}
```

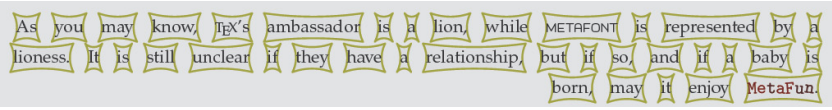
```
/Courier 40 selectfont /str ( ) def
(Tough Exercise.)
{str exch 0 exch put newpath 0 0 moveto
 str false charpath flattenpath pathbbox
 /ury exch def /urx exch def
 /lly exch def /llx exch def
 /w urx llx sub def /h ury lly sub def
 str ( ) ne {llx lly  w h rectstroke}if
 str stringwidth translate
}forall
```

32. Since then a 3x3 linear equation solver has been included in `PSlib.eps`, which (as the 2x2 solver) uses partial pivoting. These are to be preferred above the appealing Metafont/-Post symbolic equation solving functionality when the system is ill-conditioned. For those cases it is best to reformulate the problem into a better conditioned one; next best is to use pivoting strategies. In solving the radical circle problem in my Circle Inversions paper, a sub-problem was to determine the touching point of 2 circles, which is ill-posed, and therefore restated as finding the intersection point of a circle and a nearly-orthogonal line to it.



Paradigm: the use of the nested FIFO-technique,[33] that is, words are scanned and each word is scanned for its characters. A beautiful example of the use of `\phantom`.
PS paradigm: walking through a string. In PS a character's BoundingBox has to be determined. The `charbox` width is not the same as `stringwidth` of a character, see picture at right borrowed from the Redbook. The left-side bearing, a kerning(?), is included in the value of `stringwidth`. The PS-code looks simpler with `forall` scanning.

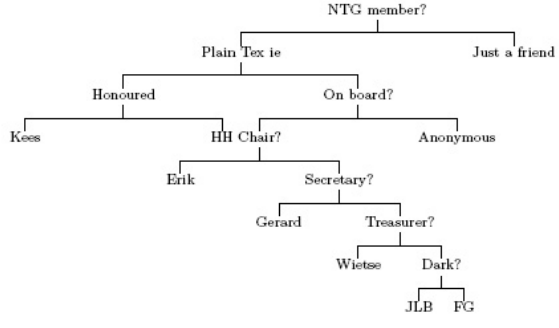
MetaFun's funny-boxed texts



Note the curved, or as Hans calls them squeezed, boxes, which can't be done nicely in TEX alone.

Paradigms: Searching, MAPS 96.2, 1996

After so many years, `BLUe.tex` amazed me by this Searching article. A variant solution of the TEXbook exercise 22.14 has been worked out. In 'Paradigms: searching,' I used a tree structure in TEX for searching. At the end of the article the tree of information was printed as shown below.
I collected my BLUe files: `blue.tex`, `fmt.dat`, `tools.dat`, `lit.dat`, and `pict.dat`, in a map and reproduced the tree by just doing what was stated in the article, and listed in the input verbatim below.[34] Et voilà.
I'm pleased by the results. BLUe surprised even me, by this unbalanced tree and the mean-and-lean data description, after so many years!




```
\input blue.tex
\thisbt{\xoffset{-400}}
\beginbt 1 NTG member?
10 Plain TeX ie?
100 Honoured?
1000 Kees
1001 HH
101 On board?
1010 Chair?
10100 Erik
10101 Secretary?
101010 Gerard
101011 Treasurer?
1010110 Wietse
1010111 Dark?
10101110 Johannes
10101111 Frans
1011 Anonymous
11 Just a friend
16 \endbt%16=checksum sort
of
\bye
```

Below from `bttool` the backtrack macro has been copied. The nodes are binary numbers, and when the `node<binary number>` is undefined the end of the branch has been reached, and backtracking is performed. `\whiteS` draws South and the leaves. Not recreational, pretty advanced macro writing. BLUe in full glory.

```
\beginbt \def\drawbt{\whiteS{120}%
   \ea\ifx\csname\node0\endcsname\relax
         \tbward\fi%Backtrack
   \S{80}\advance\k-125
   {\W{\the\k}\S{80}\edef\node{\node0}%
    \drawbt}%
    \E{\the\k}\S{80}\edef\node{\node1}%
    \drawbt\relax}
\def\tbward#1\relax{\fi}
```

Note the minimal, necessary data specifications: just the binary 'addresses' of the nodes next to their contents. TEX will handle all that is needed.
Contest: How to do this in PS or MP?[35]

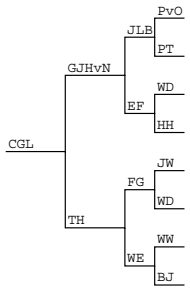
33. FIFO and LIFO sing the blues — Got it?, 1992, 1995(rev), MAPS 9(original). Bernd Raichle likes my `\fifo...\ofif` termination TEXnique.
34. Because it was with the gkp-macros I obtained not a picture cropped to the BoundingBox. In order to crop the picture I selected the picture in `Acrobat Pro` and copied it to the clipboard, and created a new cropped `.pdf`, at the expense of sharpness. I should redo it in PostScript, on occasion of EuroTEX-ConTEXt2012. Phil Taylor communicated his recent work on a real genealogy tree in TEX.

A balanced tree in TEX and PostScript

The production rule à la Lindenmayer for the balanced tree reads

$$\mathrm{Bt}_n = \mathrm{E}_n \oplus [\mathrm{N}_{n\div 2}\,\mathrm{Bt}_{n\div 2}] \oplus [\mathrm{S}_{n\div 2}\,\mathrm{Bt}_{n\div 2}], \quad \text{with} \quad n = ...256, 128, ...2,$$

Bt_n the Binary tree of order n,
E_n, N_n, S_n means draw East, North, South with step-size n
$\oplus$ means splice operator, i.e. concatenate properly,
[means store graphics state on the GS-stack and open a new one,
] means remove current graphics state off the GS-stack and recall previous.[36]




```
\def\bintree{\E{\the\n}%
   \ifnum\n=2 \eertnib\fi
   \divide\n2 {\N{\the\n}\bintree}%
              \S{\the\n}\bintree}%
\def\eertnib#1\bintree{\fi}%terminator
\let\Eold\E                %Text at leaves
\def\E#1{\global\advance\k1
  \xytxt{ \csname\the\k\endcsname}%leaf
  \Eold8}}
\def\1{CGL}\def\2{GJHvN}\def\3{JLB}
...
\def\xytxt#1{%place text #1 at x, y
 \xy{\vbox to0pt{\vss
   \hbox to0pt{\strut#1\hss}\kern0pt}}}}
\def\xy#1{%place #1 at x, y
   \vbox to 0pt{\kern-y
     \hbox to 0pt{\kern x#1\hss}\vss}}
```

```
/Bintree{%n on stack
   /n exch 2 div def
   E %draw East and add leaf
   n 8 gt {currentpoint N n Bintree
           moveto      S n Bintree}if
   /n n 2 mul def}def %end Bintree
/N{0 n     rlineto}def /W{n neg 0 rlineto}def
/S{0 n neg rlineto}def
/E{gsave  ntg k get 2 3 rmoveto show grestore
   /k k 1 add def n 16 le {25}{50} ifelse 0 rlineto
  }def
%
/Courier 12 selectfont /k 0 def
/ntg [(CGL) (GJHvN) (JLB) (PvO) (PT) (EF) (WD) (HH)
      (TH) (FG) (JW) (WD) (WE) (WW) (BJ)] def
0 -3 moveto 128 Bintree stroke
```

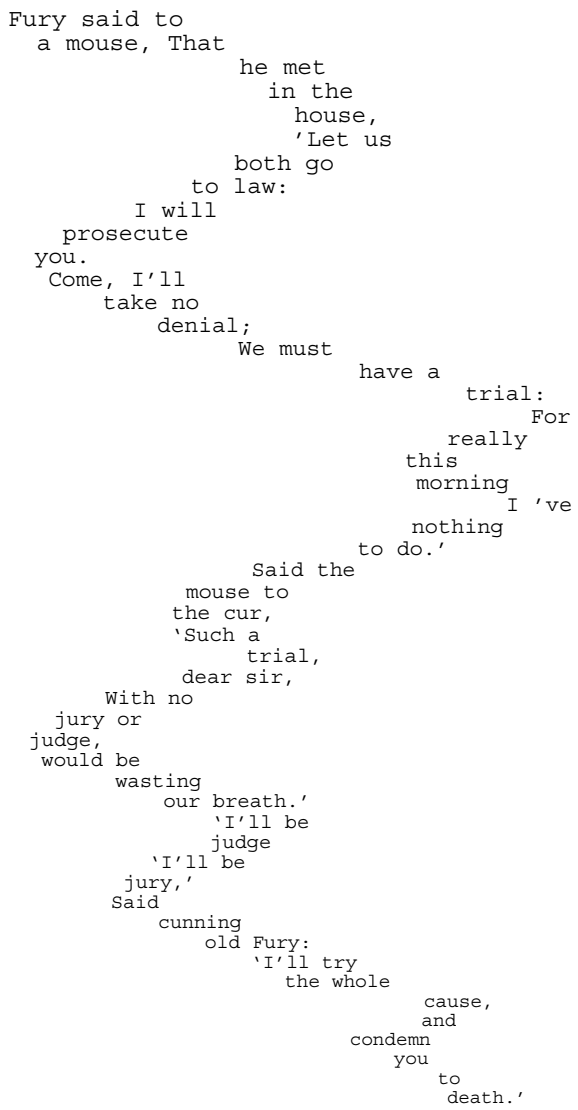
Intriguing is the use of `currentpoint` in PostScript, which saves the current position values on the stack for use in the other branch. In the TEX-version the placing of the picture on the page is cumbersome. PostScript is simpler for the purpose.
Paradigm: the wind `defs`: `N`, `E` and `W`, which resulted from the Turtle Graphics approach, are used within a recursive environment.

Alice's tale and the mouse's tail, GUST Programming Pearl 2010

This emblematic proza by Lewis Carroll has been first typeset in PostScript, by the use of `forall`, which expects an array, enclosed by [ ], and a procedure, enclosed by { }, on the stack. It is another example of printing text along a path, without an explicit PostScript path, neither is `pathtext` invoked. The array contains a necklace of strings, each enclosed by ( ), the WYSIWYG data. The procedure scales and typesets the lines. No explicit positioning by coordinates on the page nor controlling of the loop is needed. I started with PS' `pathforall`, worked on it for 15-30min, when the direct method popped up.
Paradigm: The `forall` walks through the array and delivers each element of the array on the stack.

In TEX, within a verbatim environment, the same can be achieved with mark-up in a WYSIWYG way; on the other hand one may dawdle with shifted `hbox`-es. Simplest is just to use `\obeyspaces\obeylines` and overrule TEX's default neglecting of superfluous spaces and `e-o-l`s. I was biased by TEX's automatisms and overlooked the simplest solution for quite a while.

35. E-mail solutions to `kisa1@xs4all.nl`.
36. The addition of the graphics state concept to the Lindenmayer production rules is an enrichment.




```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Alice's tale and the Mouse  tail, cgl feb 2010, 2012
%%BoundingBox: 0 -350 250 115
%%BeginSetup
%%EndSetup
/Courier 10 selectfont
/crlf { .995 dup scale
        currentpoint 10 sub exch pop  LM exch moveto } def
/LM 10 def LM 100 moveto
%array, proc and forall
[(Fury said to)
(  a mouse, That)
(                he met)
(                  in the )
(                    house,)
(                    'Let us)
(                both go)
(             to law:)
(         I will)
(    prosecute)
(  you.)
(   Come, I'll)
(       take no)
(           denial;)
(                 We must)
(                         have a)
(                                 trial:)
(                                      For)
(                                  really)
(                              this)
(                               morning)
(                                      I 've)
(                               nothing)
(                            to do.')
(                   Said the)
(              mouse to)
(             the cur,)
(             `Such a)
(                  trial,)
(               dear sir,)
(        With no)
(    jury or)
(  judge,)
(   would be)
(         wasting)
(             our breath.')
(                 `I'll be)
(                 judge)
(            `I'll be)
(          jury,')
(         Said)
(             cunning)
(                 old Fury:)
(                     `I'll try)
(                        the whole)
(                                   cause,)
(                                   and)
(                               condemn)
(                                   you)
(                                       to)
(                                        death.')
]{show crlf}forall
```

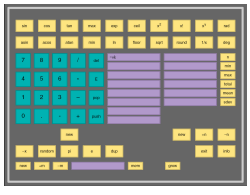
Tic-tac-toe interactivity

A tic-tac-toe application via the log-file, what was called at the time 'in dialogue with TEX,' is discussed in the Searching-article. More elaborated macros are supplied in the article. The advanced macros pay attention to for example a test for inconsistent input, or when a draw situation has arrived to stop automatically and start a new game. At the time when I wrote the macros the dialogue with TEX was via the log-file. At the moment TEXworks opens a different window for supplying the answers to TEX's questions; the questions are shown in the console window.

```
Tic-tac-toe
- - -
- - -
- - -     Supply index for +:
\index=1
+ - -
- - -
- - -     Supply index for o:
\index=5
+ - -
- o -
- - -     Supply index for +:
\index=3
+ - +
- o -
- - -     Supply index for o:
\index=2
+ o +
- o -
- - -     Supply index for +:
etcetera terminated by index 0.
```

```
\def\showboard{\immediate\write0{\1\2\3}
               \immediate\write0{\4\5\6}
               \immediate\write0{\7\8\9}}
\def\initialize{\def\1{-}\def\2{-}\def\3{-}
                \def\4{-}\def\5{-}\def\6{-}
                \def\7{-}\def\8{-}\def\9{-}}
\def\play{\initialize \loop\showboard
   \ifx\mark\markplayer
      \let\mark\markopponent\else
      \let\mark\markplayer\fi
   \immediate\write0{Supply index for \mark:}
   \read0to\index         \expandafter
   \xdef\csname\index\endcsname{\mark}
\ifnum\index>0 \repeat}%end \play
\def\markplayer{+}\def\markopponent{o}
\endlinechar-1 %TB20.18
\play \bye
```

Interactivity: Hans Hagen's calculator in ConTEXt + MetaPost + PDF + ...



Impressive and the summum of interactivity is Hans' calculator. It was said at the time that Knuth was strongly against holding up the processing of TEX, and in the meantime doing something else. Hans has exploited this use of TEX, in for example interfacing of ConTEXt with MetaPost. Fun or serious?

Paradigms: Just a little bit of PostScript, MAPS 19, 1996

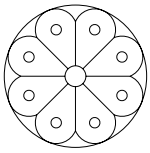
The article will be named JlPS, for short. Previewing was inconvenient via the Apple Laserwriter. `PSView` and `GhostScript` were not available on my PowerMac. All pictures in the article have been included in `PSlib.eps` on the occasion of EuroTEX-ConTEXt21012.

Yin Yang

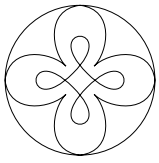
Everybody made his Yin Yang, I presume. Hobby provided it as an example in his MetaPost manual. Included is my matured coding, which is different from the coding in 'Tiling in Metafont and PostScript.'

```
%!PS-Adobe- Yin Yang. cgl July 2009
%%BoundingBox: -8 -8 70 70
/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
/circle{translate 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 circle fill grestore
gsave 0 mhR circle 1 setgray fill grestore
```

Barn and Malbork window



The left window has been done by the use of `arc` and the rotation of user space in PostScript. The right window is an exercise in using splines, the `curveto`, and choosing appropriate control points. The choice of control points I did by trial-and-error. Both are included in `PSlib.eps`.



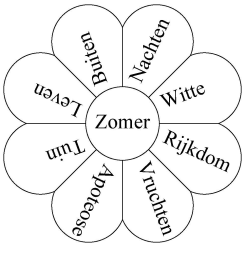
Stylistic flowers

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -26 -26 26 26
/r 18 def
10 {0 r r 270 360 arc
    r 0 r  90 180 arc
   36 rotate} bind repeat
stroke
```

The black-and-white line-drawing flower has been drawn in PostScript, see verbatim above, where use has been made of the variable user space, such that the drawing of each leaf begins and ends in (0, 0). Subtle are the choices of the circle centres and their sequence. The coding is one of my favourites to demonstrate the use of the variable user space functionality in PostScript. The gradient colouring has been done interactively in Photoshop by my wife Svetlana Morozova on occasion of the EuroTeX-ConTEXt2009. The rotation of the user space can be understood by just paying attention to the rotated coordinate axes. All that follows is drawn with respect to the rotated coordinate axes.[37] At right a circular Julia fractal 'stylistic flower.'

For the bulletin of our gardeners club

The barn window has been reused, enriched by rotated text in PS. Paradigm: rotated texts stored as array.




```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Zomer, vormgedicht, Barnwindow basis. CGL, April 2007
%%BoundingBox: -180 -180 180 180
/l 140 def /Times-Roman 30 selectfont
/m l 22.5 cos mul def /r l 22.5 sin mul def
8{r 0 moveto 22.5 rotate m 0 r -90 90 arc 22.5 rotate}repeat
%inner circle
r 0 moveto 0 0 r 0 360 arc stroke -40 -7 moveto (Zomer) show
/texts[(Witte) (Nachten) (Buiten) (Leven)
          (Tuin)  (Apoteose) (Vruchten) (Rijkdom)]def
0 0 moveto 20 rotate /r r 10 add def
0 1 7{r 0 moveto texts exch get show 45 rotate}for
```

Tiling in PostScript and Metafont — Escher's wink, MAPS 19, 1997

The article will in the sequel be referred to by TPS-MF, for short. All pictures in the article have been included in `PSlib.eps` on the occasion of EuroTEX-ConTEXt2012.

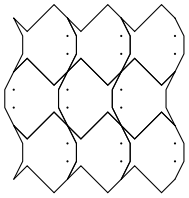
Escher's Sun and Moon

← central part
Dark birds in daylight or white birds at night?
The picture was sampled: the sampled points were provided as spline data.

Zon en maan →

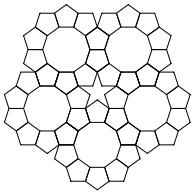
37. See the Bluebook Ch 6 More Graphics, p49, for an enlightening, simple picture.

Escher's fishes and Buddha's

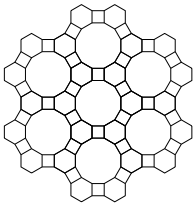


These tiles are examples of Escher's technique where the drawing extends over the boundary of the square tile, such that it matches with the adjacent tiles.
The right picture consists of 4 groups of 4 tiles, where the later are composed of rotated copies.

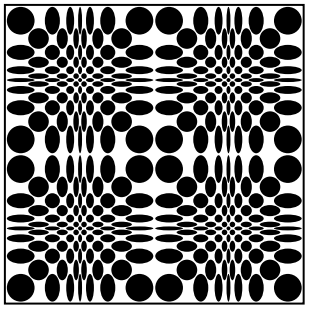
Tilings



The left figure is done by a garland of pentagons. The garland is copied 4 times. The enclosed star is spurious.
The right figure is classified by $\{4, 6, 12\}$, a nice layout for a herb garden.



Schrofer's Op Art

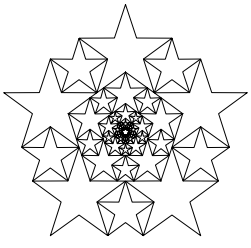


```
/Schrofer{0 begin /flipflop true def /s 5 def
/drawgc{gsave
   r c translate r abs 5 div s add c abs 5 div s add scale
   0 0 1 0 360 arc
   fill grestore}def%end drawgc
/indices [30 21 14 9 5 2 0 -2 -5 -9 -14 -21 -30] def
indices{/r exch s mul def
  gsave indices{/c exch s mul def flipflop{drawgc}if
                /flipflop flipflop not def}forall
  grestore}forall
end}def%end Schrofer
%4 tiles and border
gsave 2{gsave 2{schrofer 375 0 translate}repeat
                grestore   0 375 translate
       }repeat grestore
5 setlinewidth -190 dup 755 dup rectstroke%border
```

A nice picture is Schrofer's Op Art, of which I included a tile of four.[38]Crucial are the row and column indices. The circles and ellipses are scaled copies of the unit circle. All the 80+pictures from the 'Paradigm: Tiling in Metafont and PostScript'-article have been included in `PSlib.eps` on occasion of EuroTEXConTEXt2012.

Tiling by stars

In `PSlib.eps` this stars composition has name `tilxia`; the code is ≈ 40 lines long.



```
/star % n r on stack, n>=5
{/rstar exch def /nstar exch def
 /alfahstar 180 nstar div def
 /rinstar rstar 2 div alfahstar cos mul 1.5 sub def
 0 rstar moveto
 nstar{alfahstar rotate
     0 rinstar lineto
     alfahstar rotate
     0 rstar lineto
  }repeat stroke
}def
```

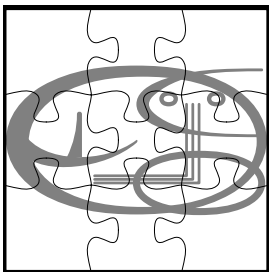
Star impression by Svetlana Morozova →

38. Willem Schrofer, 1898–1968, was a Dutch artist and teacher. In the 30s he painted abstract later figurative.
`http://nl.wikipedia.org/wiki/Willem_Schrofer`.

Puzzle with cat

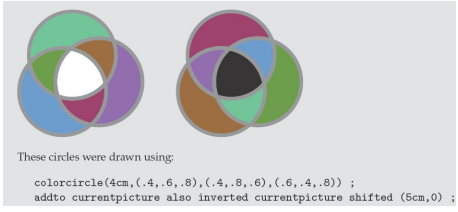
```
%MetaPost variant of \quote{cat} which was adapted from Metafont
beginfig(1); tracingstats:=proofing:=1;
path p[]; sz=25; hsize=17.5sz; vsize=10sz;
%moustache
pickup pencircle scaled .1pt;
draw (.75hsize, .75vsize)--(.75hsize, .2vsize)--
  (.333hsize, .2vsize);
draw (.725hsize, .75vsize)--(.725hsize, .225vsize)--
  (.333hsize, .225vsize);
draw (.7hsize, .75vsize)--(.7hsize, .25vsize)--
  (.333hsize, .25vsize);
z1=(  hsize,.5vsize);  %right
z2=(.5hsize,  vsize);  %top
z3=(      0,.5vsize);  %left
z4=(.5hsize,      0);  %bottom
penpos1(.05vsize,0);penpos2(.09vsize,90);penpos3(.175vsize,180);
penpos4(.075vsize,270);
%Nonlinear interpolation for extra point z25
z25=(z2{left}..{down}z3)intersectionpoint
    ((.2hsize,0)--(.2hsize,vsize));
penpos25(.15vsize,135);
penstroke z1e{up}..z2e{left}..z25e..z3e{down}..
       z4e{right}..{up}z1e;
%mouth
pickup pencircle scaled .2pt;
draw superellipse((hsize, .2vsize),(.75hsize, .4vsize),
                  (.5hsize,.2vsize),(.75hsize,0),.725);
%ear
z5=(0,.5vsize);      penpos5(1.75pt,-90);
z6=(.5hsize,.5vsize);penpos6(.8pt,0 );
p1=z5..controls (.125hsize, .333vsize) and
   (.375hsize,.333vsize)..z6;
z7=point.5 of p1;    penpos7(1.2pt,-30);
z9=point.5 of p1;%(.25hsize,.4vsize);
x9:=x9-.175pt;         penpos9(.75pt,180);
z8=(.25hsize,.75vsize);penpos8(.3pt, 180);
penstroke z6e..z7e..z5e;
penstroke z8e{down}..z9e;
%brow
z10=(hsize,vsize);penpos10(.2pt,90);
z11=(.575hsize,.9vsize);penpos[11](.5pt,135);
z12=(.5hsize, .75vsize);penpos12(.8pt,180);
z13=(.575hsize,.6vsize);penpos13(.4pt,-135);
z14=(hsize,.5vsize);    penpos14(.15pt,-90);
penstroke z10e{left}..z11e..{down}z12e..
          z13e..{right}z14e;
%eyes
p2= superellipse((sz, .375sz),
 (.5sz, .75sz),(0,.375sz),(.5sz,0),.725);
pickup pencircle scaled .1pt;
draw p2 shifted (.6hsize, .75vsize);
draw p2 shifted (.79hsize, .75vsize);
endfig;
end
```

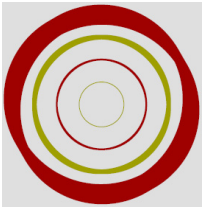
```
%!PS Puzzle of cat, cgl June 97
%%BoundingBox: -4 -39 179 144
%%Creator: MetaPost
%%CreationDate: 1996.05.03:1858
.5 setgray
newpath 178.50075 50 moveto
178.50 66.10 167.61 79.62 153.48 87.70 curveto
133.59 99.07 110.39 102.00 87.5 102.00 curveto
64.61 102.00  41.41  99.07 21.52 87.70 curveto
7.39 79.62 -3.50 66.10 -3.50 50 curveto
-3.50 33.90 7.39 20.38 21.52 12.30 curveto
41.41 0.93 64.61 -2.00 87.5 -2.00 curveto
110.39 -2.00 133.59 0.93 153.48 12.30 curveto
167.61 20.38 178.51 33.90 178.51 50 curveto closepath fill
%
%et cetera, next the puzzle overlay
%
87.5 50 translate 0 setgray .5 setlinewidth
/s 31.5 def /t .6 s mul def
/el{s 0 moveto currentpoint .8 s mul 0
    .4 s mul -.2 s mul  .2 s mul 0 curveto
   currentpoint currentpoint exch
    t t  0 t curveto
}def
%
/side{el reversepath -1 1 scale el -1 1 scale}def
%
/piece{4{0 s translate side 0 s neg translate
  90 rotate}repeat}def
%
/elinv{1 -1 scale el 1 -1 scale}def
/sideinv{elinv reversepath -1 1 scale elinv -1 1 scale}def
/ipiece{4{0 s translate sideinv 0 s neg translate
  90 rotate}repeat}def
/border{3 s mul dup moveto
        4{-3 s mul 3 s mul lineto 90 rotate}repeat
        closepath}def
border clip
ipiece stroke
-2 s mul 4 s mul 2 s mul{/i exch def
-2 s mul 4 s mul 2 s mul{/j exch def
 gsave i j translate piece stroke grestore
}for}for
7.5 setlinewidth border stroke
%%EOF
```

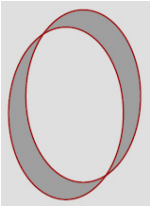


The background picture I made ≈50 years ago by hand. When I started with Metafont in 1995 it was my first graphics example. The PostScript code resulted from the MetaPost adaptation, i.e. deleting Metafont peculiarities. Both included codes are too lengthy to my taste. The PS variant shows how, after we have distilled the PS data from MP, the picture can be further enriched in PS.

MetaFun simple contrasts



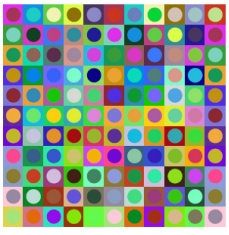


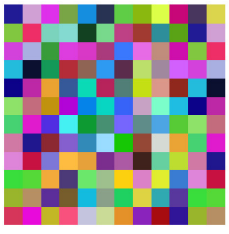


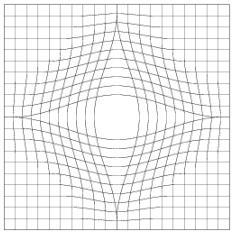
The two pictures at right make use of a varied pen in MetaPost. Calligraphic effects can be obtained.

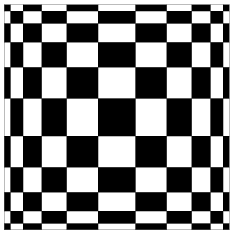
Vasarely

In 1995 I created 8 Vasarely impressions in Metafont; today I could still visualize them in BlueSky's Metafont on my PowerMac of the mid-90s.









The black-and-white pictures were visualized by, and downloaded as `.png` from, Troy Henderson's `mppreviewer`.[39] The Metafont line-picture makes use of the `interpath` functionality, Metafont book p134, which functionality is not available in PostScript, see code below, nor is there a path data-structure, alas.

```
sz=100; path p,q;
p= (-sz,0){right}...(-.9sz,0)...(0,.2sz)...(.9sz,0)...{right}(sz,0);
q= (-sz,sz)--(-.25sz,sz)--(0,sz)--(.25sz,sz)--(sz,sz);
for k= 0 upto 10: pickup pencircle scaled (.02(k+1)*pt) draw interpath(k/10, p, q); endfor
addto currentpicture also currentpicture rotated 180;
addto currentpicture also currentpicture rotated 90;
pickup pencircle scaled .1pt; draw unitsquare scaled 2sz shifted (-sz,-sz);
```

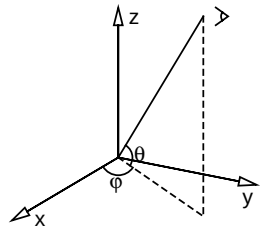
The 3 Vasarely[40] impressions left use PostScript's `rnd`, the pseudo-random number generator. The PS-code for the second picture reads

39. Troy has improved his previewer since 2009, several packages can be used now. `http://www.tlhiv.org/mppreview`. He has also provided a LaTeX previewer and a function-grapher previewer. See his TUGboat 2012 article. All have been done because installing volunteer software by a casual user has become too cumbersome.

40. Victor Vasarely born Vásárhelyi Gyözö, 1906 Pécs – 1997 Paris, was a Hungarian French artist whose work is generally seen as aligned with Op Art. His work entitled Zebra, created by Vasarely in the 1930s, is considered by some to be one of the earliest examples of Op Art. Vasarely developed his style of geometric abstract art, working in various materials but using a minimal number of forms and colours. `http://www.en.wikipedia.org/wiki/Victor_Vasarely`.

```
/r 20 def   /R r 10 mul def
/rd2 r 2 div def /-rd2 rd2 neg def /ri rd2 2 sqrt div def %ri = radius inner circle
/s 1073741823 def %/s 2 31 exp 1 sub def
/square{nrand nrand nrand setrgbcolor -rd2 dup r r rectfill
        nrand nrand nrand setrgbcolor 0 0 ri 0 360 arc fill}def
/nrand{rand s div}def 1951 srand%seed
R neg r R{/i exch def
R neg r R{/j exch def gsave i j  translate square grestore}for}for
```

Projection for emulation of space objects

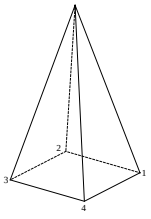


right-screw coordinate system with view direction φ, θ in the main octant

```
/ptp{% point x y z ==> x' y'
% use: /pair { x y z ptp } def
%parameters: phi, theta viewing angles
%coordinate system xyz: x to yuo y right z up, right turning
ptpdict begin /z exch def/y exch def/x exch def
x phi sin mul neg                y phi cos mul add
x phi cos mul theta sin mul neg y phi sin mul theta sin mul sub
                                 z theta cos mul add
end}bind def
/ptpdict 3 dict def
```

The idea in the projection used is that an object is viewed at in plane (computer screen) orthogonal to the view direction. In programming this is translated such that the spacial coordinates are projected onto the projection plane by `ptp`, mnemonics for point-to-pair.[41]

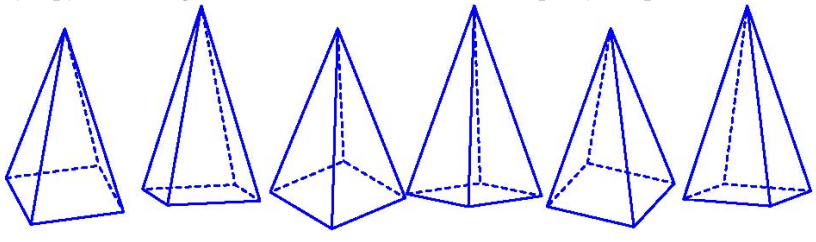
The pyramid illustrative example is the pyramid. Data of the pyramid and the pyramid code have been borrowed from `PSlib.eps`.




```
/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
   []0 setdash stroke
top moveto z1 lineto%  2 -3  rmoveto (1)show
top moveto z3 lineto% -7 -3  rmoveto (3)show
top moveto z4 lineto% -3 -10 rmoveto (4)show
stroke
top moveto z2 lineto
[2]1 setdash stroke% -10 0 rmoveto (2)show
}def  %end pyramid
```

Pyramid viewed from various viewpoints. Do compare the code of pyramid with Hobby's pyramid as given in the MetaPost manual. Equally simple, isn't it?



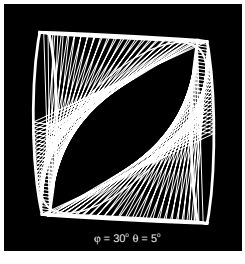
Escher's impossible cubes, TEX Education, EuroTEX-ConTEXt2009
For each corner of the cube there are 8 data-points, specified in 3D. In projection the points of intersection have been calculated as function of the viewing angle. Pretty detailed and tedious code. Too lengthy to be included here. The Metafont book contains a poor man's version, p113, exercise 13.7. The impossible cube in PostScript was written on occasion of the EuroTEX-ConTEXt2009.

41. For more detail see Appendix 0 of Gabo's Torsion, MAPS 42, 2011.

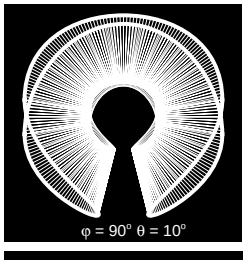
The paper is also available at `http://www.ntg.nl/maps/39/05.pdf`, next to the complete proceedings.

Paradigm: Graphics and TEX — a reappraisal of Metafont, MAPS 16, 1996

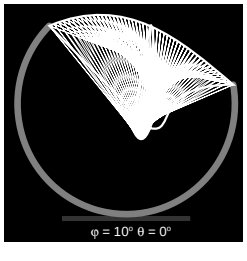
In 1996 I emulated Linear Construction No 2 in Metafont. The picture created by the original Metafont version of the emulation of Linear Construction No 2 is full-page included in LaTeX's Graphics Companion. In the Gabo's Torsion paper an improved and more accurate version in PostScript has appeared, next to some more emulations of Gabo's works. At right an animated simple versions of Linear Construction No 2. In 2011 I rewrote the emulations in PostScript. More use of projection techniques is in Gabo's emulations. The last PostScript version of the Linear Construction No 2 is the most complete and the best. The reverse video suggests the perspex material. Mathematically, the constructions are regular surfaces, meaning the surfaces are suggested by straight lines. It is said that Suspended was Gabo's favourite, because he showed the object at each-and-every exhibition.



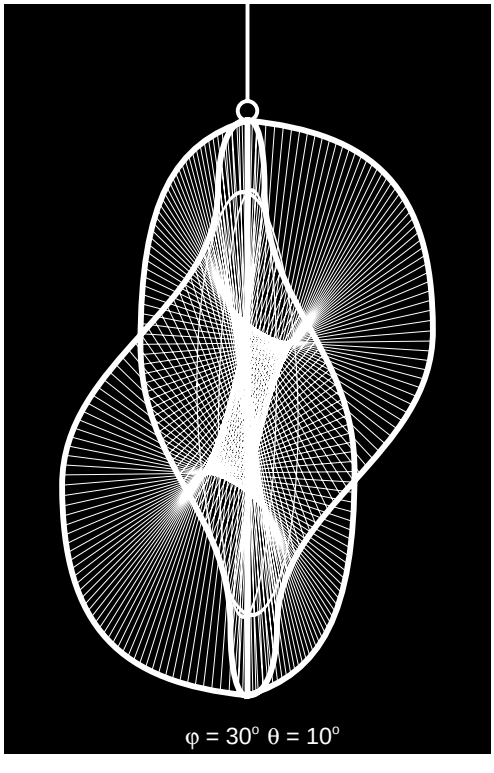
←Linear Construction No 1



←Spheric Theme
Linear Construction No 2→



←Suspended



Invokes of Gabo's emulations from PSlib.eps.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Linear Constr. in Space No 1
%%Author: Kees van der Laan, cgl 2011
%%BoundingBox: -130 -135 130 135
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%EndProlog
linearconstructionno1 showpage
```

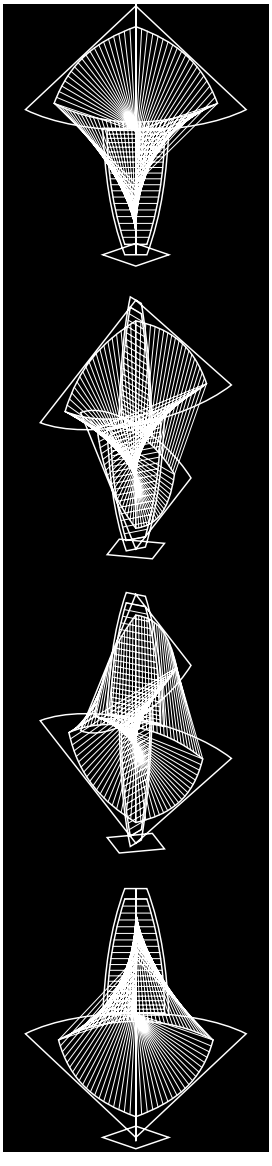
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Linear Constr. in Space No 2
%%Author: Kees van der Laan, cgl 2011
%%BoundingBox: -125 -30 125 355
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%EndProlog
linearconstructionno2 showpage
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Spheric Theme
%%Author: Kees van der Laan, cgl 2011
%%BoundingBox: -90 -95 90 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%EndProlog
sphericthemе showpage
```

Warning GhostScript can't be used for previewing with library use because GhostScript does not support the run command for file-inclusion, apparently. Do use PSView, Acrobat Pro or ...

Gabo's Torsion, MAPS 42, 2011

For the Metafont/Post aficionados my Torsion Metafont code of old is included, complete with projection and interactivity, which was not included in MAPS 42.
My emulations of Gabo's[42] objects on paper started in Metafont in 1996,[43] which marks the beginning of my using projection techniques. In the paper a few of Gabo's 3D constructions have been emulated in projection and can be viewed from various viewing angles. Torsion pictures in reverse video have been supplied below.



```
%December 1995, cgl. Gabo's torsion
tracingstats:=proofing:=1;screenstrokes;%tracingmacros:=tracingtitles:=1;
"Torsion from different viewpoints";
string yorn; message "Gabo's Torsion.";
             message "Do you wish simplest variant? (y or n):";
yorn:= readstring; size=50;
def openit = openwindow currentwindow from origin
    to (screen_rows,screen_cols) at (-size,300)enddef;
pickup pencircle scaled .005pt;
pair aux[]; path  po[], pi[];
if yorn="n": d=.1size else: d=0 fi;
r=.2size;
for b= 20:%15step10until45:
for a= 0 step30until90:
currentpicture:=currentpicture shifted (2size,0);
%The following def must be included or a, b, must be supplied as arguments.
def ptp(expr x,y,z)=(-x*cosd a +y*sind a,
                    -x*sind a *sind b -y*cosd a *sind b+ z*cosd b)enddef;
po1:=ptp(0,-size,2d)--ptp(0,0,size+2d)--ptp(0,size,2d)&
     ptp(0, size,2d)..ptp(0,0,0)..ptp(0,-size,2d)..cycle;
aux0:=.5[ptp(0,-size,2d),ptp(0,0,size+2d)];
aux1:=.5[ptp(0,size,2d),ptp(0,0,size+2d)];
pi1:=ptp(0,-size+2.5d,2.5d)..controls aux0..
     ptp(0,0,size-d)..controls aux1..
     ptp(0,size-2.5d,2.5d)...
     ptp(0, size-2.5d,2.5d)...ptp(0,0,d)...ptp(0,-size+2.5d,2.5d)..cycle;
po2:=ptp(-size,0,-2d)--ptp(0,0,-size-2d)--ptp(size,0,-2d)&
     ptp(size,0,-2d)..ptp(0,0,0)..ptp(-size,0,-2d)..cycle;
aux3:=.5[ptp(-size,0,-2d),ptp(0,0,-size-2d)];
aux4:=.5[ptp(size,0,-2d),ptp(0,0,-size-2d)];
pi2:=ptp(-size+2.5d,0,-2.5d)..controls aux3..
    ptp(0,0,-size+d)..controls aux4..
    ptp(size-2.5d,0,-2.5d)...
```

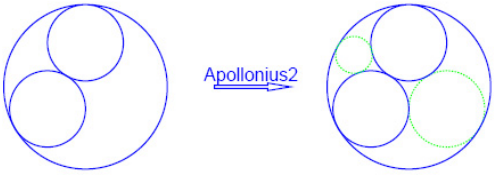
42. Naum Gabo, 1890–1977. Born Naum Borisovich Pevsner. Bryansk. Russian Constructivist. An excellent book about him and his works: Naum Gabo 60 years of Constructivism. Prestel-Verlag 1985, which appeared on the occasion of the retrospective exhibition with the same name at the Dallas Museum of Art, the Art Gallery of Ontario, the Guggenheim Museum NY, the Akademie der Künste Berlin, the Kunstsammlung Nordrhein-Westfalen, the Tate Gallery London. Wikipedia contains a short biography.
43. Graphics and TEX — a reappraisal of Metafont, 1996, MAPS 16.

```
    ptp(size-2.5d,0,-2.5d)...ptp(0,0,-d)...
    ptp(-size+2.5d,0,-2.5d)..cycle;
%Foot
po3:=ptp(r, 0,-size-2d)..ptp( .706r, -.706r,-size-2d)..
     ptp(0,-r,-size-2d)..ptp(-.706r, -.706r,-size-2d)..
     ptp(-r,0,-size-2d)..ptp(-.706r,  .706r,-size-2d)..
     ptp( 0,r,-size-2d)..ptp( .706r,  .706r,-size-2d)..cycle;
%
if yorn="n":fill po1; unfill pi1;fill po2; unfill pi2
           ;draw ptp(0,0,-size+d)--ptp(0,0,-size-2d)
           ;draw ptp(0,0,size-d)--ptp(0,0,size+2d)
      else:draw po1; draw po2; draw pi1; draw pi2 fi;
fill po3;
for k=0 upto 20: draw point .1k of pi1--point 5-.1k of  pi2; endfor
for k=0 upto 20: draw point 3+.1k of pi1--point 2-.1k of  pi2;endfor
showit; endfor endfor end
```

Circle Inversions, MAPS 40, 2010 This paper also introduces `PSlib.eps`.

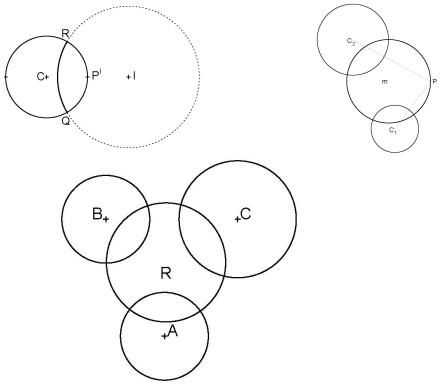
Apollonius problem

The jewel of the Circle Inversions paper is the solution of Apollonius problem: circles touching three circles. Apollonius problem is a classic, which solution I have overlooked for quite a while. The use of the `Apollonius2` PostScript `def` is no more difficult, or easier, depending on your expertise, than using high-level packages.



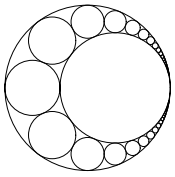

```
0   0  R 0 360 arc stroke% draw the 3 given circles
x1 y1 r1 0 360 arc stroke
x2 y2 r2 0 360 arc stroke
x1 y1 r                  % 3 circle specs for Apollonius2
x2 y2 r                  % pos r means touch external to x2 y2 r circle
0   0 R neg              % neg R means touch inside R-circle
Apollonius2              %delivers touching circles
 /rsnd1 exch def /ysnd1 exch def /xsnd1 exch def%collect data from stack
 /rsnd2 exch def /ysnd2 exch def /xsnd2 exch def
green                    %or a setdash when in b&w
xsnd1 ysnd1 rsnd1 0 360 arc stroke%draw found circles
xsnd2 ysnd2 rsnd2 0 360 arc stroke
```

The radical circle is the circle orthogonal to three circles. The library `def radical` from `PSlib.eps` avoids an ill-posed sub-problem. Below are included pictures of: two circles which intersect orthogonally, a circle through a point p which intersects two circles orthogonally, and the radical circle. How to invoke `radical` from `PSlib.eps` is shown in the PostScript snippet.



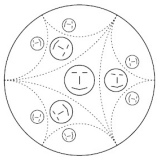

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Radical circle. CGL april2010
%%BoundingBox: -90 -90 114 105 %-r -r 3r 7r
(C:\\PSlib\\PSlib.eps) run %PS library
H14pt setfont
/r 50 def /mr r neg def
/Ax 0 def  /Ay mr  def          /A {Ax Ay} def  /Ar .75 r mul def
/Bx mr def /By r  def           /B {Bx By} def  /Br .75 r mul def
/Cx 1.25 r mul def  /Cy r def   /C {Cx Cy} def  /Cr     r def
A plus B plus C plus
newpath A Ar 0 360 arc stroke A moveto   2 0 rmoveto (A) show
newpath B Br 0 360 arc stroke B moveto -12 0 rmoveto (B) show
newpath C Cr 0 360 arc stroke C moveto   2 0 rmoveto (C) show
Ax Ay Ar
Bx By Br
Cx Cy Cr radical /radr exch def /rady exch def /radx exch def
newpath radx rady radr 0 360 arc stroke
radx 5 sub rady 14 sub moveto (R) show
```

Circle covered by circles

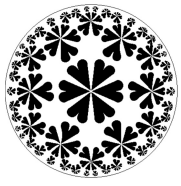


The covering of a circle by small circles I did in 1997 by using a non-linear equation solver in PostScript: `solveit`. Since I rediscovered the solution of Apollonius problem it can be programmed simpler, by the use op the `def Apollonius2`. At right a nice emulated collier of the inversion of the Mandelbrot fractal, borrowed from Lauwerier(1990): Een wereld van Fractals.

Inverted smileys and hearts



Tedious programming for inverted smiley-s. I did only 2 levels of circle inversions.
Inverted hearts was a side-effect when in search for Escher's Circle limits. The PS code was prompted by the Apple Laser Writer.

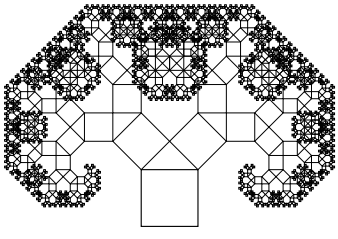


Circle inversion of a rectangular grid

The inversion has been simplified: circular curves have been straightened in the inversion.
At right the realization as a skylight window.

Pythagoras Trees, submitted MAPS, BachoTEX2012 proceedings

The Tree is a collection of scaled and rotated squares placed such that each parent square and its descendants enclose a rectangular triangle. The program is my favourite, non-trivial example of translating and rotating user space in PS. All one has to program is drawing a square and place it scaled and rotated at the right place, repetitively. This can be programmed in PostScript elegantly due to the translation and rotation of User Space functionality. Backtracking and the bookkeeping of auxiliaries is implicit.
The paper contains variants of the Pythagoras tree, such as an oblique tree and the 'X-mas' tree. More realistic trees are mentioned. The Pythagoras Tree has appeared as GUST Programming Pearl in 2011.



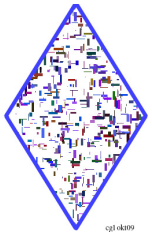
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Pythagoras Tree of squares
%%BoundingBox: -125 -20 175 200
%%BeginSetup              %crops to the prescribed BB
%%EndSetup                %when processed by Acrobat Pro
(C:\\PSlib\\PSlib.eps)run
/s 50 def                 %size of the side of the square
11 pythagorastree  pop    %order 11
%%EOF
```

à la Mondrian, MAPS 41, 2010

The compositions of coloured line pieces biased by a shade of colour, see accompanying figure,[44] are optionally enclosed in, and clipped by, a square, diamond or oval, to be specified by the user. The program is called `Mondrian` in `PSlib.eps`. In the article a

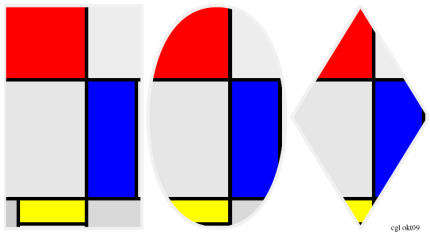
MetaPost variant was developed for comparison. The PostScript code lends itself for library use. Moreover, the use of a PostScript library `def` is more direct, a one-pass job, then the use of MetaPost, because MetaPost is a preprocessor of PostScript.[45]

←

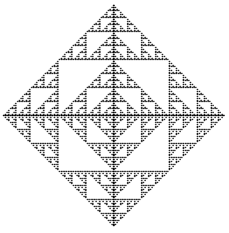


```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 620 790
(C:\\PSlib\\PSlib.eps) run
22121943 1  0  0  2 Mondrian showpage
07101951 .8 .2 .7 1 Mondrian showpage
%%EOF
```

From De Style period →



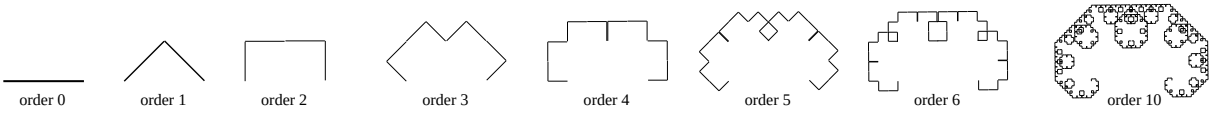
Game of Life Just to show that this game can yield fractal structures.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth Cell model a la Pickover, simplified
%%Author:  Kees van der Laan, March 2012
%%BoundingBox: -195 -195 195 195
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
64 pickover showpage
```

Classical Math fractals in PS, submitted MAPS, BachoTEX2012 proceedings

Lévy fractal An approximation of the Lévy fractal is also called a C (broken) line of a certain order. The constructive definition of various orders of C lines starts with a straight line, let us call this line C_0. An isosceles triangle with angles 45°, 90° and 45° is built on this line as hypotenuse. The original line is then replaced by the other two sides of this triangle to obtain C_1. Next, the two new lines each form the base for another right-angled isosceles triangle, and are replaced by the other two sides of their respective triangle, to obtain C_2. After two steps, the broken line has taken the appearance of three sides of a rectangle of twice the length of the original line. At each subsequent stage, each segment in the C figure is replaced by the other two sides of a right-angled isosceles triangle built on it. Such a rewriting relates to a Lindenmayer system. Paradigm: Lindenmayer production rule.



Julia fractals in PostScript, EuroTEX-ConTEXt2012

There are many Julia fractals. The one included below is my favourite. The left (incomplete) Julia fractal is obtained by inverse iteration and Monte Carlo, the right by the boundary scan method and enriched by colours by my wife Svetlana Morozova on occasion of EuroTEX-ConTEXt2012.[46] Interesting is the relationship of the various

44. Piet Mondriaan, 1872-1944, was a Dutch painter. He was an important contributor to the De Stijl art movement and group, which was founded by Theo van Doesburg. He evolved a non-representational form which he termed Neo-Plasticism. This consisted of white ground, upon which was painted a grid of vertical and horizontal black lines and the three primary colours.
`http://en.wikipedia.org/wiki/Piet_Mondrian`.

45. MetaPost does **not** interface. For example symbolic names declared in MetaPost can't be accessed in the resulting PostScript.

Julia fractals and the Mandelbrot fractal, which is the map to, and the bifurcation diagram of, the various Julia quadratic fractals.

```
%!PS-Adobe-3.0 EPSF-3.0  $$
%%BoundingBox: -165 -85 165 85
(C:\\PSlib\\PSlib.eps) run
-.8 .15 5000 JULIAMC         %<--Cloud inverse iteration
showpage
-.59 0.34 2.1 1.85 80 JULIABS%Cloud boundary scan -->
showpage
```

The late Mandelbrot[47]advocated that Fractal Geometry is better suited to model clouds and similar repetitive, natural forms than Euclidean geometry.

Conclusions

Portability in time of TEX scripts is hampered when several tools next to TEX have been used. TEX scripts which included PostScript graphics in the past, have to be adapted for use with pdfTEX.

Not only are TEX's CM bitmapped fonts too rigid with respect to font modifications and scaling, but also the philosophy of unbreakable boxes is too rigid in view of page-breaks. TEX's macro language is complicated, verbose and error-prone.

User mark-up can be reduced by letting TEX insert mark-up. Illustrations can be obtained by programming in PostScript, supported by Lindenmayer-like production rules and by projection techniques for emulation of 3D objects. Photoshop can be used as post-processor.

Printing along implicit paths can be done without the use of Adobe's pathtext and alikes.

On occasion of EuroTEX-ConTEXt2012 PSlib.eps has been extended by more than 175 pictures, from JlPS, TPS-MF, and from my fractal geometry work, i.e. translations of Lauwerier's BASIC codes into PostScript, and contains of sept 2012 ≈ 300 defs next to constants and colour names. The defs of Adobe's Bluebook are included and are also available in a separate file. The test programs of the Bluebook are also available in a separate file.

BLUe.tex, fmt.dat, tools.dat, pic.dat for TEX-alone pictures (and the relatively new cousin PSlib.eps, the library for PostScript pictures)[48] next to lit.dat, can be of use for Ben Lee User of the TEXbook fame, even after 17 years. They survived several computer migrations. Pictures of pic.dat can be reused and adapted. I undusted the unbalanced binary tree jewel.

Metafun I (mis)used for viewing old Metafont graphics. Bluesky's Metafont on my old PowerMac is no longer needed.

Not only is the use of TEX&Co recreational, the attendance of (Euro)TEX-meetings is highly recreational and instructive, especially the Polish GUST BachoTEX's with their bonfire and guitars at night. They were like holidays for me.

The mark-up of this paper does not adhere to the promise of the ideal marked-up texts; a lot of adjusting had to be done in order to obey the limitations of the page-size due to the vbox-s with unbreakable verbatim and picture elements next to each other. Letting float these elements would have yielded a mess.

Do realize that typesetting Bridge, Chess, or ... positions is several orders of magnitude less complicated than developing Bridge, c.q. Chess, playing programs.

46. Lauwerier, H.A(1987): FRACTALS — meetkundige figuren in eindeloze herhaling. Aramith. (Contains programs in BASIC. Lauwerier, H.A(1991): Fractals: Endlessly Repeated Geometrical Figures, Translated by Sophia Gill-Hoffstadt, Princeton University Press, Princeton NJ1. ISBN 0-691-08551-X, cloth. ISBN 0-691-02445-6 paperback. 'This book has been written for a wide audience ... ' Includes sample BASIC programs in an appendix. Audience: Instructors, (high-school) students, and the educated layman.)
47. Mandelbrot(1982): The Fractal Geometry of Nature. W.H. Freeman and Co.
48. Introduced in Appendix 1 of Circle Inversions, MAPS40, 2010. http://www.ntg.nl/maps/40/03.pdf.

Developers do the typesetting as an aside. Keep the right balance between form and contents.

'A professional starts where an amateur ends', to quote G.E.Forsythe, my greatest hero. Room for professionals.

> *"It's a myth to believe that each-and-every (La)TEX, ConTEXt, LuaTEX, or ...-user can produce printing-house typographic quality."*

It would be better if users are more modest and strive after preprint results. A preprint is correct with respect to contents and language. To achieve typographic printing-house quality requires another level of non-TEXnical expertise. Typographical corrections should be strictly local and have no global effects, avoiding introducing new typographical errors.

IDE My PC runs 32 bits Vista, with Intel Quad CPU Q8300 2.5GHz assisted by 8GB RAM. I visualize PostScript with `PSView` and convert into `.pdf` via `Acrobat Pro 7`.[49] My cripple PostScript editor is just Windows 'kladblok (notepad), and sometimes I misuse TEXworks for the purpose.' I use Adobe's `EPSF`-feature to crop pictures to their `BoundingBox`. The cropping is necessary for inclusion in documents.

Pictures made by the `gkp`-macros are still viewed in my `BLUe.tex` system of 1995. Metafont pictures are viewed in BlueSky's Metafont which runs on my PowerMac of 1996. No `.eps` or so as result. MetaPost pictures I drop on Henderson's `mppreviewer` and get `.png` in return. Old Metafont I can view in Hans Hagen's MP-interfacing program as well, next to via my BlueSky Metafont on my old PowerMac.

For document production I use TEXworks IDE with the plain TEX engine, pdfTEX, with as few as possible structuring commands borrowed from `BLUe.tex` — adhering minimal TEX mark-up. I use the Terminal font in the edit window with the pleasing effect that comments remain vertically aligned in the `.pdf` window. For checking the English spelling I use the public domain `en_GB` dictionary and hyphenation patterns `en_GB.aff` in TEXworks.

Prior to sending my `PDF`'s by email the files are optimized towards size by Acrobat Pro.[50] The bad news with respect to `.eps` into `.pdf` conversion is, that Acrobat 10 Pro X does not allow for the `run` command for library inclusion.

Errors of TEX ...

It is not told anywhere, but the rigid, bitmapped, unscalable CM-fonts is THE logical error of the twin TEX&Metafont, of which we suffer up till today.

> *"I spend a whole day on trying to create the Metafont-logo example, the Metafont book Appendix E, on my PowerMac of 1996. In vain, without results. I got io.300gf and io.tfm, but lacked the (old) tools to go on.*
> *The other day it took me roughly half an hour to create Adobe's example Type 3 font, Redbook p100. The Adobe's process is less complicated and not lumbered by confusing and complicating bitmap-inheritances from the past. For the purpose of creating Wordart in the spirit of Jackowski&Ryćko the Adobe Type 3 process is good enough."*

Moreover, I experience the boxes-approach as too rigid, little flexible, hampering for example easy page-breaks with floating (misplaced) pictures as result, as well as the impossibility to use footnotes, endnotes or ... from within a box; a 21st century tool unworthy.

Compared with PostScript, TEX's macro language is more complex, as can be seen from the examples in the paper. But ... we have to live with it, in want for something

49. PSView is extremely fast as previewer, allows PS library inclusion via the run command as well, reacts elegantly on errors by showing the results so far and supplies error messages via a pop-up GhostScript window, but ... doesn't provide for `.pdf` output, alas.

50. Courtesy Péter Szabó, EuroTEX-ConTEXt2009.

simpler, better, but also open source and equally-well documented.
Let us not make it more complicated by adding too big and too complex software.

TIA-simpler-WTDI

Errors of pdfTEX

The logical error in pdfTEX is that it does not allow for EPSF inclusion. The use of PostScript via `DVIPS` had earned its place in the TEXworld: rich, powerful and much used.

Another weak point is the lack of maintenance.[51] To develop a software tool is one thing to maintain it is quite another. A big disadvantage of the volunteer world: lack of follow-up.

Errors of TEXworks

Sometimes lines disappear in the edit-pane, as if printed on 1 line??? Very unhandy, I can't even edit these hidden lines.

Wishes under MS XP, MS Vista or MS System 7

For TEXworks I would like menu options `.eps` → `.pdf` and `.mp` → `.pdf`.
A decent IDE for MetaPost and PostScript.
Better PDF-viewer in TEXworks.
Accurate BoundingBox values via `pathbbox` in 1-pass.
BLUe as format in TEXworks.
Maintenance pdfAllTEX, and to allow for PostScript in pdfTEX.

Post-Conference

After my presentation Herbert Voss showed me his `PSTricks`,[52] which is a continuation and extension of the work of Timothy Van Zandt and Dennis Girou. Impressive, very impressive! Especially his 3D extensions.

`PSTricks` uses (harnessed) PostScript under the hood. The user-interface strongly reminds me of LaTEX-'s picture-environment. As far as I understand it, Timothy just implemented LaTEX's picture-environment in PostScript, via (one-way) interfacing. This entails that LaTEX users did not have to learn something new and received better value. However, the drawback is that the graphics is not backed up by an imaging model, and nasty things from the picture-environment are inherited.

It can't be used with pdf(La)TEX, because pdf(La)TEX does not allow for PostScript. Undoubtedly, the longer processing path via PostScript can be included as menu item in TEXworks, my TEX-editor.[53]

$$\text{Script} \xrightarrow{\text{pdfTEX}} \text{PDF} \qquad \text{vs} \qquad \text{Script} \xrightarrow{\text{TEX}} \text{DVI} \xrightarrow{\text{DVIPS}} \text{PS} \xrightarrow{\text{Distiller}} \text{PDF}.$$

In principle I favour the 3-steps process, in practice I use the 1-step fast way.

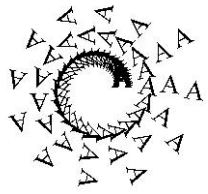
It's a pity that the code for the π-decimals picture, p294, has not been supplied in the book, so I can't compare it with my π-decimals code, as shown earlier and supplied in `PSlib.eps`.

Next best, I imitated Voss' example of rotated `A`'s, more-or-less, which reminds me of Adobe's rotated word `Adobe`, Bluebook p98. The picture is also given in the Graphics Companion p357.[54]

In `PSTricks`' code is too much one has to remember to my taste, too many and too varied braces, {...} , (...), and [...] ... moreover, the data `A` has to be supplied three times.

51. When working with colours the weird `\pdfliteral{1 0 0 0 k}` and `\pdfliteral{1 0 0 0 K}` have to be included?!?
52. PSTricks —Graphics and PostScript for TEX and LaTEX. UIT Cambridge. ISBN 978-1-906860-13-4. The Graphics Companion devotes to PSTricks: Ch 5 Harnessing PostScript inside LaTEX, and Ch 6 The main PSTricks packages, p213–p466, all-in-all 253p.
53. Although I don't know how to do that at the moment.
54. Another example of text along a spiral, explicit, is on p451 of the Graphics Companion, which comes close to typesetting along an implicit spiral.

Personally, I abhor the (curly) braces mania, and favour minimal mark-up; providing the data A three times is not minimal.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Herbert Voss, p233
%%BoundingBox: -40 -40 40 40
%%BeginSetup
%%EndSetup
/Times-Roman 10 selectfont 0 0 moveto /kern 5 def
0 .5 33{rotate kern 1 moveto (A) show
        /kern kern 1.029 mul def}for
```

```
\usepackage{pstricks, pst-node, multido}
\begin{pspicture}(4.5, 3.5)
   \cnode*(2,2){4pt}{A}
   \multido{\nA=0+10. \rB=)+0.5}{110}{%
      \nput[rot=\nA, labelsep\rB pt]%
           {\nA}{A}{A}}
\end{pspicture}
```

Acknowledgements

Thank you Adobe for your maintained, adapted to LanguageLevel 3 since 1997, good old, industrial standard PostScript and Acrobat Pro (actually DISTILLER) to view it, Don Knuth for your stable plain TEX, Jonathan Kew for the TEXworks IDE, Hàn Thế Thành for pdf(La)TEX,

Thank you Bogusław Jackowski for supplying me with old artistic material from GUST, and some more.

Thank you Herbert Voss for your comments, that we have met and that we stay on speaking terms.

Thank you Jos Winnink and Henk Jansen for proofing an early draft and the latter also for proofing the final version. MAPS editors for improving my use of English and last but not least Taco Hoekwater for procrusting my plain TEX preprint note into MAPS format.

James Ensor's impression of recreational 'Breskens'

So long and thanks for all the fish.
My case rests, have fun and all the best.

Kees van der Laan
kisa1@xs4all.nl

Appendix: BoundingBox via pathbbox in 1-pass

The verbatim left shows my current trial-and-error cropping, while at right cropping is done on-the-fly in 1-pass, at the expense of providing the path double.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Cropping
%%BoundingBox: 0 0 115 23
%%BeginSetup
%%EndSetup
/Times-Roman 30 selectfont
/rays{120{0 0 moveto 108 0 lineto 1.5 rotate
     }repeat stroke}def
0 1  moveto (StarLines) true charpath clip
newpath 50 -15 translate rays
showpage
%%EOF
```

```
%!PS-Adobe-3.0
%%Title: One-pass cropping, LRM 2
/Times-Roman 30 selectfont
0 0 moveto (StarLines) false charpath flattenpath pathbbox
/ury exch def /urx exch def ...
/w urx llx sub cvi def /h ury lly sub cvi def
<</PageSize [w h]>>setpagedevice
newpath
/rays{120{0 0 moveto 108 0 lineto 1.5 rotate
     }repeat stroke}def
0 1  moveto (StarLines) true charpath clip
newpath 50 -15 translate rays
showpage
%%EOF
```

Appendix: Binary tree macros

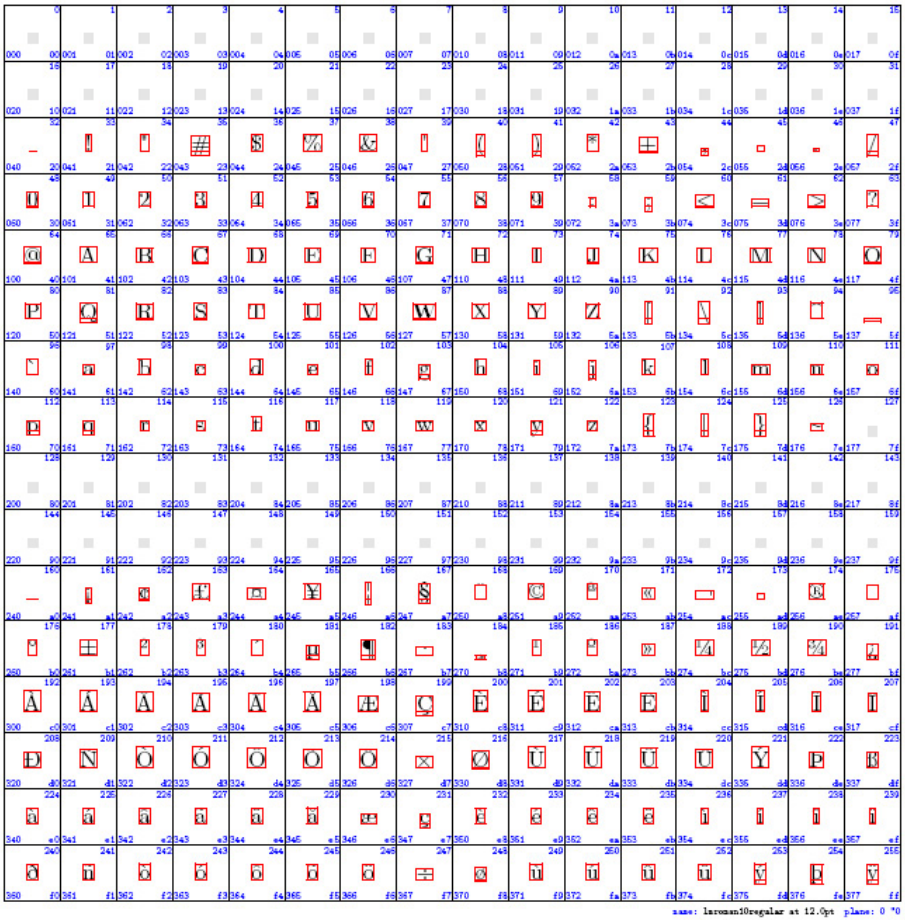
Included below are my stand-alone balanced binary tree TEX macros of old, taken from `tools.dat` with the necessary declarations from `blue.tex` added, next to my tiny, superior, and more clear PostScript variant on occasion of EuroTEX-ConTEXt2012.

```
\newdimen\x   \x0pt \newdimen\y   \y0pt \newcount\n   \newcount\k   \k0
\newdimen\unitlength \unitlength1ex     \newdimen\linethickness \linethickness1pt
\def\xy#1{%Function: place #1 at x, y
   \vbox to0pt{\kern-\y \hbox to0pt{\kern\x#1\hss}\vss}}
\def\xytxt#1{%Function: place text #1 at x, y
   \xy{\vbox to0pt{\vss \hbox to0pt{\strut#1\hss}\kern0pt}}}
\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}}}
\def\E#1{\xy{\vbox to0pt{\vss
   \hrule width#1\unitlength
          height\linethickness
          depth0pt\vss
  }}\advance\x#1\unitlength}
\def\W#1{\advance\x-#1\unitlength{\E{#1}}}
%
\def\bintree{\E{\the\n}%
   \ifnum\n=2 \eertnib\fi
   \divide\n2 {\N{\the\n}\bintree}%
              \S{\the\n}\bintree%
   \multiply\n2}
\def\eertnib#1\bintree{\fi}

\let\Eold\E
\def\E#1{\global\advance\k1
  \xytxt{\csname\the\k\endcsname}
  \Eold8}

%data
\def\1{CGL}\def\2{GJHvN}\def\3{JLB}\def\4{PvO}
\def\5{PT} \def\6{EF}   \def\7{WD} \def\8{HH}

\offinterlineskip \n=8 \bintree
\bye
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Binary Tree biased by Lindenmayer production rule
%%BoundingBox: -1 -250 250 250
%%BeginSetup
%%EndSetup
%%BeginProlog
/Bintree{% value of n on stack
  /n exch 2 div def
  E %draw East and add leave
  n 16 gt {currentpoint N n Bintree
           moveto        S n Bintree}if
   /n n 2 mul def}def %end Bintree
/N{0 n     rlineto}def
/S{0 n neg rlineto}def
/E{gsave  ntg k get 2 3 rmoveto show grestore
    /k k 1 add def
     60 0 rlineto
  }def
/Courier 16 selectfont /k 0 def
/ntg [(CGL) (GJHvN) (JLB) (PvO) (PT) (EF) (WD) (HH)
      (TH) (FG) (JW) (WD) (WE) (WW) (BJ)] def
%%EndProlog
%
% Program
%
0 -3 moveto 256 Bintree stroke showpage
```

The unbalanced tree jewel of old in cripple TEX, I should rewrite in PostScript, next to providing for viewing part of a huge tree, by some sort of window on the tree.

Appendix: Latin Modern Roman 16x16 font table (pane 1)



Above is included P0, with the usual glyphs (no euro, however), and some accented characters, AE ligature... Most of the 21 planes are nearly empty. The digits and the alphabet glyphs have the same digital address as in Knuth's 7bit table. Quite another question is how to use Latin Modern and TEX-Gyre.

P1: unusual accents and some double accented characters
P2: accents as such
P3: a few Greek symbols
P5: double embellished characters, 'accents, accents, accents ...'
P6: promille and euro with address 254, as well as pound sterling
P7: TM
P19: contains oldstyle digits beginning with address 060.

BLU does not much profit from all this generality; scientific communication in one language, English, simplifies enormously. Keep simplifying on your mind.

Julia fractals in PostScript

Fractal Geometry II

In memory of Hans Lauwerier

Abstract
Lauwerier's BASIC codes for visualization of the usual Julia fractals: JULIAMC, JULIABS, JULIAF, JULIAD, JULIAP, of the Mandelbrot fractal MANDELx, MANDIS, MANDET and his codes for the advanced circular symmetric Julia fractals JULIAS, JULIASYMm, JULIASYM, FRACSYMm, as well as the classical 1D bifurcation picture Collet, have been converted into PostScript defs. Examples of use are included. A glimpse into Chaos theory, in order to understand the principles and peculiarities underlying Julia sets, is given. Bifurcation diagrams of the Verhulst model of limited growth and of the Julia quadratic dynamical system — M-fractal — have been included. Barnsley's triples: fractal, IFS and equivalent dynamical system are introduced. How to use the beginnings of colours in PostScript is explained. How to obtain Julia fractals via Stuif's Julia fractal viewer, and via the special fractal packages Winfract, XaoS, and Fractalus is dealt with. From BASIC codes to PostScript library defs entails software engineering skills. The paper exhibits experimental fractal geometry, practical use of minimal TEX, as well as ample EPSF programming, and is the result of my next step in acquainting myself with Lauwerier's 10+ years work on fractals.

Keywords
Acrobat Pro, Adobe, art, attractor, backtracking, Barnsley, BASIC, bifurcation, Cauchy convergence, chaos, circle symmetry, dynamical systems, EPSF, escape-time algorithm, Feigenbaum constant, fractal dimension D, Fractalus package, FIFO, fractal geometry, IDE (Integrated development Environment), IFS (Iterated Function System), Julia, Lauwerier, Mandelbrot, mathematical software, minimal encapsulated PostScript, minimal plain TeX, Monte Carlo, μ-geometry, periodic doubling, Photoshop, PSlib, PSView, repeller, self-similarity, software engineering, Stuif's previewer, TEXworks, (adaptable) user space, Verhulst growth model, Winfract package, XaoS fractal package.

Contents

Introduction

Gaston Julia, a French mathematician, in his award-winning paper of 1918 discovered fractals, theoretically, avant la lettre. He studied among others recursions of the type

$$z_{i+1} = z_i^2 + c, \quad z_i, c \in \mathbb{C} \quad i = 0, 1, 2, ...$$

Fixed-points: $\{l_{1,2} \mid l = l^2 + c\} \rightarrow l_{1,2} = .5 \pm .5\sqrt{1 - 4c}$. ∞ is attractor for $|z_0|$ sufficiently large.

The boundary of the domain in $\mathbb{C}$ for which z_0 does not fly away to ∞ is called a Julia fractal.

Julia discerned three classes of fractals: plane-filling (rare), dust-like and connected (contours in contours in ...).

The answer to the question by Mandelbrot[1]

> "For which values of c will the Julia fractal, J(c), be line-like and for which dust-like?"

marks the development of fractal geometry.

In the sequel we will

- ☐ discuss methods for drawing Julia repellers in PostScript
- ☐ show various Julia fractals
- ☐ show the use of `Stuif's Julia previewer`
- ☐ show the use of `WinFract`, `XaoS`, `Fractalus`[2]
- ☐ explain the various conversions in the appendices, with the `defs` included in my `PSlib.eps`
- ☐ tell how the beginnings of colouring can be done in PostScript, in the last appendix.

Although the paper is physically thick, I hope that the reader will recognise the logicallly thin parts. Parts can be read independently.

The section Catching-up is for the unwary. The section Barnsley's Triples is food for mathematicians. In the footsteps of Lauwerier the reader is invited to experiment with the PostScript programs and redo Lauwerier's exercises as posed in his booklets. Playing with `Stuif's Julia previewer`, `WinFrac`, `XaoS` or `Fractalus` is fun, and ... fascinating.

The LaTeX Graphics Companion contains some rudimentary classical Math fractals in MetaPost in section 4.4.3. No systematic programming approach à la Lindenmayer enriched with PostScript concepts, nor the use of PostScript's powerful facility to transform User Space, nor minimal PostScript, nor EPSF, are mentioned.[3]

The paper is my next step in acquainting myself with Lauwerier's 10+ years work on fractals.

The XaoS movie is an appetizer to the world of Julia fractals and its associated Mandelbrot map.

Catching up: quadratic dynamical systems

The purpose of this section is to make the unwary aware of the fact that innocent-looking iterative processes might converge, might bifurcate, repeatedly, or finally end-up in chaos.

---

1. Benoît B. Mandelbrot(Warsaw 20 November 1924 – Cambridge 14 October 2010) was a French American mathematician. Born in Poland, he moved to France with his family when he was a child. Mandelbrot spent much of his life living and working in the United States. Barnsley, M.F.& M.Frame(2012): Influence on Benoît Mandelbrot on Mathematics. `http://www.ams.org/notices/201209/index.html`.
2. `WinFract`, `XaoS` and `Fractalus` are fractal packages with rich colour and zoom facilities, for 'free to use' available on the WWW under the GNU public license. Stuif's Julia previewer is a Java Applet accessible via his WWW.
3. Understandable from the viewpoint to refrain from lower-level tools like PostScript.

In my education we practised a lot of iteration processes,[4] for example in the various zero finding processes. The accompanying picture shows the usual convergence for the determination of the intersection point of the parabola $y = x^2 + c$ and the line $y = x$ with $c \in [-.75, 25]$, for example c=-.5.

Julia Model with c=-.5 $x_{i+1} = f(x_i) = x_i^2 - .5, \quad i = 0, 1, 2, \ldots$
Fixed-points of $f(x)$:

$$l = l^2 - .5 \quad \rightarrow \quad \begin{aligned} l_1 &= .5(1 + \sqrt{3}) \approx 1.367 \\ l_2 &= .5(1 - \sqrt{3}) \approx -.367 \end{aligned}$$

Stability at fixed-points of $f(x)$:

$$|f'(l_{1,2})| = |2l_{1,2}| \quad \rightarrow \quad \begin{aligned} |f'(l_1)| &= |1 + \sqrt{3}| \approx 2.7321, \text{ repeller} \\ |f'(l_2)| &= |1 - \sqrt{3}| \approx 0.7321, \text{ attractor.} \end{aligned}$$

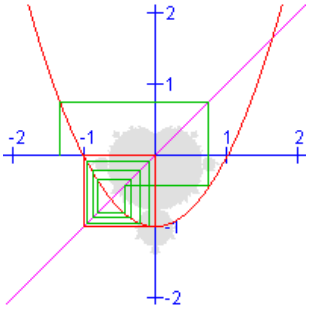
Critical point of $f(x)$: $\{x \mid f'(x) = 0\} \quad \rightarrow \quad x = 0.$

Julia Model with c=.25, the cusp in the M-fractal: $x_{i+1} = f(x_i) = x_i^2 + .25, \quad i = 0, 1, 2, \ldots$

The intersection point becomes a tangent point. For $x_0 \in [-.5, .5]$ the dynamical system converges; for a starting value outside the interval the dynamical system diverges.

But ... what if we take $c = -1$?

On the World Wide Web, WWW for short, I found a fractal primer,[5] aimed at a broad audience with Math at the high-school level. Julia sets, Fatou domains, (strange) attractors, the Feigenbaum number and bifurcation are introduced. Stuif discusses the intersection point of the parabola, $y = x^2 - 1$, and the straight line $y = x$, supported by (Java) applets for experimentation.
Important concepts in dynamical systems are: fixed-points and stability at the fixed-points, the critical points, and the so-called strange attractors c.q. repellers.



Julia Model with c=-1 $x_{i+1} = f(x_i) = x_i^2 - 1, \quad i = 0, 1, 2, \ldots$[6]
Fixed-points of $f(x)$:

$$l = l^2 - 1 \quad \rightarrow \quad \begin{aligned} l_1 &= .5(1 + \sqrt{5}) \approx 1.618, \qquad \phi \\ l_2 &= .5(1 - \sqrt{5}) \approx -.618, \quad -1/\phi \end{aligned}$$

Stability at the fixed-points of $f(x)$:

$$|f'(l_{1,2})| = |2l_{1,2}| \quad \rightarrow \quad \begin{aligned} |f'(l_1)| &= |1 + \sqrt{5}| > 1, \text{ repeller} \\ |f'(l_2)| &= |1 - \sqrt{5}| > 1, \text{ repeller.} \end{aligned}$$

Critical point of $f(x)$: $\{x \mid f'(x) = 0\} \quad \rightarrow \quad x = 0.$
Because of the repeller nature at the fixed-points we don't expect convergence.

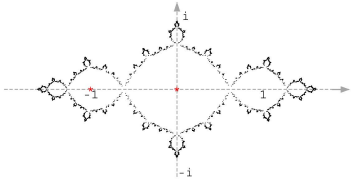
4. Fractals raised a new awareness of, a new insight in, iterative processes as can be witnessed from the title of Mandelbrot's invited paper 'Fractals and the Rebirth of Iteration Theory,' in Peitgen c.s.(1986)
5. http://www.stuif.com/fractals/index.html
6. The general case with c complex, will be treated later.

Non-convergence, with a flip-flop behaviour with values 0 and -1, the so-called strange attractors. The splitting-up is called bifurcation.

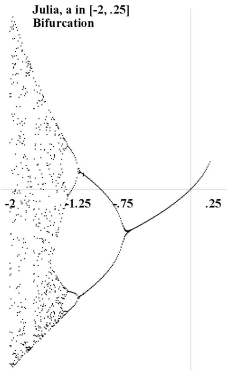
Bifurcation condition for 2-period:

$$\begin{cases} x &= w^2 - 1 \\ w &= x^2 - 1 \end{cases} \quad \rightarrow \quad \begin{matrix} x &= 0 \\ w &= -1 \end{matrix} \qquad \text{the flip-flop behaviour.}$$

The associated Julia fractal, J(-1), picture looks like



Starting points within the Julia fractal, J(-1), don't fly away to infinity. They either stay on the fractal in a chaotic way or converge to the strange attractors 0 and -1.



In the accompanying bifurcation diagram, the behaviour of the Julia dynamical system for real parameter, a, is summarized.
For $a \in [-.75, .25]$ there is the usual convergence.
For $a \in [-1.25, -.75]$ 2-periodic bifurcation pops up.
A little further higher bifurcations come into light as well as the chaotic behaviour
Beyond -2 the Julia fracal flies away to ∞ except for the dust Julia fractals.
In literature I did not stumble upon the bifurcation diagram for the Julia system with real parameter.
The general complex parameter bifurcation diagram is the M-fractal.

Verhulst limited growth model is related to the Julia dynamical system. Moreover, Lauwerier has extensively studied this model; his results have been summarized below.

Verhulst model $x_i = f(x_{i-1}) = a x_{i-1}(1 - x_{i-1}) \quad i = 0, 1, 2, \ldots, \quad 0 < a \leqslant 4.$
Fixed-points of f(x):

$$l = a l(1 - l) \quad \rightarrow \quad \begin{matrix} l_1 = 0 \\ l_2 = 1 - 1/a \end{matrix}$$

Stability at the fixed-points of $f(x)$:

$$|f'(l_{1,2})| = |a(2l_{1,2} - 1)| \; \rightarrow \; \begin{matrix} f'(l_1) = |a|, & \text{attractor for } 0 < a < 1 \\ f'(l_2) = |a - 2|, & \text{attractor for } 1 < a < 3 \end{matrix}$$

The last 20 values of $x_n = a x_{n-1}(1 - x_{n-1}), \quad n = 0, 1, 2, \ldots, 100$, are shown below for a few values of a; the (non)convergence behaviour is striking.

Bifurcation condition for 2-period:

$$\begin{cases} x = w(aw - (a-1)) \\ w = x(ax - (a-1)) \end{cases} \quad \rightarrow \quad \begin{matrix} x = (1 + a + \sqrt{a^2 - 2a - 3})/(2a) \\ w = (1 + a - \sqrt{a^2 - 2a - 3})/(2a) \end{matrix}$$

The class of iterations for non-linear functions have earned a place of their own in nowadays Math. The field is called dynamical systems.

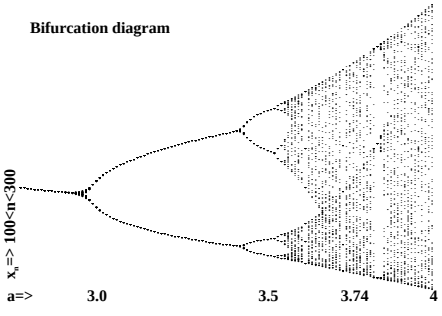
Behaviour Verhulst model

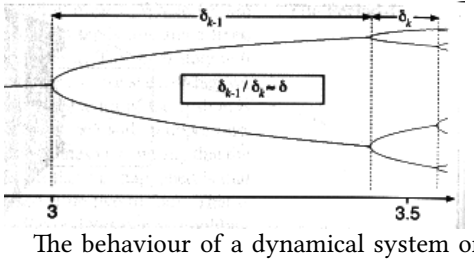
Lauwerier(1987, ch 6, paragraph Getal van Feigenbaum) summarizes the behaviour of the Verhulst model as function of a:[7]

$0 < a <= 1$	$\Rightarrow$	$x_n \to 0$	super stable equilibrium
$1 < a <= 2$	$\Rightarrow$	$x_n \to 1 - 1/a \approx .6875$	monotone, stable equilibrium
$2 < a <= 3$	$\Rightarrow$	$x_n \to 1 - 1/a \approx .6875$	oscillating, stable equilibrium
$a \approx 3.2$	$\Rightarrow$	$x_n \to \approx .51$ and $\approx .78$	2-cycle bifurcation
$a \approx 3.45$	$\Rightarrow$	$x_n \to \approx .50, \approx .88, \approx .38, \approx .83$	4-cycle bifurcation
$a \approx 3.54$	$\Rightarrow$	$x_n \to$ ...	8-cycle bifurcation
$a \approx 3.83$	$\Rightarrow$	$x_n \to$ ...	3-cycle bifurcation
$a \approx 3.84$	$\Rightarrow$	$x_n \to$ ...	6-cycle bifurcation
$3.56 < a <= 4$	$\Rightarrow$	$x_n \to$ ...	chaotic or periodic

The bifurcation diagram repeats itself if we zoom in. It behaves like a fractal. Isn't a miracle that we could find zeroes in the past at all?

Lauwerier's tiny bifurcation diagram of 1987, his program `COLLET`, has been improved with better legend and anachronisms with respect to screen-size has been removed in PostScript. It is called `bifurcation` of which the result has been included. The numbers 3.0 etc. indicate for which values of a a periodic doubling, or otherwise, shows up. Lauwerier(1996) discusses in detail the bifurcation diagram. Moreover, programs are provided for showing magnified parts of the diagram. He also treats the case a=4, analytically. The various periodic behaviours and the chaos are not specific for this function. In 2D, and for complex values of z_0, the Julia set gave rise to interesting pictures: fractals!
Informally, the Fatou set of the function consists of values with the property that all nearby values behave similarly under repeated iteration of the function, and the Julia set consists of values such that an arbitrarily small perturbation can cause drastic changes in the sequence of iterated function values.





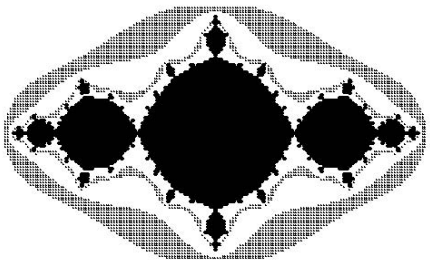
Chaos research yielded
Feigenbaum's universal constant $\delta \approx 4.6692$:
the quotient of successive bifurcation lengths.
Feigenbaum's constant appears in physics, chemistry, ...

The behaviour of a dynamical system on the Fatou set is 'regular', while on the Julia set its behaviour is 'chaotic'. In the case $z^3 = 1$, see the accompanying illustration, courtesy Georg-Johann Lay Wikipedia, when starting with a point on the 'necklace' we can't tell beforehand to which zero the process will converge.
Explanation. The Julia set (in white) is visualized which associates Newton's method for locating the zeroes of the 3rd degree polynomial equation by the iteration $z_{i+1} = \frac{2}{3}z_i + \frac{1}{3z_i^2}$ to the Julia fractal. The colours indicate which starting point converges to which zero. It shows that close-by starting points converge to different zeros in an irregular, intriguing pattern.

7. For a=3.2 substitution in the above formula yielded .8 instead of .78. Maybe 3.2 is not accurate enough?

Barnsley's triples

Barnsley(1988) considers triples: dynamical system, equivalent IFS and attracting fractal. His escape time algorithm is similar to the JULIAP casu quo JULIAF methods. The theoretical, rigorous math approach is food for mathematicians. The accompanying J(-1), by Barnsley's program, has a contour for which the escape number is 5. The broader band are the points with escape number 2. The points with escape number infinity are considered to lie on and within the fractal, the raison d'être of the escape-time algorithm.



Julia fractals

Properties

- ☐ is a point on the Julia set, J, then all its successors and predecessors are on J
- ☐ the Julia set is point symmetric, $z^2 = (ze^{i\pi})^2$, which property is used to save computation time
- ☐ J(c) is x-axis mirror symmetric with J($\bar{c}$), like the Mandelbrot fractal
- ☐ there are 3 kinds of Julia fractals: plane-like, line-like and dust-like
- ☐ if $\lim_{k\to\infty} z_k(a,b) = \infty, z_0 = 0$ then J(a,b) is dust-like
- ☐ the points around the fractal are repelled from the fractal, J(a,b) is a repeller.

Relation Verhulst model and the Julia dynamical system

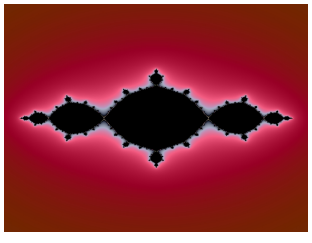
The Julia dynamical system is a sort of 2D extension of the Verhulst growth model, because of working in $\mathbb{C}$, with all the peculiarities of the 1D Verhulst model inherited, Lauwerier(1996, ch8).

Verhulst: $\xi_n = a\xi_{n-1}(1-\xi_{n-1})$ $\overset{\xi=.5-x/a}{\longrightarrow}$ Julia: $x_n = x_{n-1}^2 + a/2 - a^2/4$

Julia: $x_n = x_{n-1}^2 + c$ $\overset{x=a(.5-\xi)\wedge c=a/2-a^2/4}{\longrightarrow}$ Verhulst: $\xi_n = a\xi_{n-1}(1-\xi_{n-1})$

Mandelbrot's map of Julia fractals

Mandelbrot was interested in: for what values of c is a Julia fractal line-like and for what values dust-like. The 'map of Julia fractals', the bifurcation diagram, is the famous fractal named after Mandelbrot, M-fractal for short. When e.g. the white + in the M-picture, coordinates (-1, 0), is pointed at by the cursor, with the Julia mode active of `Fractalus`, then the J(-1, 0) fractal is opened in another window.

→ 

Peitgen c.s.(1986) considered the 'M-fractal as map' important as can be witnessed from the full-page illustration by Milnor.
Barnsley(1988) contains a similar picture.
It gives a survey of the various interesting Julia fractals and their position in the M-fractal.

Zooming-in, or ... going for details

Drawing fractals as such is one thing. Visualization of the endless repetition, the microscopic detail, is another. The (near) repetition of the main picture in the details, this near to self-similarity property, as well as the new forms that arise make fractals intriguing.

Zooming-in the M-fractal, zooming-in again, again and again is shown in the accompanying picture, which is borrowed from the Mandelbrot Wikipedia.

Lauwerier(1994, 1996) contains the programs MANDET(ail) and MANDIS(tance) for visualization of a detail of the M-fractal. The J-fractals and the M-fractal have their playground on the square inch: μ-geometry!

Lauwerier(1994) also provided for a zoom program, MZOOML, in BASIC (3p), which I consider no longer relevant. It is overruled by the easy to use zoom facilities of Winfract, XaoS, Fractalus, ...

Use of the various PostScript defs

The PostScript defs expect on the stack: the values for the problem parameter a, b, the coordinates of the fractal domain specified by the values for the upper-right corner $\mathbf{x}_{ur}$, $\mathbf{y}_{ur}$, and the maximum number of iterations. The sharper the specifications of the fractal domain the faster the program. In the header comment of the EPSF program the values for the BoundingBox, BB for short, should be supplied in order to get a cropped picture. The BB-values are default 100 (scaling factor) times the fractal domain values.

Fractal	a	b	$\mathbf{x}_{ur}$	$\mathbf{y}_{ur}$
San Marco	-1	0	1.65	0.85
Dragon	-0.7454	0.1103	1.5	0.9
Douady	-0.12	0.74	1.2	1.3
Douady conj	-0.12	-0.74	1.2	1.3
Dendrit	0	1	2.1	1.85
Lightning	-1.029	0.386	1.65	0.85
Leaves	0.11	0.66	1.3	1.3
Cusp	0.25	0	0.85	1.1
4 spirals	0.55	0	0.85	0.85
Cloud	-0.59	-0.34	1.45	1

-1.5 0 5000 JULIAMC
San Marco (flat)

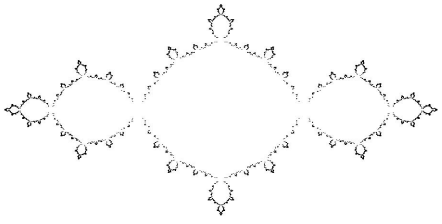
Using JULIAMC

The def JULIAMC implements inverse iteration. The parameters of the def JULIAMC are: the real and imaginary part of the problem parameter c, i.e. a and b, and the number of points of the fractal to be generated, n. It is the simplest Julia fractal generator.

$$z_i = z_{i-1}^2 + c, \quad i = 1, 2, 3, \ldots \qquad \rightarrow$$

$$\text{Inverse:} \quad z_{i-1} = \pm\sqrt{z_i - c}, \quad i = n + 10, \ldots, 1, \quad |z_{n+10}| <= 1.$$

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -165 -85 165 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run%PSlib(rary) inclusion
%%EndProlog
%
% Program ---the script---
%
-1 0 5000 JULIAMC         %SanMarco
showpage
%%EOF
```



In order to have the picture cropped for EPSF use appropriate BB-values have to be inserted.

The Julia set for $z_i = z_{i-1}^2 + c, \quad i = 1, 2, 3, \ldots$ is a repeller; the Julia set for the inverse iteration is an attractor. The inverse operation is a 2-valued function because of the square root. It is enough to use one of the values randomly. This random use explains the suffix MC, from the Monte Carlo casino, in the name.

The BASIC code JULIAMC and its conversion into a PostScript def is discussed in the first appendix. If you, kind reader, don't want to bother with the use of the PostScript library, then just include the PostScript def, in place of the library inclusion as follows.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -165 -85 165 85
%%BeginSetup
%%EndSetup
%%BeginProlog
```

```
/JULIAMC{%stack: a, b, maxk.
        %==> Julia set of z^2 + a + ib via Inverse Iteration
...
end}bind def
%also include the corresponding dictionary
/JULIAMCdict 11 dict def
%also centershow for centering the dots
/centershow{%string --> show string centered
gsave dup stringwidth pop 2 div neg 0 rmoveto show grestore} def
%%EndProlog
%
% Program ---the script---
%
-.75 0 5000 JULIAMC%SanMarco
showpage
%%EOF
```

-.75 0 5000 JULIAMC, San Marco, D≈1.27

This inclusion makes clear how handy a `PSlib.eps` is. It keeps a program small and clear, with all `def`s stored together, separately and well-organized. For exact `BB`-values use the values prompted by `pathbbox` and print the values via the library `def showobject`. Finally, insert these values, `llx lly urx ury` in the header comment `%%BoundingBox: llx lly urx ury`, for an exact cropped picture in the next run, where the picture can also be reused as `EPSF`. A 2-pass process.[8]

0 1 5000 JULIAMC
Dendrit, D≈1.2

-.8 .15 5000 JULIAMC
Dragon

.25 0 5000 JULIAMC
Club

.11 .66 5000 JULIAMC
Leaves

-1.03 .386 5000 JULIAMC
Lightning

.55 0 5000 JULIAMC
4 Spirals

In the figures the lines are not of equal thickness, some parts are even blank. To compensate for this there are the boundaryscan method and the distance formula method.

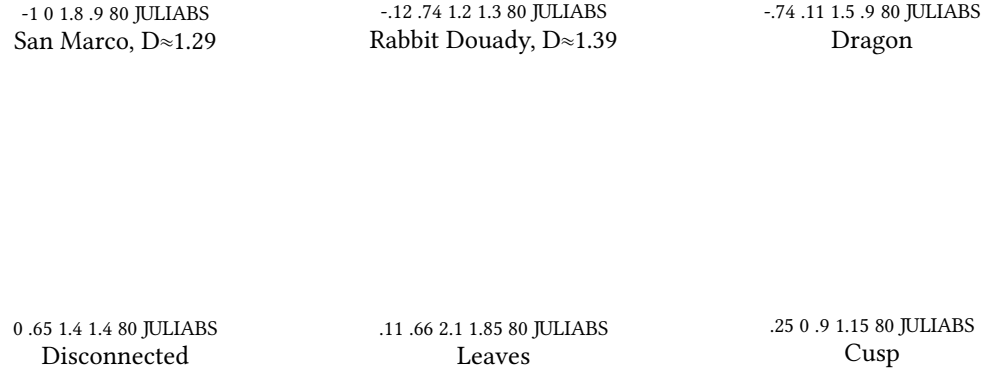
Using JULIABS

`JULIABS` implements the Boundary Scan method. The parameters of the `def JULIABS` are: the real and imaginary part of the problem parameter `c`, i.e. `a` and `b`, the upper-right corner of the rectangle where the fractal is located, i.e. x_{ur}, y_{ur}, and the maximum number of iterations per grid point, i.e. `kmax`.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -180 -90 180 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run  %inclusion of PSlib(brary)
%%Endprolog
```

8. How to obtain the same effect in a 1-pass job for EPSF, see Recreational use of TEX&Co. This proceedings

```
%
%   Program ---the script---
%
1 .7 scale 0 0 195 0 360 arc%scale (circle to oval)
lightblue fill              %fill oval by background colour
blue -1 0 1.8 .9 80 JULIABS %blue San Marco
showpage
%%EOF
```



There is no contour as path, just dots. From this collection of points I was not able to construct a contour in PostScript; in Photoshop, yes. Dendrit and Lightning were not obtained by this method.

Using JULIAF

`JULIAF` implements Filling the area of the fractal. The parameters of the `def JULIAF` are: the real and imaginary part of the problem parameter `c`, i.e. `a` and `b`, the upper-right corner of the rectangle where the fractal is located, i.e. x_{ur}, y_{ur},[9] the maximum number of iterations per grid point, i.e. `kmax`.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -150 -90 150 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%EndProlog
%
%   Program the script
%
gsave
1 .7 scale 0 0 150 0 360 arc %oval
lightblue fill               %fill oval by background colour
grestore                     %end effect of scaling
blue -.8 .15 1.5 .9 50 JULIAF%blue Dragon
showpage
%%EOF
```

9. Default the BB values are a factor 100 times the values of the rectangle where the fractal is located.

-1 0 1.65 .85 80 JULIAF
San Marco

-.12 .74 1.3 1.2 80 JULIAF
Rabbit of Douady

-.55 0 1.45 1 50 JULIAF
Rounded San Marco

Using JULIAP

JULIAP implements the Pixel method. The parameters of the def JULIAP are: the real and imaginary part of the problem parameter c, i.e. a and b, the upper-right corner of the rectangle where the point-symmetrical fractal is located, i.e. x_{ur}, y_{ur}, and the maximum number of iterations per grid point, i.e. kmax.

Colour-banded Julia fractals are obtained. The number of iterations, the escape number, needed to 'escape to ∞,' is used for the selection of the colour. The closer the initial point z_0 lies to the fractal the longer it takes to pass the threshold of the ∞-attractor, i.e. the larger the escape number. For the points within the fractal the escape number is defined by the number of iterates for which 2 successive iterates are close enough to each other, the Cauchy criterion. For the case of periodic cycles the escape number is the maximum number of iterates, i.e. kmax.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -160 -140 160 140
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %Inclusion of PSlib(rary)
%%EndProlog
%
%   Program, the script
%
gsave
1 .7 scale 5 0 85 0 360 arc%ellips
lightblue stroke           %background colour ellips
grestore                   %end effect of scale and colour
-.55 0 1.6 1.4 50 JULIAP   %colour-banded Julia fractal
showpage
%%EOF
```

-1.3 0 1.65 .85 80 JULIAP
San Marco flat var.

-.12 .74 1.3 1.2 80 JULIAP
Rabbit of Douady

.11 .66 1.3 1.3 80 JULIAP
Leaves

We might ask ourselves the question:
How to colour by gradients?
The primitive way is to scale and colour the scaled-lines by slightly changing colours.
Another way is to make a contour of the fractal, in Photoshop for the time being, and apply colouring by gradients. The accompanying cloud example has been obtained in 2009 in this way; the doughnut earlier.

PostScript 3 provides facilities for graceful and high-quality colouring by gradients

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 100 400 400 600
100 400 300 200 rectclip
<</ShadingType 2 /ColorSpace [/DeviceRGB] /Coords [100 400 400 600]
  /Function<</FunctionType 2
             /Domain [0 1]
             /C0 [1 0 0] /C1 [.5 1 .5]%colours at end points
             /N 1 >>
>>
shfill%shadefill
showpage
```

Using the powerful JULIAD

JULIAD implements the use of the Distance formula. The parameters of the def JULIAD, are: the real and imaginary part of the problem parameter c, i.e. a and b, the upper-right corner of the rectangle where the fractal is located, i.e. x_{ur}, y_{ur}, the maximum number of iterations per grid point, i.e. kmax, and the distance formula parameter, e.g. 0.01. The pictures below show the richness of detail obtained.

-.775 .1103 1.5 .9 75 .01 JULIAD

-.7454 .1103 1.5 .9 75 .01 JULIAD

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -150 -90 150 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run            %Inclusion of the PSlib(brary)
%%EndProlog
%
%   Program, the script
%
gsave
1 .7 scale 0 0 150 0 360 arc          %oval
lightblue fill                        %fill the oval by background colour
```

```
grestore
blue -.7454 .1103 1.5 .9 75 .01 JULIAD%blue, scaled inner structured Dragon
showpage
%%EOF
```

-.74 .11 1.5 .9 75 .01 JULIAD
Dragon

-1.03 .386 1.65 .85 50 .01 JULIAD
Lightning

-.55 0 2.1 1.85 50 .01 JULIAD
Nesting of clouds

-1.3 0 1.65 .5 50 .01 JULIAD
San Marco flat var.

-.12 .74 1.2 1.3 50 .01 JULIAD
Rabbit of Douady

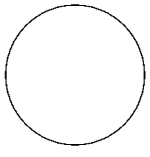
.11 .66 1.3 1.3 50 .01 JULIAD
Leaves

-1 0 1.65 .85 50 0.1 JULIAD
San Marco, f=.1

-1 0 1.65 .85 50 .01 JULIAD
San Marco, f=.01

-1 0 1.65 .85 50 .001 JULIAD
San Marco, f=.001

Compared with earlier methods J(.11, .66) looks different, apparently because of the more accurate method. This raises the question: What does the real fractal look like?



0 0 1.1 1.1 50 0.01 JULIAD

Degenerated cases
$\leftarrow$ J(0), a circle
J(-2), a line $\rightarrow$
J(0) $\xrightarrow{z+\frac{1}{z}}$ J(-2)

-2 0 2.1 0.1 50 .01 JULIAD

J(-3.45) is disconnected dust

The invoke `-3.45 0 3 .25 50 .0000001 JULIAD` did not yield more details; the dust reminds me of Cantor. The dust does not contain more details, apparently.

I was curious whether once one point has been found of the dust, the others would be obtained by gambling inverse iteration: 4 values were found $\pm 2.62, \pm 1.98$.

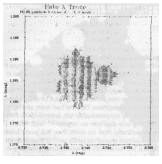
Using MANDEL and variants

Mandelbrot in 1980 answered the question posed in the introduction:

For which values of c will the Julia fractal, J(c),
be line-like and for which values dust-like?

He was surprised, but ... realized the relevance.

The picture consists of a cardioid, some circular bulbs, hairy details and stretching out with an 'antenna'. So nice to find a real-life application where a classical Math contour, cardioid, pops up.



Mandelbrot's first M-fractal

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox:  -210 -135 85 135
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%Endprolog
%
%   Program ---the script---
%
MANDEL
%respectively
%MANDELzw
%MANDELzwcontour
showpage
%%EOF
```

Mandelbrot also elaborated on the fractal dimension notion. The M-fractal curve, and surface, have fractal dimension D=2.

Fixed-points of the Julia quadratic dynamical system: $\{l \mid l = l^2 + c\}$.

Stability of the dynamical system: $|f'(z)| < 1$.
$z = e^{i\varphi}/2$ lies on the boundary of $|f'(z)| = 1$, i.e. $|2z| = 1$.
Substitution of $z = e^{i\varphi}/2$ in the equation for the fixed-point, yields

$$c = a + ib = e^{i\varphi}/2 - e^{2i\varphi}/4 \rightarrow \begin{cases} a = \cos(\varphi)/2 - \cos(2\varphi)/4 \\ b = \sin(\varphi)/2 - \sin(2\varphi)/4 \end{cases},$$

the equation of a cardioid.
The c-values for which J(c) has 1 attracting fixed-point lie within the M-cardioid.

2-period bifurcation of $f(z) = z^2 + c \rightarrow \begin{cases} z = w^2 + c \\ w = z^2 + c \end{cases} \quad \wedge \quad |f'(z)f'(w)| < 1.$

From the 2 equations we yield $zw = c + 1$. Together with $|f'(z)f'(w)| = 4zw$ we arrive at $|c + 1| < 1/4$, the circle left of the cardioid. The 2-period bifurcation occurs for c in the circle $C_{(-1,\frac{1}{4})}$.

More, and higher, bifurcations are in the necklace of surrounding circle-shaped bulbs, which themselves have surrounding necklaces in a fractal way.[10]

The most amazing of the Mandelbrot fractal is that by magnification slightly different repetitions of the main picture show up.

Wim W. Wilhelm's Mandelbrot fractal via Asymptote Wim works with Asymptote and LaTeX within the TeXnicCenter IDE, and also uses Mathematica. I have included his program and picture (circular-cropped by Photoshop) to please Asymptote-LaTeX users. Handy are the complex data type `pair`, the Metafont-like `path` datatype and the reflection operator about a line, next to the C++-like language features.

```
size(10cm,0);
real mandelbrot(pair c, real r, int count=100)
{int i=0; pair z=c;
 do { ++i; z=z^2+c;} while (length(z) <= r && i<count);
 return (i<count) ? i/count : 0;}
real r=4, step=.01, xmin=-2.25, xmax=.75, ymin=-1.3, ymax=0;
```

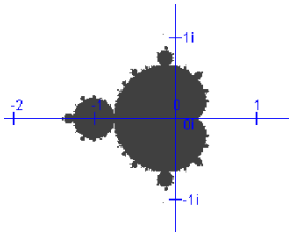
10. Lauwerier(1996, p195).

```
real x=xmin, y=ymin;
int xloop=round((xmax-xmin)/step);
int yloop=round((ymax-ymin)/step);
pen p; path sq=scale(step)*unitsquare;
for(int i=0; i < xloop; ++i)
 {for(int j=0; j < yloop; ++j)
  {p=mandelbrot((x,y),r,20)*red;
   filldraw(shift(x,y)*sq,p,p);
   y += step;
  }x += step; y=ymin;
 }
add(reflect((0,0),(1,0))*currentpicture);
```

Using MANDET

The form and size of the M-fractal is known and depicted in the left picture below, courtesy Stuif. Lauwerier provides programs for viewing details of the M-fractal: MANDET(ail) and MANDIS(tance). The first 2 parameters of MANDET denote the centre of the rectangle of the fractal domain, i.e. ac and bc (e.g. in 6 decimals). The third parameter, d, is half the width, and height,[11] of the fractal rectangle to be viewed. The 4th parameter is the maximum number of iterations, kmax, of the inner-loop. After termination of the inner-loop, the value of the inner-loop variable is used as escape number for determination of the colour.

I chose for MANDET and MANDIS the rectangular BoundingBox: -400 -300 400 300, wired-in, conform the shape of the M-fractal; no deformation by scaling. The grid-points $\{i \mid i = -400, -399, \ldots, 399, 400\}$ and $\{j \mid j = -300, -299, \ldots, 299, 300\}$ of the BoundingBox are scaled down to the grid-points of the fractal square by $\{(a_i, b_j) \mid (ac + d * i/400,\ \ bc + d * j/400)\}$. The approach is similar to Lauwerier, who maps the full-screen on the fractal area.



Form and size of M-fractal

Mapping BoundingBox onto fractal rectangle

By halving the d parameter the magnification is doubled. More details are obtained by increasing kmax.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox:  -400 -300 400 300
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%Endprolog
%
%   Program ---the script---
%
-1.953712 0 .000049 100 MANDETzw%Black and White variant
% ac      bc d      kmax        %parameter mnemonics
showpage
%%EOF
```

11. The latter is wired-in, scaled in the def to 75%, in accordance with the shape of the M-fractal

a	b	d	kmax	time
-1.953712	0	0.000049	200	
-1.927199	0	0.0005	200	31
-1.749057	0.000306	0.000004	300	
-1.28408	0.42726	0.000625	100	75
-1.256362	0.38032	0.008	100	74
-0.91667	0.26667	0.06	200	90
-0.7789	0.1344	0.0032	150	
-0.7777	0.1355	0.0037	150	125
-0.74691	0.10725	0.0008	800	744
-0.746371	0.098641	0.00001	300	
-0.7392	0.1745	0.0028	200	123
-0.698	0.3785	0.004	100	113
-0.160651	1.036793	0.00001	600	
-0.15067	1.04504	0.000006	200	
-0.102324	0.95716	0.000007	200	
-0.1011	0.9563	0.0016	200	75
-0.023262	0.999253	0.003	100	59
-0.01556	1.02071	0.0015	100	63
0.28	0.28	0.00028	400	
0.2812	0.00948	0.00005	300	
0.2813	0.0107	0.002	200	184
0.25	0	0.1	100	
-0.75	0	1	100	
-1.25	0	1	100	

`-1.927199 0 .0005 100 MANDET` `-1.25636 0.38032 .08 100 MANDET`

`-1.749057 0.000306 .04 100 MANDET` `-0.7489 0.1073 0.004 100 MANDET`

Parameter values for `MANDET` and `MANDIS` of interesting areas to be magnified have been supplied in Lauwerier(1994, 1996(with timings)) and have been copied in the accompanying table, with some values added by me, prompted by `Fractalus`.

The timing of Lauwerier's most expensive detail with centre (-0.74691, 0.10725), width d=0.0008 and maximum of iterations kmax=100,[12] took ≈38s by `PSView` and ≈40s by `Acrobat Pro`. Nearly a factor 20 faster than Lauwerier's timings. The included pictures are more impressive when viewed full-screen.

A few stepwise magnifications have been given below. Compared to the professional pictures, even with Peitgen c.s.(1986) of 25 years ago, the 'Museum pictures,' the results are just an amateur's[13] beginnings.

`-.7454 0 1.5 100 MANDET` `-.7454 0 1 100 MANDET` `-.7454 0 .5 100 MANDET` `-.7454 0 .25 100 MANDET`

Using MANDIS

In order to visualize the hairy details the distance formula is used in `MANDISm`, where the suffix m denotes monochrome. The first 2 parameters denote `c`, i.e. `a` and `b` (e.g. in 6 decimals). The third parameter is half the width of the fractal domain to be viewed. The 4th parameter is the maximum number of iterations, i.e. `kmax`, of the inner-loop. The value of the loop variable is used as escape number for determining the colour. The fifth parameter, `f`, is used as threshold for the closeness to the M-fractal, e.g. .00005.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox:  -400 -300 400 300
%%BeginSetup
%%EndSetup
```

12. 100 is sufficient
13. An allude to G.E. Forsythe's 'A professional starts where an amateur ends.'

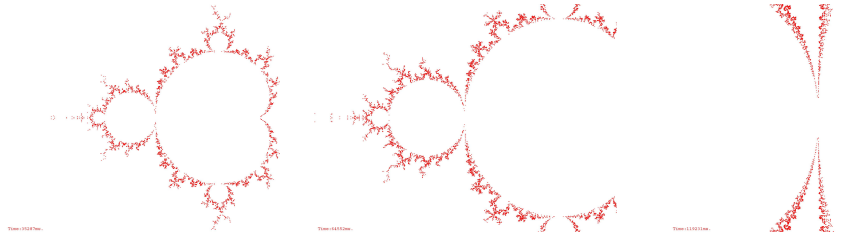
```
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%Endprolog
%
%   Program ---the script---
%
-.72 0  1.4 50 .0005 MANDISm
%for the right picture the invoke reads
/Courier 100 selectfont
begintime
-.16 1.03 .025 100 0.00005 MANDISm
-390 -295 moveto (Time:) show
  usertime begintime sub showobject
                  (ms.) show
showpage
%%EOF
```

-.72 0 1.4 50 .00005 MANDISm -.16 1.03 .025 150 .00005 MANDISm

The same mapping as in MANDET from the wired-in BoundingBox -400 -300 400 300 onto the domain of the M-fractal has been used.

Lauwerier(1990, p132–134) discusses the relationship of the quadratic Julia dynamical system with the Verhulst bifurcation diagram. For real c the a-axis reflects the Verhulst behaviour, which stretches out into the full cardioid and the full circle because of complex values of c.

A few stepwise magnifications have been given below, similar to those for MANDET.



-.74 0 1.5 100 .00005 MANDISm -.74 0 1 100 .00005 MANDISm -.74 0 .25 100 .00005 MANDISm

M-cardioid In Courant(1937, p267)[14] the cardioid is mentioned as a special case of the epicycloid, which results when a circle rotates around a circle. The equations for the cardioid, parametric in $\varphi \in [0, 2\pi]$, read

$$x = \cos(\varphi)/2 - \cos(2\varphi)/4$$

$$y = \sin(\varphi)/2 - \sin(2\varphi)/4$$

The cardioid has been draw, with the main circle centre at (-1,0) and radius .25, next to it; see accompanying picture. The cardioid is of the same size as the M-fractal cardioid.
In order to have an idea of the scale in the M-fractal picture the relevant numbers are shown underneath the drawing. The a- and b-axis have been drawn dashed.
If $|4a^2 + 8a + b^2| < 3.75$ then (a,b) lies inside the cardioid.
If $|a + ib + 1| < .25$ then (a,b) lies inside the circle.
Lauwerier(1995) contains a BASIC program for drawing a cardiod, CARDIO. My PostScript program for the accompanying M-Cardioid picture is straightforward and has been included in PSlib.eps.

Wim W. Wilhelm communicated his compact specification for drawing the cardioid

```
ParametricPlot[{Cos(fi)/2-Cos(2 fi)/4, Sin(fi)/2-Sin(2 fi)/4}, (fi, 0, 2 pi)]
```

I'm not sure whether Lauwerier would have loved these features. I do like the concise Math specifications, progress has been made since the 90s.

14. Courant, R(1937, sec.ed.): Differential and Integral Calculus.

Circle symmetric Julia fractals

In Lauwerier(1996) the research of Field and Golubitsky(1992) is mentioned with respect to circle symmetric fractals. They chose the 2D dynamical system, $z_{n+1} = F(z_n, \bar{z}_n)$ which must obey

$$F(z, \bar{z}) = \bar{c}F(cz, \bar{c}\bar{z})$$

in order that the resulting pseudo-Julia fractal will be circle symmetric, Lauwerier(1996, p122).

Use of JULIAS

The dynamical system used by Field c.s. in JULIAS for an m-fold rotation symmetric pseudo-Julia fractal reads, Lauwerier(1996, p129)

$$z_{n+1} = z_n(a + b/z_n^m + c)) \quad \text{with} \quad 1 = a + b/(1 + c).$$

The parameters are a, b, N the number of points, kmax, the maximum number of iterations per grid point.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -180 -140 180 140
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run%Inclusion of PSlib(rary)
%%EndProlog
%
%   Program, the script
%
1 .75 scale 3 0 105 0 360 arc%ellips
clip                          %clip rest of picture to ellips
.5 .3 200 100 JULIAS          %Circle Symmetric Julia fractal, scaled elliptically
showpage
%%EOF
```

Use of JULIASYMm

Another dynamical system used by Field c.s. in JULIASYMM for an m-fold rotation symmetric pseudo-Julia fractal reads, Lauwerier(1996, p124),

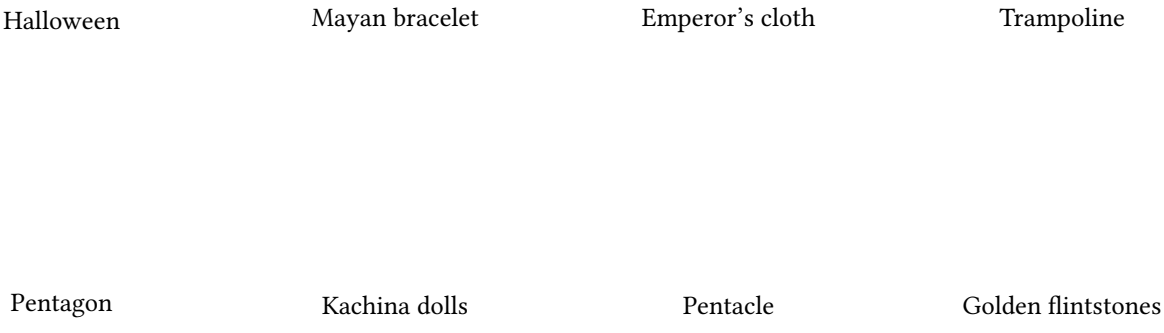
$$z_{n+1} = z_n(a + bz_n\bar{z}_n + c\,\mathrm{Re}(z_n^m)) + d\bar{z}_n^{m-1} \quad \text{with} \quad 1 = a + b/(1 + c).$$

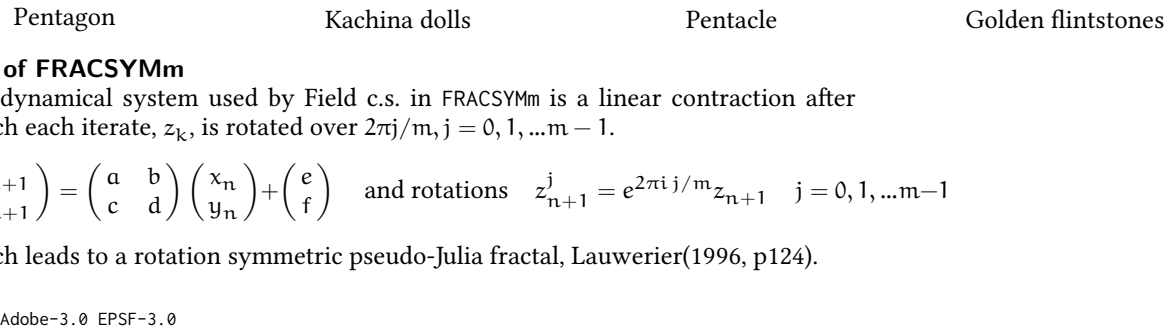
The parameters are a, b, N the number of points, kmax, the maximum number of iterations per grid point,

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 125 40 515 440
%%BeginSetup
%%EndSetup
%%DocumentFonts: Courier
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
%
%   Program the script
%
-2.08 1 -.1 .167 150 7 5000 JULIASYMm%Mayan bracelet
showpage
%%EOF
```

Example values for the parameters are summarized in the table with the corresponding figures underneath.

Nickname	a	b	c	d	h	m	col	kmax
Halloween	-2.7	5	1.5	1	250	6	9	7000
Mayan bracelet	-2.08	1	-0.1	0.167	150	7	8	7000
Emperor's cloth	-1.806	0.1806	0	1	250	5	5	7000
Trampoline	1.56	-1	0.1	-0.82	150	3	6	7000
Pentagon	2.6	-2	0	-0.5	150	5	5	10000
Kachina dolls	2.409	-2.5	0	0.9	200	23	8	7000
Pentacle	-2.32	2.32	0	0.75	200	5	8	7000
Golden flintstones	2.5	-2.5	0	0.9	175	3	6	7000





Use of FRACSYMm

The dynamical system used by Field c.s. in FRACSYMm is a linear contraction after which each iterate, z_k, is rotated over $2\pi j/m, j = 0, 1, ...m-1$.

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \quad \text{and rotations} \quad z^j_{n+1} = e^{2\pi i\, j/m} z_{n+1} \quad j = 0, 1, ...m-1$$

which leads to a rotation symmetric pseudo-Julia fractal, Lauwerier(1996, p124).

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -250 -250 250 250
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
%
%   Program the script
%
%a   b    c  d  e  f sc m  kmax  def-name
-.1 .35  .2 .5 .5 .4 6 175 4000 FRACSYMm
showpage
%%EOF
```

The parameters are the similarity transformation constants a, b, c, d, e, f, the scaling factor, the rotation symmetry, and kmax, the maximum number of iterations per grid point,

The following pictures have been created by FRACSYMm with the parameters given underneath.

-.4 .75 .2 -.3 0 .4 200 55 2000 -.15 .75 .2 -.3 .075 .4 250 50 2000 -.25 -.3 .14 -.26 .5 .5 175 12 3000 .45 -.1 -.31 .45 .1 .2 500 11 3000

.4 -.1 -.35 .4 .01 .2 500 9 3000 .45 -.1 .3 -.4 .15 .1 500 8 3000 -.1 .35 .2 .5 .5 .4 150 6 4000 .5 0 0 .5 .5 0 200 5 4000

Use of some packages from the WWW

"It is my philosophy that not only we should show in our User Group publications what TEX&Co can do for us, but also mention programs, or tools, which perform the same task, in order to choose the best tool for the purpose."

Use of Stuif's Julia previewer Stuif's WWW contains a button for the Julia Mandelbrot applet.

Moving with the cursor over the M-fractal, left, and clicking the mouse yields the corresponding Julia fractal. The pictures at right show J(.325, .417), options: details and the fractal.

Use of Winfract It opens with a window displaying the M-fractal with a menu bar.

Within the context of this note the Mandelbrot/Julia toggling is interesting. It allows to go from a point in the M-fractal to show the corresponding Julia fractal.

Pictures are stored in `.par` format, whatever that means. Pictures can also be saved on the clipboard and opened in Photoshop te be stored in a format of choice.

The help file contains the topics

- ☐ Whats New?
- ☐ File Menu
- ☐ Fractals Menu
- ☐ View Menu
- ☐ Colors Menu
- ☐ Help Menu
- ☐ Fractal Formula Selection
- ☐ Zooming in on an Image
- ☐ Mandelbrot/Julia Toggling
- ☐ Color-Cycling
- ☐ Fractint-Style Help and Prompts
- ☐ Coordinate Box options
- ☐ Limitations in Winfract
- ☐ Distribution Policy, Contacting the Authors, The Book
- ☐ A list of Winfract and Fractint Authors

Use of the XaoS real-time fractal zoomer XaoS has been made by Jan Hubicka&Thomas Marsch[15] and has been put in the public domain under the GNU license.

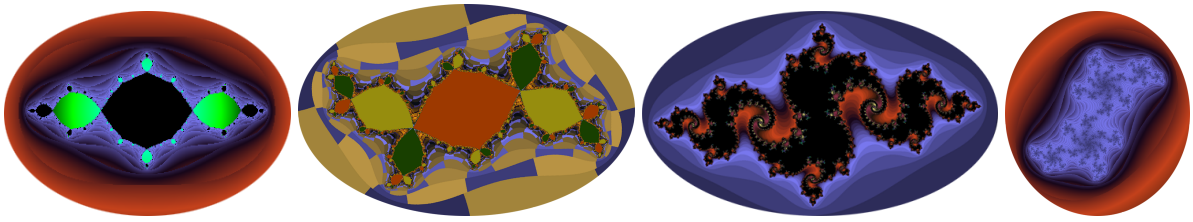
A lot of prefab fractals are available via the menus to experiment with or just to watch in amazement. A picture is saved as `.png` file. There is a tutorial and a help file. Jan Hubicka's movie 'An introduction to Fractals' is fascinating, and gives more than a book ever can provide. I did not find out yet how to circularly crop a picture. Manually altering the window size crops the picture. Zooming goes by the left and right mouse button. In order to obtain the circle-inversion of the Mandelbrot fractal (right picture) one has to select $1/\mu$ through the hierarchy of menus `Fractal` $\rightarrow$ `Plane` $\rightarrow$ $1/\mu$. The inverse Mandelbrot has been coloured inside via `incolouring mode`.[16]

Sierpinski Mandelbrot Mandelbrot detail Mandelbrot4 Invers Mandelbrot

For the variation of colour, second, fourth and fifth picture, there are various options under `incolouring` mode, i.e. inside the fractal, and `outcolouring` mode, i.e. outside the fractal. For example `Fractal` $\rightarrow$ `Incouloring Mode` $\rightarrow$ `real/imag`. There is also a pseudo 3D option. In short, a lot of options to experiment with and to obtain coloured fractals, which are not so easy to realize via PostScript, to put it mildly.

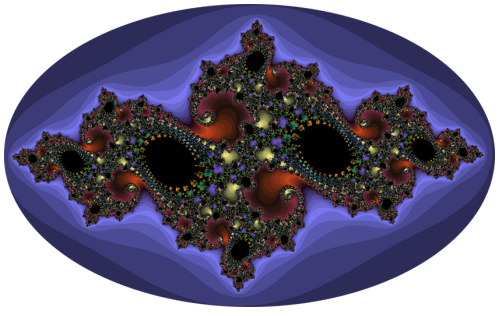
Julia fractals via XaoS Within the context of this note, I was interested to specify my own Julia fractals. For fractals to be specified by the user there is the sub-menu `User formula`, see accompanying picture, where the Julia fractal `J(-1, 0)` can be specified by `z^2-1`, and in general the Julia fractal `J(a, b)` by `z^2+{a;b}`, with a the real part and b the imaginary part of $c \in \mathbb{C}$ in $z_{n+1} = z_n^2 + c$.

15. Since 1996–2008, Version 3.5
16. An inverse Mandelbrot picture in blue and yellow is used on the cover of Lauwerier(1990).



J(.8, 0) San Marco J(-.12, .74) Douady J(-.8, .15) Dragon J(.11, .66) Dendrit

Without previous experience with XaoS it took me roughly 5 hours to produce this section, where the pictures have been cropped by PhotoShop.



J(-.7454, .1103)

Use of the Fractalus package Fractalus is a Julia fractal generator for Windows 32- and 64bits, made by Kari Korkeila and available[17] in the public domain under the GNU license.

Features

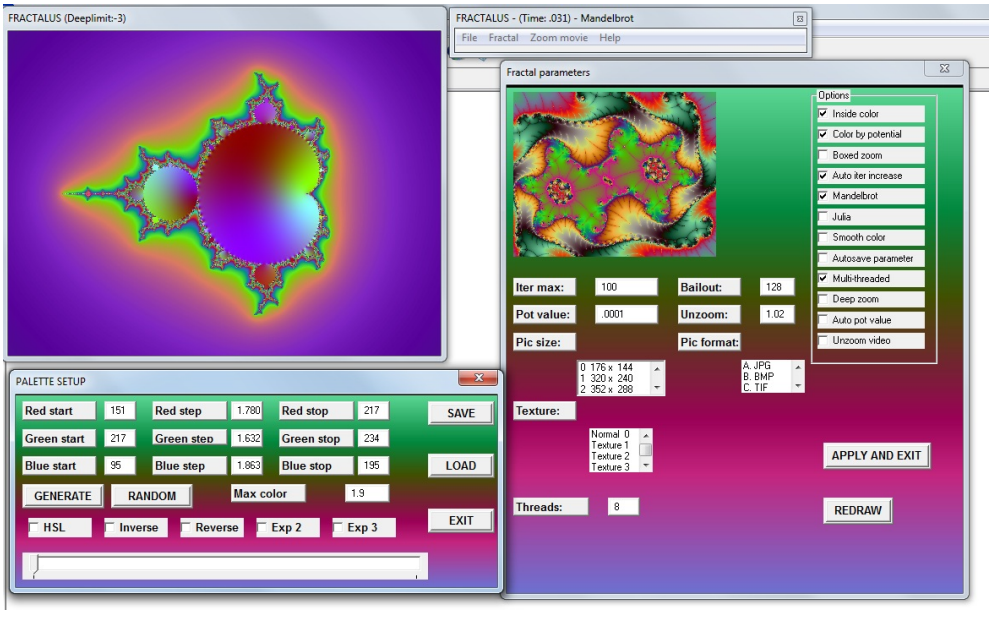
- ☐ fast and small optimized executable (only 760 kb)
- ☐ truecolor rendering
- ☐ saves and loads parameter-files (deep zoom parameters can be loaded with image without re-calculating)
- ☐ saves and loads palettes
- ☐ saves pictures as .png, .tiff, .bmp and .jpg; selectable image size from 176x144 to 8192x6140
- ☐ largest image size may not work on all systems - default is .jpg
- ☐ Mandelbrot and Julia fractal types currently supported
- ☐ 28 different drawing styles or textures
- ☐ palette manipulation
- ☐ browse fractal parameters with image preview
- ☐ easy zooming and unzooming with right and left mouse-button
- ☐ make zoom movies from fractal parameter files (max resolution 1920x1080) deep zoom also supported
- ☐ make zoom movies from .jpg image sequences
- ☐ real-time zoom & unzoom movie (only 80 bit accuracy supported and quite fast machine required)
- ☐ unlimited deep zooming for Mandelbrot and Julia fractals (more than 80 bits used - tested with over 800bit accuracy or over 200 decimals)
- ☐ multi-thread support with multi-core processors
- ☐ portable application (download zip file)
- ☐ graphical Julia finder

17. Since 2011. Current Version 5.751.

The last feature is very interesting within the context of this note: visualization of Julia fractals.

Menus There are 4 menus shown in the picture below. The most relevant for direct use are

- FRACTALUS - Mandelbrot, main menu, with roll-down selection menus.
- FRACTALUS (Deeplimit:-3): displays the fractals, from which the picture can be saved. The parameter value c can requested via the Fractal roll-down menu by the selection of the Location-item.



Quick-start guide from the WWW

- Zoom with left mouse click. Unzoom with right mouse click
- Select palette from menu - click random button until you find good palette
- Save picture and/or as parameter-file.
- For deep zooms some tips
  Use small image size with deep zooms and use boxed zoom.
  Rendering deep zooms can be quite slow.
  Turn on auto-save option when rendering deep zooms.

There is no user guide. Apparently, one just has to play with it.

Generation of Jullia fractals Select in the Fractal roll-down menu the item Julia finder. For the position c of the cursor in the main Mandelbrot window, the Julia fractal $z^2 + c$ will be shown real-time, on-the-fly, in a second window. If you right-mouse click, the Julia fractal will appear in the main window after which it can be saved as .jpg, .bmp, .tif or .png at various sizes, to be selected in the fractal parameters window. In the accompanying figure the white cross denotes the position of the cursor and the small window shows the corresponding Julia fractal. For those who just like the beauty of fractals, like my sister, there is an extensive Gallery of prefab fractals.

Without previous knowledge of Fractalus it took me roughly two days to produce this section. Definitely a tool to have at hand.

Annotated References

□ Introductory surveys:
http://en.wikipedia.org/wiki/Dragon_curve
http://www.stuif.com/fractals/index.html
http://en.wikipedia.org/wiki/Julia_set
http://en.wikipedia.org/wiki/Mandelbrot_set
http://en.wikipedia.org/wiki/List_of_fractals_by_Hausdorff_dimension.

□ Adobe Red, Green and Blue Books. The musts for PostScript programmers.

□ Biography of H.A. Lauwerier: http://bwnw.cwi-incubator.nl/cgi-bin/uncgi/alf.

□ Deubert, J: Acumen Journal. http://www.acumentraining.com/acumenjournal.html. Highly educative, with respect to PostScript, PDF and XPS. (From the november2011 issue: PostScript Tech - Transparency in PostScript Using pdfmark PostScript implements a strictly opaque imaging model; objects painted on the page completely obscure anything on the page beneath them. That is, unless you are handing your PostScript file to Distiller; in that case your PostScript code can draw translucent objects on the final PDF page using the Distiller-only pdfmark operator.)

□ Fractal Foundation. http://www.fractalfoundation.org.

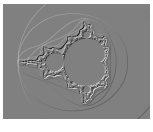
□ Gleisk, J(1987): CHAOS — making a new science. Penguin.
(An introduction to and survey of the world of non-linearity, strange attractors and fractals. The accompanying picture has the following legend: ...The complex boundaries of Newton's method. The attracting pull of four points — in the four dark holes — creates 'basins of attraction,' each of different color, with a complicated fractal boundary. The image represents the way Newton's method for solving equations leads from different starting points to one of four possible solutions (In this case the equation is $z^4 - 1 = 0$) ...)

□ Goossens, M(2007, sec.ed.) c.s.: LaTeX Graphics Companion. ISBN 978 0 321 50892 8.

□ Heck, A(2005): Learning MetaPost by Doing. MAPS 32. (A tutorial with nice practical examples.)

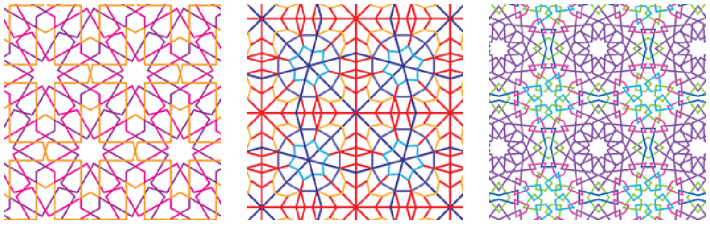
□ Hobby, J.D(1995): Drawing Graphs with MetaPost. CSTR Report 16.

□ Hubicka J, T. Marsch(1996–2008) XaoS3.5 — a real-time fractal zoomer.
(Along with the package comes a fascinating and well-done movie 'An introduction to Fractals.')



□ Jackowski, B, P. Strelczyk, P. Pianowski(1995-2008): PSView5.12.
bop@bop.com.pl. (Extremely fast previewer for .eps and .pdf.
Allows PSlib(rary) inclusion via the run command.
Error messages appear in the pop-up GhostScript window.)

□ Kari Korkeila(2011): Fractulus — a Fractal generator for Windows. `http:/personal.inet.fi/koti/fractulus`.
□ Knuth, D.E, T. Larrabee, P.M. Roberts(1989): Mathematical Writing. MAA notes 14. The Mathematical Association of America.
□ Knuth, D.E(1990, 10th printing): The TEXbook. Addison-Wesley. ISBN 0-201-13447-0. (A must for plain TEXies. I never read a manual so many times. Well ...Adobe's RGB-books come close.)
□ Kroonenberg, S(2007): Epspdf, easy conversion between PostScript and PDF. EuroBachoTEX.
□ Lauwerier, H.A(1987): Analyse met de Microcomputer. Epsilon 7.
□ Lauwerier, H.A(1987): FRACTALS — meetkundige figuren in eindeloze herhaling. Aramith. (Contains BASIC programs. Lauwerier, H.A (1991): Fractals: Endlessly Repeated Geometrical Figures. Translated by Sophia Gill-Hoffstadt, Princeton University Press, Princeton NJ. ISBN 0-691-08551-X, cloth. ISBN 0-691-02445-6 paperback. "This book has been written for a wide audience ... " Includes sample BASIC programs in an appendix. With respect to Julia fractals it contains only the programs JULIAB(acktracking) and MANDEL, for the black-and-white banded Mandelbrot fractal. Intended for instructors, (high-school) students,) and the educated layman.)
□ Lauwerier, H.A(1988): Meetkunde met de Microcomputer. Epsilon 8.
□ Lauwerier, H.A(1989): Oneindigheid — een onbereikbaar ideaal. Aramith. ISBN 90 6834 055 7.
(With respect to Julia fractals it contains the BASIC programs `JULIAS`, `JULIAP` and `MANDELP` the latter is more efficient than the one in Lauwerier(1987). This booklet filled up holes in my knowledge of number systems. The regular continued fraction expansion for ϕ, the golden ratio, with only 1-s, I'll never forget. Audience: Instructors, (high-school) students, and the educated layman.)
□ Lauwerier, H.A(1990): Een wereld van FRACTALS. Aramith. ISBN 90 6834 076 X. (A sequel and updated version of Lauwerier(1987). Intended for instructors, (high-school) students, and the educated layman.)
□ Lauwerier, H.A(1992): Computer Simulaties — De wereld als model. Aramith. ISBN 90 6834 106 5. (The last chapter is called Orde en Chaos, and applies to this paper.)
□ Lauwerier, H.A(1994): Spelen met Graphics and Fractals. Academic Service. ISBN 90 395 0092 4. (An inspiring book with Math at the high-school level for a wide audience. The BASIC programs I consider outdated for direct use. Intended for instructors, (high-school) students, and the educated layman.)
□ Lauwerier, H.A(1995): Symmetrie, Kunst en Computers. Aramith. (The permutation group P_n is used to classify various symmetries. The possible tilings of the plane by regular polygons are discussed and their mathematical classification is explained. Tilings of Duat and Penrose are included. The Platonic and Archimedic polyhedra are treated. The symmetries in the (pseudo) Julia and Mandelbrot fractal are mentioned. Intended for instructors, (high-school) students, and the educated layman. The BASIC codes are available from the file-server of the THE.)



□ Lauwerier, H.A(1996): Chaos met de Computer. Epsilon Uitgaven, Utrecht. ISBN 90 5041 043 X4. (Treats the restricted growth model in detail and elaborates on extensions into 2D, among others. Intended for instructors, (high-school) students, and the educated layman.)

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox:  -440 -90 430 345
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%EndProlog
%
% Program
%
lorenzclipped%Lorentz attractor
showpage
%%EOF
```

□ Minsky, M(1969): Form and Content in Computer Science. Turing Award lecture. In: ACM Turing Award Lectures. The first twenty years 1966-1985. ACM Press. (With TEX and similar quality tools for typesetting, I get the impression that pre-press publications within the TEX-community are out of balance: too much attention is paid to form.)
□ Peitgen, H.O, H.Jürgens, D. Saupe(2004 sec.ed.): Chaos and Fractals. New frontiers of Science. (Images of the fourteen chapters of this book cover the central ideas and concepts of chaos and fractals as well as many related topics including: the Mandelbrot set, Julia sets, cellular automata, L-systems, percolation and strange attractors. This new edition has been thoroughly revised throughout. The appendices of the original edition were taken out since more recent publications cover this material in more depth.[18] Instead of the focused computer programs in BASIC, the authors provide 10 interactive JAVA-applets for this second edition via `http://www.cevis.uni-bremen.de/fractals`. An encyclopaedic work. Multiple Reduction Copying Machines, as model for feedback systems, are associated with IFSystems, The Barnsley fern has been elaborated upon and the chaos game is explained. Audience: No mathematical sophistication is required, so a broad audience is aimed at. It portrays the new fields: Chaos and fractals, in an authentic manner. The first edition has been typeset by (La)TEX&Co, the second edition as well, I presume.)
□ Stone Soup Group(1990–2008): `Fractint` and `Winfract` fractal packages. Released in the Public Domain under GNU license.
□ Swanson, E(1986, rev.ed.): Mathematics into Type. American Mathematical Society.
□ Szabó, P(2009): PDF output size of TEX documents. Proceedings EuroTEX2009/ConTEXt, p57–74. (Various tools have been compared for the purpose.)
□ Van der Laan, C.G(1995): Publishing with TEX. Public Domain. (See TEX archives. `BLUe.tex` comes with `pic.dat` the database of pictures in TEX-alone. No Julia fractals, but some strange attractors have been done by TEX-alone. The advantage of TEX alone approach is portability in place and time. All chapters

18. Curious is that the data compression appendix has been taken out while the source still contains blind alleys to the absent appendix; the source has not been adapted, nor are wavelets mentioned, which are used in `.png` and `.jpg` compression. Although the BASIC codes have been taken out, the text is still BASIC biased as can be whitnessed from the attention paid to Turtle Graphics; no recursion which BASIC lacks. The material on the history of the calculation of the digits of π is nice and informative, but a bit out of context.

still compile under pdfTEX without adaptation. The inelegance of the font aspects is inherited from the bitmap-fonts plain TEX-engine. A successor of this work should be based on an OTF-TEX-engine.)
□ Van der Laan, C.G(unpublished, BachoTEX workshop): TEXing Paradigms. (A plea is made for standardized macro writing in TEX to enhance readability and correctness. Topics: Plain's `items` extended, Headache (about headings), Two-part macros. Parametrization I — Options, The wind and hallfwinds (macros for turtle graphics in TEX), It's all in the game — Dialogue with TEX, Loops, Searching, Sorting, Just a little bit of PostScript, FIFO and LIFO sing the BLUes, Syntactic Sugar.)
□ Van der Laan, C.G(2009): TEX Education — an overlooked approach. EuroTEX2009-3rdConTEXt proceedings. (Launched my `PSlib.eps` library.)
□ Van der Laan, C.G(2010): Circle Inversions —with a serious undertone—. MAPS 42. (Contains the solution of Apollonius problem as a PS `def`. `PSlib.eps` is introduced.)
□ Van der Laan, C.G(2011): Gabo's Torsion. MAPS 42. (Contains also a summary of the PostScript language and its developments.)
□ Van der Laan, C.G(2012): Pythagoras Trees in PostScript — Fractal Geometry 0. EuroBachoTEX2012-proceedings (MAPS 44). (Submitted Informatsionnie Texnologii i Matematicheskoe Modelirovanie.)
□ Van der Laan, C.G(2012): Classical Math Fractals in PostScript — Fractal Geometry I. EuroBachoTEX2012-proceedings (MAPS 44). (Submitted Informatsionnie Texnologii i Matematicheskoe Modelirovanie.)
□ Veith, U(2009): Experiences typesetting mathematical physics. Proceedings EuroTEX2009/ConTEXt, p31–43. (Practical examples where we need to adjust TEX's automatic typesetting.)

I don't own any of the following Classical Fractal books but pick up Barnsley(1988, 2006) and Mandelbrot(1982) from the library:[19]

□ Barnsley, M.F(1988): Fractals Everywhere. Academic Service.

(Famous from this book is the IFS for a fern. IFS-s, equivalent dynamical systems as well as the associated fractals are treated within a theoretical framework for Fractal Geometry. The accompanying illustration is used in ch. 3 to illuminate the idea of a contractive transformation on a compact metric space.)

Chapter 7 Julia sets. Chapter 8 Parameter Spaces and Mandelbrot sets. BASIC programs are included: 3.8.1 Example of Deterministic Algorithm; 3.8.2 Random Iteration Algorithm; 6.2.1 Fractal Interpolation; 7.1.1 Example of the Escape Time Algorithm. (No efficiency short-cuts such as use of symmetry in the codes. I could not spot the data reduction process of the Andes Indian girl picture.)
□ Barnsley, M.F(2006): SuperFractals. Patterns of Nature. Cambridge University Press. 464p. (.....Superfractals would be a superb addition to the bookshelves of any scientists who use fractal analysis techniques in their research, be they physicist, biologist or economist. The author concludes by promising that the introduction of superfractals will revolutionize the way mathematics, physics, biology and art are combined, to produce a unified description of the complex world in which we live. After reading this book, I have no doubt that he is correct. From NATURE Vol 445 18 January 2007.)
□ Falconer, K(1997): Techniques in Fractal Geometry. Wiley. ISBN 978-0-471-95724-9. (Peitgen c.s. mention that this book contains an adequate technicaL discussion of the fractal dimension.)

19. The problem with the classics like Barnsley(1988, 2006) is that they contain too much in too advanced Math. It is easier for me to read Lauwerier, who understands the matter and with his magical intuition summarizes the relevant issues in not too sophisticated Math. But ... the pictures in those classics are usually breathtaking.

□ Falconer, K.J(2003, sec.ed.): Fractal Geometry. Mathematical Foundations and Applications. Digital and ordinary book.
□ Field, M, M.Golubitsky(1992): Symmetry in Chaos. Oxford University Press.
□ Field, M, M.Golubitsky(2009): A search for Pattern in Mathematics, Art and Nature, (Symmetry suggests order and regularity whilst chaos suggests disorder and randomness. Symmetry in Chaos is an exploration of how combining the seemingly contradictory symmetry and chaos can lead to the construction of striking and beautiful images. This book is an engaging look at the interplay of art and mathematics, and between symmetry and chaos. The underlying mathematics involved in the generation of the images is described. This second edition has been updated to include the Faraday experiment, a classical experiment from fluid dynamics which illustrates that increasing the vibration amplitude of a container of liquid causes the liquid to form surface waves, instead of moving as a solid body. This second edition also includes updated methods for numerically determining the symmetry of higher dimensional analogues of the images. As well as this, it contains new and improved quality images.)
□ Golubitsky, M(1997): Introduction to nonlinear Dynamical Systems and Chaos. Springer-Verlag.
□ Fischer, Y (): Fractal Image Compression Theory and Applications.
□ Mandelbrot, B(1982); The Fractal structure of Nature. ISBN 07 167 11869.
(This essay of erudition, originality and insight, is about the fundamentals of Math: Euclidean geometry and the dimension concept are widened up into fractal geometry and fractal dimension. A plea is made for fractals to describe natural phenomena: clouds, waves, landscapes, length of coasts with the ill-posed question of length answered by: each coastline has a fractal dimension, The 'montruous' curves of 19^{th} century Math find a honourable niche in fractal geometry, with a fractal dimension added.)

□ Peitgen, H.O, P.H. Richter(1986): The Beauty of Fractals. Images of complex dynamical systems. Springer-Verlag. (Frontiers of Chaos. Verhulst Dynamics. Julia Sets and their Computergraphical generation. Sullivan's Classification and Critical Points. The Mandelbrot Set. External Angles and the Hubbard Trees. Newton's method for Complex Polynomials: Cayley's Problem. Newton's method for Real Equations. A discrete Volterra-Lotka System. Magnetism and Complex boundaries. Yang-Lee Zeros. Invited contributions:
Fractals and the Rebirth of Iteration Theory by B. Mandelbrot;
Julia Sets and the Mandelbrot Set by A. Douady;
Freedom, Science, and Aesthetics by G. Eilinger;
Refractions on Science into Art by H.W. Franke.
The book ends with DO IT YOURSELF and DOCUMENTATION.)

Conclusions

It was pleasure, educative and inspiring to read Lauwerier's booklets. Some of his algorithms have found a wider audience by conversion of his BASIC codes into PostScript `defs`, hopefully.

I have no experience in running BASIC programs nor do I know how to include the resulting pictures elegantly in my publications. The EPSF results of PostScript programs can easily and time-proven, be included in my pdf(La)TeX[20], Word, ... documents, or in other PostScript programs.

20. After conversion, alas. `\psfig` functionality has been lost. Happily, ConTEXt and LuaTEX allow EPSF inclusion.

Lauwerier(1990) mentions a lot of applications of fractals in the world around us, to which I did not pay attention in this note, but demonstrates the relevance of studying fractals.

A half year ago, I didn't know how to draw fractals on my PC. Now I have a suite of PostScript `defs` available in my `PSlib.eps` library, and I'm aware of several fractal packages, which also allow drawing user specified Julia fractals, with zoom, colour and transformation facilities.

The relation between the M-fractal and the various Julia fractals is

- □ The M-fractal is a map of Julia fractals. To each point `(a, b)` of the map belongs a Julia fractal `J(a,b)`.
- □ The M-fractal is a bifurcation diagram of Julia fractals.

In the past 15 years nice fractal software has been released in the public domain.

I get once more the impression that I should learn JAVA (Stuif, Peitgen) in order to create animated Fractals, in pursuit of the animated Cabri software for hyperbolic geometry.

Lauwerier, 25 years ago, was more than right in his vision that the PC would become a household tool and could be used for playing with fractals and for experimenting the world around us, as witnessed in Lauwerier(1992).

'Het Wiskunde boek' states that fractals have renewed and raised interest in Mathematics.

Before publishing consult the Wikipedia on aspects of the subject as well as Wolfram's knowledge base `http://www.wolframalpha.com`.

It would be nice to have fractal contours for the line fractals and colouring by gradients.

It is curious how the adherence to structured programming — preferring the MetaPost preprocessor instead of PostScript, casu quo preferring LaTeX instead of minimal plain TeX ...— is overdone in the TeX-community. In the case of neglecting PostScript the champagne is thrown away with the cork,[21] i.e. the powerful transformation of User Space of PostScript is passed by, so is PostScript's more accurate arithmetic, which is of higher accuracy than the arithmetic in MetaPost.[22] The often praised virtue of formal, symbolic solutions of equations in MetaPost, neglects the numerically more stable pivoting strategies.

However ... what about TeX&Java or LaTeX&Asymptote&TeXnicCenter?

Happily, I was not aware of `Fractalus` when I started working on this note, because otherwise I might not have converted Lauwerier's BASIC codes into PostScript.

I don't expect the double precision BASIC codes to translate easily in MetaPost, because of the limited accuracy of the current MetaPost.

Working on this note has widened my horizon of creating pictures by PostScript; I also became aware of the usefulness of the `PSView` previewer, for previewing `.eps` as well as `.pdf`.

21. I drink Moldavian champagne and reuse the corks ànd bottles for my home-made wine and cider.
22. PostScript adopted the accuracy and arithmetic of the underlying computer architecture. The PostScript LRM 3 table B.1 states as accuracy for reals: 8 significant decimal digits. I verified it experimentally and found 10^{-7}.

TEX mark-up For the symbols of the number systems $\mathbb{I}$, $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, which curiously are not provided for in plain TEX, I used the AMS (blackboard) font `msbm10`. I look out for the day that OpenTypeFonts will be universally available, to start with availability in TEX and PostScript.

In-line verbatim is marked up by starting and ending the verbatim text by a vertical bar, `|`,borrowed from `BLUe.tex`, which on its turn learned from the mark-up of the TEXbook source. However, the use of the `|`-symbol is lost for the mark-up of the absolute value; happily, there is still the more verbose alias `\vert`.

I have used in the appendices `vbox`-s next to each other for parallel listings of the program texts — BASIC and its converted PostScript `def` — which inhibits proper page breaks, but doesn't harm in the appendices. I don't know how to provide macros for local elegantly marked up multi-column texts in `\vbox`-s, which allow for proper page breaks. How to cope with page-breaks and for example pictures is a well-known problem. Usually the picture is floated on the page or placed on the next page. TEX provides insertion commands to support floating, e.g. the 2-part macro `\topinsert` and `\endinsert`, TEXbook p115–116. Ill-placement of figures is taken for granted. Another approach is to bundle the (colour) pictures in a special quire, as applied by Lauwerier.

Inserting (large) `.pdf` pictures lead to problems with the Acrobat viewer. As far as I understand it the `.pdf` figure has to be processed by Acrobat before insertion. Beforehand converting the pictures into reasonable-sized `.jpg` solved the problems, slowness, with viewing by Acrobat. Processing by TEXworks became faster as well.

The table where as function of the Malthus parameter a the various kinds of bifurcations are summarized is typeset by straightforward use of `\halign`. The tables of pictures are not typeset by `\halign` nor by the tabbing mechanism, because I found it more flexible just to use `box`-s. For typesetting of the tables with vertically aligned decimal dots, I had to refresh my knowledge of `\halign`. For the typesetting of vertically aligned decimal dots I did not follow the approach of the TEXbook p241, where each leading non-significant zero is hidden by the `?`-mark-up. I used 2 columns: one for the integer part and the other for the fractional part, which is a bit of error-prone mark-up, however, ... once used to it ... . As a consequence the text in the heading over these 2 columns is centred by using `\span` instead of `&`. The separation of the heading from the table date I did by `\noalign{<verticalmaterial>}`, the TEXbook, p237, or using a `\strut`. The `\halign` template has `\sevenstruts` for more pleasing lines in the data part. The font used for the tables is 7pt.

The picture of the grid is not created by use of TEX's `\leaders`, but created in simple straight minimal PostScript, because then the picture can be more easily reused. The labels form an integral part of the picture: an all-in-one self-contained picture. The picture of the Cardioid is created in PostScript, where the arrow heads of the coordinate axes are done by `arrow` from Adobe's Blue Book, p138–139.

Because `\eqalign` allows only '2 columns' I modified `\eqalign` for `BLUe.tex` into a variant which allows more 'columns,' and used that variant as well.

The simple bar-chart-like graphs for the convergence behaviour of the restricted growth model have been created in straight minimal PostScript as well; no MetaPost Graph macros needed.

The $\stackrel{\mathrm{L}}{=}$ composed relational operator is marked up by `\mathrel {\mathop =^{\rm L}}` and not by `$\buildrel\rm L \over=$`, TEXbook p437, which is OK when no surrounding Math-relational-operator space is needed.

I don't like the ill-placing of floated pictures. My inserted pictures suffer from the same inconvenience as in Word: changing the text might disturb the layout, such that the pictures will become ill-placed. I use 2 methods for placing pictures: the first one is a `\line` with 2 `\vbox`-s, one for the text and the other for the picture. The other creates white-space at the right margin via the use of `\hangindent` and `\hangafter`, TEXbook p102, which adheres more to minimal mark-up and is more general than

my earlier adapting the value of `\hsize` within a group ended by `\par`. The pictures are inserted in the blank space via the TEXnique mentioned in TEXbook Appendix D p389, by my macros `\insertpdf`, casu quo `\insertjpg`. The mark-up is highly similar to the mark-up of the dangerous bends in the TEXbook. These dangerous bends are just marked up by `\danger`, which takes care of providing the open space by using `\hangindent` and `\hangafter`, and places at the open space the picture, which is stored as character in the `manual` font (See theTEXbook Appendix B for the macro `\danger`).

The non-automatically font scaling within context of TEX is a nuisance. I was caught by that Math symbols in the abstract and footnotes don't comply with 7pt. I had to switch explicitly to `\seveni`. Maybe, I should write a Math scaling, to be applied when needed.

For the circle symmetric fractals Acrobat 7 performed slow, very slow. `PSView` previewed just by a fillip, but ... without yielding a `.pdf`-file. `MANDET` gave an overflow message in Acrobat, while `PSView` still showed the result.

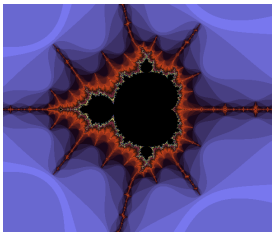
TEXworks2010 fell silent when working on this note, crashed an unacceptable number of times with the message: `Assertion failed.`?!? Installation of TEXworks 2011 solved the problem.

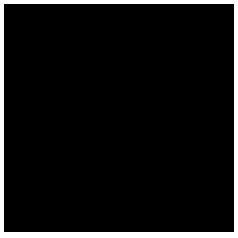
Pictures for the slides are in `.png` format with transparent background; I could not obtain transparency via `.jpg` format.

For the slides with blue-coloured Math text I had to insert when using pdfTEX `\pdfliteral{0 0 0 0 k}` for blue-coloured text as well as `\pdfliteral{0 0 0 0 K}` for blue-coloured lines, e.g. to get for example the horizontal line in the enlarged sqrt-symbol or the horizontal bar in `\over` blue-coloured. Weird, very weird.[23] 21st-century software unworthy.

Finally, as an aside a plea for the developers of the successor of TEX

> "Successors of TEX should be freed from the wired-in discrete fonts, and embrace scalable, context-aware fonts."





PostScript Working on this note has deepened my knowledge of PostScript, though there is still much to learn. For example, I don't know as yet how to use PostScript's `pathbbox`, which prompts the actual BB, such that an exact cropped picture can be obtained, in a 1-pass job.

It is a pity that pdfTEX does not allow for pictures in PostScript.[24]

Advanced use of colour-gradients is on my mind, maybe some time, some day ...

My PostScript programming reflects the procedural style of programming of the 60s of the last century; 21st Century programming wraps procedural codes into a package structure, with access by interactive menus, abstracting from the various computer operating systems.

23. In 2009 I already experienced this inconvenience with commutative diagrams.
24. 15 Years ago I used psfig to include my PS-pictures in TEX documents. It is a pity that pdfTEX did not build on this tradition. Happily, LuaTEX and ConTEXt allow .eps inclusion.

Acknowledgements

Thank you Adobe for your maintained, adapted to LanguageLevel 3 since 1997, good old, industrial standard PostScript and Acrobat Pro (actually DISTILLER) to view it, Don Knuth for your stable plain TEX, Jonathan Kew for the TEXworks IDE, Hàn Thế Thành for pdf(La)TEX, Hans Lauwerier for your nice, educational booklets with so many inspiring examples and clear Math exposé.

Thank you Nice Temmme for the reference to the AMS notices of 2012.

Thank you Jos Winnink for proofing a near-finished version and for your as usual valuable comments, such as the mark-up of tables in smaller corps, and providing for the first example of use a variant with the library def, and what is needed more, included. Wim W. Wilhelm for your comments and for your example of the M-fractal in Asymptote, your modern vector-graphics tool as successor of MetaPost. Wim's parametric-plots tool looks handy to me. Henk Jansen for detailed proofing and his suggestion for the word quire, as well as the phrase 'throw away the champagne with the cork.' My sister, Martha van der Laan, for drawing my attention to XaoS, which she uses from an artistic viewpoint. MAPS editors for improving my use of English and last but not least Taco Hoekwater for procrusting my plain TEX note into MAPS format.

IDE My PC runs 32 bits Vista, with Intel Quad CPU Q8300 2.5GHz assisted by 8GB RAM. I visualize PostScript with PSView and convert into .pdf via Acrobat Pro 7.[25] My PostScript editor is just Windows 'kladblok (notepad).' I use the EPSF-feature to crop pictures to their BoundingBox, ready for inclusion in documents. For document production I use TEXworks IDE with the plain TEX engine, pdfTEX, with as few as possible structuring macros taken from BLUe.tex — adhering minimal TEX mark-up. I use the Terminal font in the edit window with the pleasing effect that comments remain vertically aligned in the .pdf window.

For checking the spelling I use the public domain en_GB dictionary and hyphenation patterns en_GB.aff in TEXworks.[26]

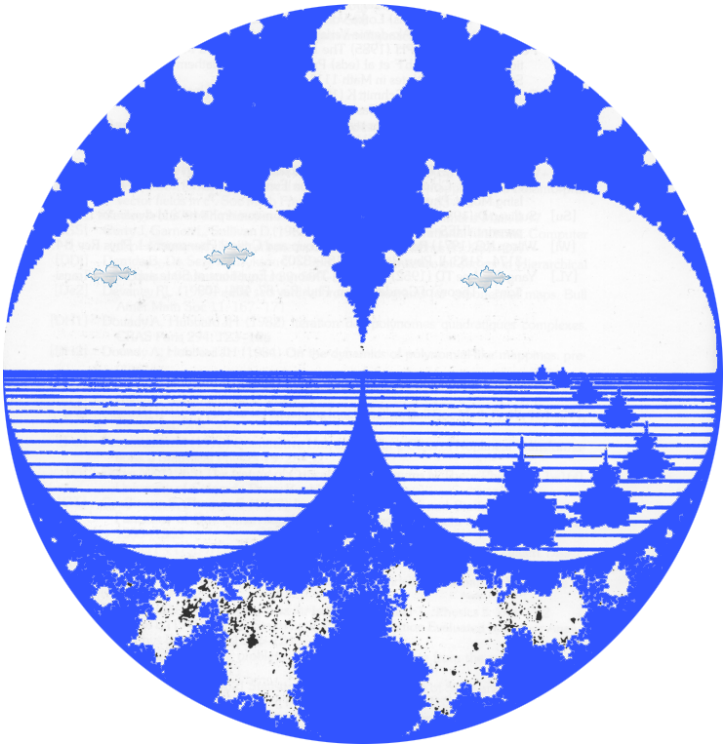
Prior to sending my PDF's by email the files are optimized towards size by Acrobat Pro.

The bad news with respect to .eps into .pdf conversion is, that the newest Acrobat 10 Pro X does not allow for the run command for library inclusion. Photoshop did not accept .eps M-cardiod (Could not ...). AI-CS5 showed the picture full-page, not cropped to the BB. The result could be saved as .pdf, .svg. Maybe I should buy Jaws PDFcreator, which is advised by John Deubert. The trial version seems to do what I want, also handles library inclusion. But ... it does not crop. For the time being I'll try it out, taking their watermark stamp for granted.

My case rests, have fun and all the best.

Kees van der Laan
kisa1@xs4all.nl

25. PSView is extremely fast as previewer, allows PS library inclusion via the run command as well, gives error message via a GhostScript window, but ... doesn't provide for .pdf output, alas.
26. Why not deliver TEXworks on the TEX Collection DVD with the dictionaries already inserted? Agreed, inserting them is easy.



Mandelbrot's view of 'Breskens'

Appendix: Lauwerier's BASIC codes into PostScript

Lauwerier used overtime various variants of BASIC: starting with the GW-BASIC interpreter in the 80s and ending with the PowerBASIC compiler in the 90s.

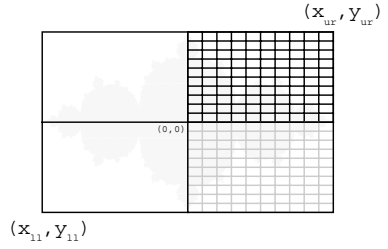
BASIC screen settings and interrupting commands have been neglected in the conversion, because PostScript is batch-oriented and abstracts from the various screens and printer devices. The artefacts `XM=320` and `YM=200` in the BASIC codes reflect the screen size of those days 640×400 pixels.

Single precision is used in PostScript throughout.

The real part and imaginary part of the problem constant $c \in \mathbb{C}$ are called `A` and `B` in BASIC and provided as parameters to the `def`s in PostScript; they are stored in the local dictionary as `a` and `b`.

BASIC uses `SIN` and `COS` with the angle given in radians; PostScript assumes angles in degrees.

`SQR` converts into `sqrt`.



Instead of `DELH` and `DELV` (halfwidths) the equivalent x_{ur} and y_{ur}, coordinates of right-upper corner of the fractal domain, are used as variables for the point-symmetric rectangular which contains the fractal associated with `c`, because these quantities are more easily associated with PostScript's `BoundingBox`, which is just the scaled fractal domain. In PostScript these values are provided as arguments to the `def`s and stored locally as x_{ur} and y_{ur}. However, in `MANDET` and `MANDIS`, for details of the M-fractals, I have chosen for a fixed-sized `BoundingBox`, similar to Lauwerier's screen-size, where the various magnifications are shown.

The nodes of the grid, which cover the 'first quadrant' of the rectangular which contains the fractal, are tested in nested loops on whether they belong to the JULIA set or not. If $b \neq 0$ the nodes of the 4^{th} quadrant of the rectangle also have to be tested. The other nodes of the fractal domain are point-symmetric with the tested nodes.

From the inner-loop Lauwerier escapes pragmatically by an escape `GOTO`, for testing the next node, when the iterated value is of no longer use. The escape translates directly into `exit` from the (loop)procedure as argument of PostScript's `loop`. The maximum number of iterations of the inner loop is stored in the variable `KMAX` in BASIC, which is an argument to the `def` in PostScript, and stored locally as `kmax`.

The test criterion is method-dependent.

For colours Lauwerier used a permutation array `COL`, such that he could select a few colours, e.g. 10, in a convenient order from the 16 standard colours in BASIC.

Standard in BASIC: 0=black, 1=blue, ... , 14=yellow, 15=bright white

Lauwerier's order of selection: 0=black, 1=blue, 2=light blue, ... , 8=brown, 9=yellow.

```
REM ***Colours***
DIM COL(8) : DATA 0,0,1,9,4,12,6,14,2
FORKI=0 TO 8 : READ COL(K) : NEXT K
REM Colours are selected by assignment of an integer to L
PSET( I,J), L ; PSET( I,-J), L
```

```
/Courier 14 selectfont
/colours [/black /black /blue /lightblue
          /red /lightred /brown /yellow /green] def
%Colours are selected by assignment of an integer to l
colours l get cvx exec
i j moveto (.) centershow i j neg moveto (.) centershow
```

For `PSET(i,j)` the dots of the `Courier` font at 12pt have been used and are centered.

In the BASIC codes for $j = 0$, the x-axis, the pixels are printed twice; in PostScript this yields too black results. In the converted codes this double printing is avoided. For printing the pixels `centershow` from `PSlib.eps` has been used.

Appendix: JULIAMC into a PostScript library def

The program performs kmax+10 inverse iterations of which the first 10 are neglected.

Inverse Iteration:
$$x_{k-1} = (x_k - a)/2 \pm \sqrt{((x_k - a)/2)^2 + ((y_k - b)/2)^2} \qquad k = \text{kmax+10}, \ldots, 1$$
$$y_{k-1} = (y_k - b)/(2x_{k-1})$$

RND < .5 is translated into rand 1073741823 gt, because PostScript's random generator, rnd, yields an integer in the range 0 to $2^{31} - 1$, maximum integer (Mersienne's number), while BASIC's RND yields a number in the range 0 to 1.

```
A=.3 : B=0
KMAX=40000: K=0 : X=RND : Y=RND
DO WHILE K<KMAX AND INKEYS$="" 'interrupt possibility
   X1=(X-A)/2 : Y1=(Y-B)/2 : R=SQR(X1*X1+Y1*Y1)
   IF RND < .5 THEN
      X=SQR(R+X1) : Y=SQR(R-X1)
      IF Y1<0 THEN Y=-Y
   ELSE
      X=-SQR(R+X1) : Y=-SQR(R-X1)
      IF Y1<0 THEN Y=-Y
   END IF
   IF K>10         THEN PSET( X,Y) : PSET(-X,-Y)
   IF K>10 AND B=0 THEN PSET(-X,Y) : PSET( X,-Y)
   K=K+1
LOOP
END
```

```
/JULIAMC{%stack: a, b, maxk.
         %a+ib complex constant c,  maxk maximal iterations
         % ==> Julia set of z^2 + a + ib
JULIAMCdict begin /Courier 12 selectfont
/kmax exch def /b exch def /a exch def
/nextpoint{/x1 x a sub 2 div def /y1 y b sub 2 div def
           x1 dup mul y1 dup mul add sqrt dup%R
          /y exch x1 sub sqrt y1 0 lt{neg}if def
          /x exch x1 add sqrt def
}bind def
/printxy{x s y s        moveto (.) centershow
    x s neg y s neg     moveto (.) centershow %point symmetry
    b 0 eq y 0 ne and
       {x s y s neg moveto (.) centershow
        x s neg y s moveto (.) centershow}if  %symmetry x- and y-axes
}bind def
/s {100 mul} def                              %scaling
/nrand rand 2147483647 div def                %random number in [0,1]
/x nrand def /y nrand def                     %start values in [0,1]
10{nextpoint}repeat                           %discard 10 iterations
kmax{nextpoint
     rand 1073741823 gt {/x x neg def /y y neg def}if
     printxy
    }repeat
end}bind def
%
/JULIAMCdict 11 dict def                      %local dictionary
```

In his first book on Fractals Lauwerier also provided for the backtracking method, by pursuing both points of the inverse iteration. Not needed.

Appendix: JULIABS into a PostScript library def

The program tests the size of $z_k^2 + c$ for $k = 0, 1, 2, 3, \ldots,$ kmax at the 4 corners of a (small) square around each node of the 'grid.' When not at all 4 cornerpoints the sequence z_n goes to ∞, (4 in the program) i.e. the sequence z_n stays finite, then the grid point is considered to belong to the boundary of the fractal, because on the boundary (and inside the contour) the sequence of points remain on the fractal (or stay finite).

```
REM***JULBS of z*z+c, Boundary Scan
A=0 : B=.65
KMAX=80 : DELV=1.3 : DELH=1.3
N1=200: N2=INT(N1*DELV/DELH)
IF B=0 THEN N3=0 ELSE N3=N2
FOR I=0 TO N1 : FOR J=-N1 TO N2
   T=0 : IF INKEYS$<>"" THEN END
   X=(I+.5)*DELH/N1 : Y=(j-.5)*DELV/N2 : GOSUB jcycle
   X=(I+.5)*DELH/N1 : Y=(j+.5)*DELV/N2 : GOSUB jcycle
   X=(I-.5)*DELH/N1 : Y=(j-.5)*DELV/N2 : GOSUB jcycle
   X=(I-.5)*DELH/N1 : Y=(j+.5)*DELV/N2 : GOSUB jcycle
   IF T=0 OR T=4 THEN GOTO repeat
   PSET (I,J), 14 ; PSET (-I,-J), 14
   IF B=0 THEN PSET (I,-J), 14 ; PSET (-I,J), 14
repeat:
NEXT J : NEXT I
END
jcycle:
FOR K-1 TO KMAX
   X1=X*X-Y*Y+A : Y1=2*X*Y+B
   IF X1*X1+Y1*Y1>4 THEN T=T+1 : EXIT FOR
   END IF
   X=X1 : Y=Y1
NEXT K
RETURN
END
```

```
/JULIABS{%Stack: a b (a+ib in C) x_ur y_ur kmax
         %==> fractal
JULIABSdict begin /Courier 12 selectfont
/kmax exch def /y_ur exch def /x_ur exch def
/b exch def /a exch def
/step .01 def /hstep step 2 div def /sc {100 mul} def
/print-xsc-ysc{xsc     ysc     moveto (.) centershow
         xsc neg ysc neg moveto (.) centershow%point symmetry
         b 0 eq {xsc     ysc neg  moveto (.) centershow
                 xsc neg ysc      moveto (.) centershow}if
        }def
/jcycle{/k 0 def
        {x dup mul y dup mul add 4 gt{/t t 1 add def exit}if
         /xn x dup mul y dup mul sub a add def
         /y 2 x mul y mul b add def /x xn def
         /k k 1 add def
         k kmax eq {exit}if
        }loop
       } bind def
0 step x_ur{/xgrid exch def /xsc xgrid sc def  %for i, loop over grid
   b 0 eq{0}{y_ur neg}ifelse step y_ur
     {/ygrid exch def /ysc ygrid sc def%for j
      /t 0 def
      /x xgrid hstep add def
      /y ygrid hstep sub def jcycle
      /x xgrid hstep add def
      /y ygrid hstep add def jcycle
      /x xgrid hstep sub def
      /y ygrid hstep add def jcycle
      /x xgrid hstep sub def
      /y ygrid hstep sub def jcycle
      t 0 ne t 4 ne and {print-xsc-ysc}if
   }for %j
}for %i
end}bind def
%
/JULIABSdict 40 dict def
```

Appendix: JULIAF into a PostScript library def

The forward iteration is implemented. When the iterate goes to ∞ then the iterate is skipped; when after kmax iterations the iterate stays finite, < 100 say, the point is considered to be part of the JULIA fractal or inside the fractal.

```
REM***JULIAFill of z*z+c, Pixel method***
REM**Lauwerier(1994, p142)***
A=-.8 : B=.15 'Dragon filled
KMAX=200 : DELV=1 : DELH=1.6
N1=250: N2=INT(N1*DELV/DELH)
IF B=0 THEN N3=0 ELSE N3=-N2
FOR I=0 TO N1 : FOR J=N3 TO N2
   X=I*DELH/N1 : Y=J*DELV/N2
   FOR K=1 TO KMAX
      X1=X*X-Y*Y+A: Y1=2*X*Y+B : S=X*X+Y*Y
      IF S>1000 THEN GOTO repeat
      X-X1 : Y=Y1
   NEXT K
   PSET (I,J) ; PSET (-I,-J)
   IF B=0 THEN PSET (I,-J) ; PSET (-I,J)
repeat: NEXT J : NEXT I
END
```

```
/JULIAF{%Stack: a b x_u y_u kmax
        %       c=a+ib, window of fractal, maximum iterates
        %==> Filled JULIA fractal
JULIAFdict begin /Courier 12 selectfont
/kmax exch def /y_u exch def /x_u exch def
/b exch def /a exch def
/pr{x sc y sc moveto (.) centershow
    x sc neg y sc neg moveto (.) centershow
    b 0 eq{x sc y sc neg moveto (.) centershow
           x sc neg y sc moveto (.) centershow}if}def
/sc {100 mul} def
/step 0.01 def
0 step x_u{/x exch def                                    %grid
   b 0 eq{0}{y_u neg}ifelse step y_u{/y exch def /z_i y def /z_r x def
      /k 0 def
      {/aux z_r dup mul z_i dup mul sub a add def%new iterate
       /zn_i 2 z_r mul z_i mul b add def /zn_r aux def
       z_r dup mul z_i dup mul add 100 gt{exit}if          %infty, outside
       zn_r z_r sub dup mul zn_i z_i sub dup mul add .0001 lt{pr exit}if
       /k k 1 add def
       k kmax eq{pr exit}if
       /z_r zn_r def /z_i zn_i def                        %inside
      }loop
   }for %j
}for %i
end}bind def
%
/JULIAFdict 22 dict def
```

Appendix: JULIAD into a PostScript library def

A powerful forward iteration program which iterates over $z_{n+1} = z_n^2 + c$ and the derivative $s_{n+1} = 2s_n z_n$. JULIADist can be used advantageously where other programs fall short.

At the heart lies the distance formula, Lauwerier(1990, 1996)[27]

$$d(z_0, J) \approx |z_n| \log |z_n| / |\frac{dz_n}{dz}|.$$

A curious, compact formula where $d(z_0, J)$ is expressed in $|z_n|$ and $|\frac{dz_n}{dz}|$.

My Complex Analysis skills have become a bit rusty after so many years of non-use, so I take Lauwerier's formula for granted.

```
REM***Lauwerier(1996) p112***
REM*** JULIA with distance formula***
A=-.55 : B= .15 'c=a+ib
DELH=2 : DELV= 1
N1=240 : N2=INT(N1*DELV/DELH) : F=.6
IF B=0 THEN N3=0 ELSE N3=-N2
FOR I=0 TO N1
   FOR J=N3 TO N2
     IF I=0 AND J=0 THEN GOTO nextpoint
     X=I*DELH/N1 : Y= J*DELV/N2 : u=1 : V=0
     FOR K=0 TO 200
     X1=X*X-Y*Y +A : Y1=2*X*Y+B
     U1=2(U*X-V*Y) : V1=2*(U*Y+V*X)
     S1=X1*X1+Y1*Y1+1E-10: S2=LOG(S1)
     S3=SQR(U1*U1+V1*V1+1E-10)
     IF S1>256 OR S3>256 THEN
        DIST=SQR(S1)*S2/S3
        IF DIST<F*DELH/N1 THEN
           PSET(I,J), 14 : PSET(-I,-J), 14
           IF B=0 THEN PSET(I,-J), 14 : PSET(-I,J), 14
           END IF GOTO nextpoint
        END IF
     END IF
     x=X1 : Y=Y1 : U=U1 : V=V1
     NEXT K
nextpoint: NEXT J : NEXT I
END

J(.1,-.8) =>
```

```
/JULIAD{%Stack: a b x_ur y_ur kmax f
        % c, window of fractal, maximum iterates, distance threshold
        %==>JULIA fractal by distance formula
JULIADdic begin /Courier 12 selectfont
/f exch def/kmax exch def /y_ur exch def /x_ur exch def
/b exch def /a exch def
/step .01 def /sc {100 mul} def %/f .001 def%thickness
0 step x_ur{/xl exch def /xsc xl sc def
 b 0 eq {0}{y_ur neg}ifelse step y_ur{/y exch def /x xl def
  /ysc y sc def
  /u 1 def /v 0 def
  /k 0 def
   {/k k 1 add def
    k kmax eq {exit}if
      /x1 x dup mul y dup mul sub a add def
      /y1 2 x mul y mul b add def
      /u1 2 u x mul v y mul sub mul def
      /v1 2 u y mul v x mul add mul def
      /s1 x1 dup mul y1 dup mul add def
      /s2 s1 0.00001 add log def
      /s3 u1 dup mul v1 dup mul add .1 add sqrt def
      s1 256 gt s3 256 gt or
       {/dist s1 sqrt s2 mul s3 div def
        dist f lt
        {xsc ysc moveto (.) centershow
         xsc 0 ne{xsc neg ysc neg moveto (.) centershow}if
         b 0 eq{xsc ysc neg moveto (.) centershow
                xsc neg ysc moveto (.) centershow}if
        }if
        exit
       }if
       /x x1 def /y y1 def /u u1 def /v v1 def
    }loop%k
 }for%j
}for%i
end}bind def
%
/JULIADdict 31 dict def
```

27. Formula in Lauwerier(1996) contains the log of the derivative; an error. Lauwerier(1990) is better.

Appendix: JULIAP into a PostScript library def

JULIAP, appended P stands for Pixel method, performs the forward iteration. The process is based on the property that when z_0 is outside the Julia fractal the iterate goes to ∞ and when z_0 belongs to the fractal it remains on the fractal in a chaotic way. The value of the inner-loop variable is maintained and when the value of the iterate is greater than 100, say, the index of the iterate, called escape number, is associated with a colour and the initial point, z_0, is coloured by this colour. If the initial point is on the Julia fractal or converges to an attractor inside the fractal then the loop control variable is again associated with a colour and the initial point, z_0, coloured by this colour. Either $k = kmax$, and z_0 was on the fractal already, or when $k < kmax$ on the escape fulfilled the test by the Cauchy criterion, the sequence converged to an attractor. The points z_0 are taken systematically from the set of nodes of the $2N_1 - 1 \times 2N_2 + 1$ grid laid over the rectangle with left lower corner (`-delh`, `-delv`) and right upper corner (`delh`, `delv`). The user must know beforehand that the fractal lies within the rectangle and supplies `delh` and `delv` as parameters to the invoke of `JULIAP`.

```
REM***Lauwerier(1996, p148):  Graphics and Fractals***
A=-.55 : B= .15 'c=a+ib
DELH=2 : DELV=1.8'adapted
N1=200 : N2=INT(N1*DELV/DELH)
IF B=0 THEN N3=0 ELSE N3=-N2
FOR I=0 TO N1
   FOR J=N3 TO N2
     X=I*DELH/N1 : Y= J*DELV/N2
     FOR K=0 TO 200
     X1=X*X-Y*Y +A : Y1=2*X*Y+B
     S =X*X+Y*Y : S2=LOG(S1)
     S1=(X-X1)*(X-X1)+(Y-Y1)*(Y-Y1)
     IF S  >100 THEN L=1 +k mod 15 : GOTO graphics
     IF S1<.001 THEN L=1 +k mod 15 : GOTO graphics
     x=X1 : Y=Y1
   NEXT K: L=0
graphics: PSET(I,J), L : PSET(-I,-J), L
          IF B=0 THEN PSET(I,-J), L : PSET(-I,J), L
   NEXT J : NEXT I
END
```

```
/JULIAPdict 25 dict def
%
/JULIAP%On stack: a b, x_ur y_ur kmax
       %          c, right-upper of window of fractal, maximum iterates
       %==>Banded coloured Fractal
{JULIAPdict begin /Courier 12 selectfont
/kmax exch def /y_ur exch def /x_ur exch def /b exch def /a exch def
/sc {100 mul} def /step 0.01 def
/colours [/black /blue /lightblue /green /lightgreen
           /red /lightred /brown /yellow] def
0 step x_ur{/xgrid exch def /xsc xgrid sc def
 b 0 eq{0}{y_ur neg}ifelse step y_ur{/y exch def /ysc y sc def
  /x xgrid def
  /k 0 def /l 0 def
  {/k k 1 add def
    k kmax eq {exit}if
      /x1 x dup mul y dup mul sub a add def
      /y1 2 x mul y mul b add def
      /s x1 dup mul y1 dup mul add def
      /s1 x x1 sub dup mul y y1 sub dup mul add def
      s   100 gt {/l k 9 mod def exit}if
      s1 .001 lt {/l k 9 mod def exit}if
      /x x1 def /y y1 def
   }loop%k
   colours l get cvx exec
   xsc ysc moveto (.) centershow
   xsc neg ysc neg moveto (.) centershow
   b 0 eq {xsc ysc neg moveto (.) centershow
           xsc neg ysc moveto (.) centershow
          }if
 }for%j
}for%i
end}bind def
```

Appendix: JULIAS into a PostScript library def

The dynamical system for an m-fold rotation symmetric fractal reads, Lauwerier(1996, 129), borrowed from Field&Golubitsky(1992))

$$z_{n+1} = z_{n+1}(a + b/(z^m + c)) \quad \text{with} \quad 1 = a + b/(1 + c).$$

The parameters of JULIAS are a, b, m and kmax.

```
REM ***Kleuren***
DIM COL(8) : DATA 0,0,1,9,4,12,6,14,2
FOR I=0 TO 8 : READ COL(I) : NEXT I
M=8 : A=.5 : B=.35: C=(A+B-1)/(1-A) : N=200
R=3.5 'Radius circle window
FOR I=0 TO N : FOR J=0 TO I
   X=I*R/N : Y=J*R/N
   IF X*X+Y*Y >R*R THEN  GOTO repeat
   FOR K=1 TO 200
      X2=X*X-Y*Y : Y2=2*X*Y
      X4=X2*X2-Y2*Y2 : Y4=2*X2*Y2
      X8=X4*X4-Y4*Y4 : Y8=2*X4*Y4
      XN=X8+C : YN=Y8
      S=XN*XN+YN*YN+1E-8
      XT=XN/s : YT=-YN/S
      X1=A+B*XT :Y1=B*YT
      X0=X*X1-Y*Y1 : Y0=X*Y1+Y*X1
      S0=X0*X0+Y0*Y0 : S1=(X8-1)*(X8-1)+Y8*Y8
        IF S0<.0001 OR S1<1E-08 THEN
           L=COL(1+K mod 8) : GOTO graphics
        END IF
      X=X0 :Y=Y0
   NEXT K : L=0
graphics:
   PSET( I,J), L ; PSET( I,-J), L
   PSET(-I,J), L ; PSET(-I,-J), L
   PSET( J,I), L ; PSET( J,-I), L
   PSET(-J,I), L ; PSET(-J,-I), L
repeat:
NEXT J : NEXT I
END
```

```
/JULIAS{%On stack: a b n kmax, n is n-fold symmetry
        %==> Circle Symmetric Julia fractal
JULIASdict begin /Courier 12 selectfont
/kmax exch def /n exch def /b exch def /a exch def
/colours [/black /black /blue /lightblue
          /red /lightred /brown /yellow /green] def
/c a b add 1 sub 1 a sub div def
/r 3.5 def
0 1 n{/i exch def
 0 1 i{/j exch def
  /x i r mul n div def
  /y j r mul n div def
  /k 0 def /l 0 def
  x dup mul y dup mul add r dup mul lt
  {%\type{z}<\type{r}
  {/k k 1 add def
    k kmax gt {exit}if %exit k-loop
    /x2 x dup mul y dup mul sub def
    /y2 2 x mul y mul def
    /x4 x2 dup mul y2 dup mul sub def
    /y4 2 x2 mul y2 mul def
    /x8 x4 dup mul y4 dup mul sub def
    /y8 2 x4 mul y4 mul def
    /xn x8 c add def
    /yn y8 def
    /s xn dup mul yn dup mul add .00001 add def
    /xt xn s div def
    /yt yn neg s div def
    /x1 a b xt mul add def
    /y1 b yt mul def
    /x0 x x1 mul y y1 mul sub def
    /y0 x y1 mul y x1 mul add def
    /s0 x0 dup mul y0 dup mul add def
    /s1 x8 1 sub dup mul y8 dup mul add def
    s0 .0001 lt s1 .00000001 lt or{/l k cvi 8  mod def exit}if
    /x x0 def
    /y y0 def
   }loop %next k
   colours l get cvx exec
   i j         moveto (.) centershow
   i j neg     moveto (.) centershow
   i neg j     moveto (.) centershow
   i neg j neg moveto (.) centershow
   j     i     moveto (.) centershow
   j     i neg moveto (.) centershow
   j neg i     moveto (.) centershow
   j neg i neg moveto (.) centershow
   }if%\type{z}<\type{r}
}for%j
}for%i
end}bind def
%
/JULIASdict 35 dict def
```

Appendix: JULIASYMm into a PostScript library def

The requirement on the 2D dynamical system $F(z,\bar{z})$ in order that the strange attractors are rotation symmetric, reads

$$F(z,\bar{z}) = c * F(cz,\bar{c}\bar{z}) \quad \text{with} \quad c^m = 1 \quad \bar{c} = 1/c \quad \text{the conjugate.}$$

For $F(z,\bar{z})$ Field&Golubitsky(1992) used

$$F(z,\bar{z}) = z(a + bz\bar{z} + cz^m) + d\bar{z}^{m-1}.$$

```
A=2.5 : B=-2.5: C=0 : D=.9 : H=100 : N=6 :
XM=320 :  YM=240  : KMAX=2000 :
K=0 :  L=14 : PI=4*ATN(1)
DIM C(N), S(N)
FOR I=0 TO N-1
   C(I)=COS(2*PI*I/N) : S(I)=SIN(2*PI*I/N)
NEXT I
X=RND/10 : Y= RND/10              'start
DO WHILE K<KMAX
FOR I=0 TO N-1 : FOR J=0 TO I
   S=X*X+Y*Y : X1=X : Y1=Y
   FOR I=1 TO N-2                 'Z^N
      X2=X*X1-Y*Y1 : Y2=X*Y1+Y*X1
      X1=X2 : Y1=Y2
   NEXT I
   XN=X*X1-Y*Y1
   P=A+B*S+C*XN
   XNEW=P*X+D*X : YNEW=P*Y-DY1:
   XH=H*XNEW : YH=H*YNEW          'scaling
   FOR J=1 TO N-1    'rotation
      XHJ=C(J)*XH-S(J)*YH
      XHJ=S(J)*XH+C(J)*YH
      PSET(XM+XHJ, YM+YHJ), L
   NEXT J
   X=XNEW : Y=YNEW
   K=K+1
LOOP: BEEP
END
```

```
/JULIASYMm{%On stack: a b c d h n xm ym kmax
          %=> circle symmteric fractal
JULIASYMmdict begin /Courier 12 selectfont
/kmax exch def /ym exch def /xm exch def
/n exch def /h exch def
/d exch def /c exch def /b exch def /a exch def
/co n array def /si n array def
0 1 n 1 sub{/i exch def
   co i 360 i mul n div cos put
   si i 360 i mul n div sin put
   }for
22121943 srand
/x rand 21474836470 div def
/y rand 21474836470 div def
/k 0 def
{/k k 1 add def
 k kmax gt{exit}if
 /s x dup mul y dup mul add def
 /x1 x def /y1 y def
 1 1 n 2 sub{/i exch def
    /x2 x x1 mul y y1 mul sub def
    /y2 x y1 mul y x1 mul add def
    /x1 x2 def /y1 y2 def
    }for%i
 /xn x x1 mul y y1 mul sub def
 /p a b s mul add c xn mul add def
 /xnew p x mul d x1 mul add def
 /ynew p y mul d y1 mul sub def
 /xh h xnew mul def
 /yh h ynew mul def
 0 1 n 1 sub{/j exch def
    /xhj co j get xh mul si j get  yh mul sub def
    /yhj si j get xh mul co j get  yh mul add def
    xm xhj add ym yhj add moveto (.) centershow
    }for%j
 /x xnew def
 /y ynew def
 }loop%k
end}bind def
%
/JULIASYMmdict 35 dict def
```

Appendix: FRACSYMm into a PostScript library def

```
A=-.1 : B=.4: C=.2 : D=.5 : E=.5 F=.4
SC=175 : M=7 : KMAX=4000
K=0 : : L=14
DIM C(M), S(M)
FOR I=0 TO M-1
   C(I)=COS(2*PI*I/N) : S(I)=SIN(2*PI*I/N)
NEXT I
X=RND/10 : Y= RND/10              'start
DO WHILE K<KMAX
  Z=X : X=A*X+B*Y+E : Y=C*Z+C*Y+F
  FOR J=0 TO M-1
    X1=C(J)*X-S(J)*Y : Y1=S(J)*X+C(J)*Y
    IF K>32THEN PSET(SC*X1, SC*Y1), L
  NEXT J
  N=INT(M*RND) : K=K+1
  X=C(N)*X1-S(N)*Y1
  Y=S(N)*X1+C(N)*Y1
LOOP
END
```

```
/FRACSSYMm{%Stack: a b c d e f m sc kmax
          %==> Cirle Symmetric Jula fractal by ITS
FRACSYMmdict begin /Courier 12 selectfont
/kmax exch def /sc exch def /m exch def
/f exch def /e exch def /d exch def
/c exch def /b exch def /a exch def
/co m array def /si m array def
0 1 m 1 sub{/i exch def
   co i 360 i mul m div cos put
   si i 360 i mul m div sin put
   }for
22121943 srand
/x rand 21474836470 div def
/y rand 21474836470 div def
/k 0 def
{/k k 1 add def
 k kmax gt{exit}if
 /z x def
 /x a x mul b y mul add e add def%transform
 /y c z mul d y mul add f add def
 0 1 m 1 sub{/j exch def         %rotate
    /x1 co j get x mul si j get y mul sub def
    /y1 si j get x mul co j get y mul add def
    k 32 gt{ x1 sc mul y1 sc mul moveto (.) centershow}if
    }for %j
/n m rand 2147483647 div mul cvi def
/x co n get x1 mul si n get y1 mul sub def
/y si n get x1 mul co n get y1 mul add def
}loop%k
end}def
%
/FRACSYMmdict 25 dict def
```

Appendix: Lauwerier(1987, p156) Mandelbrot program

This section contains the program which in fact yields the answer to the question posed in the introduction:

> "For which values of c will the Julia fractal, J(c), be line-like and for which dust-like?"

I not only 'converted' his first code for the Mandelbrot set, but also improved his coding, made it straight and better readable, IMHO.

From the Mandelbrot Wikipedia I borrowed the following pseudo-code, which clearly exhibits the structure of an M-fractal code,[28]without efficiency short-cuts.

```
For each pixel on the screen do:
x0 = scaled x coordinate of pixel (must be scaled to lie somewhere in the mandelbrot X scale (-2.5 to 1)
y0 = scaled y coordinate of pixel (must be scaled to lie somewhere in the mandelbrot Y scale (-1, 1)
x = 0  y = 0
iteration = 0   max_iteration = 1000
while ( x*x + y*y < 2*2  AND  iteration < max_iteration )
  {xtemp = x*x - y*y + x0  y = 2*x*y + y0   x = xtemp
   iteration = iteration + 1}
color = iteration  plot(x0, y0, color)
```

Purpose of he program Visualize in the (a,b)-plane the points (a,b) for which J(a,b) is connected.

Property If $\lim_{k\to\infty} z_k(a,b) = \infty$ with $z_0 = (0,0)$ then J(a, b) is dust-like.

```
100 p1= -2.5 : p2=-1.5 : p3=1.5 : p4=1.5
110 N1=250 : N2=INT(.833*N1*(P4-P2)/(P3-P1)}
120 FOR I=-N1 TO N1 : A=(({N1-I)*P1+(N1+I)*P3/(2*N1)
130     FOR J=0 TO N2 : B=(({N2-J)*P1+(N2+J)*P3/(2*N2)
140         X=A : Y=B
150         FOR K=0 TO 50
160         Z=X : X=X*X-Y*Y+A
170               y=2Y*Z+B
180         IF X*X+Y*Y>16 THEN GOTO 200
190        NEXT K
200        PSET(320+I, 200-J), K MOD 2
205        PSET(320+I, 200+J), K MOD 2
210     NEXT J
220 NEXT I
230END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Mandelbrotzw filled, of z^2+a+ib
%%Author: H.A. Lauwerier(1987): Fractals, p156
%%Creator: Kees van der Laan, June 2012
%%BoundingBox:  -400 -300 400 300
%%BeginSetup
%%EndSetup
%%BeginProlog
/man1987{%==> Mandelbrot fractal
man1987dict begin /Courier 12 selectfont
/sc {400 mul} def                                   %scaling for output
/pr{a sc b sc moveto (.) centershow
    a sc b sc neg moveto (.) centershow             %symmetry
    b 0 ne{a sc b sc neg moveto (.) centershow}if}def
/a_l -1.75 def /b_l -1 def /a_u .5 def /b_u 1 def%domain M-fractal
/kmax 50 def
a_l .01  a_u{/a exch def                            %over grid points
   0 .01 b_u{/b exch def
      /x a def /y b def /k 0 def
       {/z x def                                    %loop k; z auxiliary
        /x x dup mul y dup mul sub a add def        %real part z^2+a+ib
        /y 2 y mul z mul b add def                  %imag part z^2+a+ib
        x dup mul y dup mul add 16 gt{exit}if       %\type{z_k}>16?(infinite)
        /k k 1 add def
        k kmax eq {pr exit}if                       %z_kmax=finite
       }loop
   }for%j
}for%i
end}bind def
%
/man1987dict 16 dict def
%%Endprolog
%
%   Program ---the script---
%
man1987 showpage
%%EOF
```

28. Or see the code of Wim W. Wilhelm, given earlier.

The black figure denotes the points for which J(a,b) is line-like, also called connected.

Coding Improvements

- ☐ the naming of the domain of the $M_{a,b}$-fractal, with $a \in [a_l, a_u]$ and $b \in [b_l, b_u]$;
- ☐ improved bounds for the $M_{a,b}$-fractal domain
- ☐ the loop for points on the a-axis starts with a_l and ends by a_u, the loop for points on the b-axis starts with 0 and ends by b_u
- ☐ the coordinates for printing are scaled values of the loop control variables a and b
- ☐ the dots are centered
- ☐ no fancy k MODE 2 is used for black-and-white bands, nor coloured bands
- ☐ `%%BoundingBox:` a_l b_l a_u b_u, each quantity scaled by `sc`
- ☐ the confusion between z_0 and (a, b) as starting value is avoided; better is to talk about the universal starting value $z_0 = 0$, which entails $z_1(a, b) = a + ib$
- ☐ the calculation of $\sqrt{a^2 + b^2}$, the 2-norm of (a,b), is protected against unnecessary intermediate overflow
- ☐ `kmax` varies in the various programs, subjective.

In Lauwerier(1989) the program `MANDELP` is more efficient because the inner (k-)loop is skipped for points inside the cardioid or inside the circle, $C_{(-1,.25)}$.

In Lauwerier(1994, 1996) the Mandelbrot program has been further improved by the use of the distance formula which yields better (hairy) details of the M-fractal. We will come back on both aspects in the next appendix `MANDELx`.

Appendix: MANDISm into a PostScript library def

At th heart lies the distance formula, Lauwerier(1996,116) similar as in JULIAD

$$d(c, M) \approx |p_n| \log |p_n| / |\frac{dp_n}{dc}|.$$

```
XM=320 : YM=240 'half old screen size
DELH=1.35 : DELV=1.1 : AC=-.65
N1=200 : N2=INT(N1*DELV/DELH)
FOR I=-N1 TO N1 : A=AC+I*DELH/N1
 FOR J=0 TO N1 : B=J*DELV/N2
   S1=4*(A*A+B*B) : S2=S1-2A+1/4
   IF S1+8*A+15/4<0 THEN COL=14 : GOTO graphics
   IF S2-SQR(S2)+2*A-1/2<0 THEN COL=14 : GOTO graphics
   X=A : Y=B : U=1 : V=0
   FOR K=0 TO 100
     X1=X*X-Y*Y+A : Y1=2*X*Y+B
     U1=2*(U*X-V*Y)+1 : V1=2(U*Y+V*X)
     W1=X1*X1+Y1*Y1
     X=X1 : Y=Y1 : U=U1 : V=V1
     DIS=SQR(W1)*LOG(W1)/SQR(U1*U1+V1*V1)
     IF W>64 THEN
        IF DIS>.4*DELH/N1 THEN COL=0 ELSE COL=14
             GOTO graphics
        END IF
  NEXT K : COL=14
graphics:
  PSET(XM+I,YM-J), COL : PSET(XM+I,YM+J), COL
 NEXT J
NEXT I
END
```

```
/MANDISm{%stack: ac bc del kmax f
         %(centre of detail, width, max number iterations distance
         % parameter
         %==> mandelbrot fractal by distance formula monochrome
MANDISmdict begin /Courier 12 selectfont
/f exch def /kmax exch def /del exch def /bc exch def /ac exch def
/pr{i j moveto (.) centershow}def
/n 400 def %mimics fixed  BB window
n neg 1 n{/i exch def
  -.75 n mul 1 .75 n mul{/j exch def
     /a ac i n div del mul add def
     /b bc j del mul n div add def
     /x a def /y b def /u 1 def /v 0 def
     /k 0 def%loop over k
     {/x1 x dup mul y dup mul sub a add def
      /y1 2 x mul y mul b add def
      /u1 2 u x mul v y mul sub mul 1 add def
      /v1 2 u y mul v x mul add mul def
      /s1 x1 dup mul y1 dup mul add  .0000001 add def
      s1 128 gt
      {/s2 s1 .0000001 add log def
       v1 abs 1 gt
         {/s3 u1 v1 div dup mul 1 add sqrt v1 abs mul def}
         {/s3 v1 u1 div dup mul 1 add sqrt u1 abs mul def}ifelse
       /dist s1 sqrt s2 mul s3 div def
       dist f lt {red pr}if
       exit
      }if
      /x x1 def /y y1 def /u u1 def /v v1 def
      /k k 1 add def
      k kmax eq {exit}if
     }loop%k
   }for%j
}for%i
end}bind def
%
/MANDISmdict 30 dict def
```

Appendix: MANDELx into a PostScript library def MANDEL and variants

```
DELH=1.6 : DELV=1.34 : AC=-.65
N1=260 : N2=INT(N1*DELV/DELH)
DIM COL(8) : DATA 0,1,9,2,10,4,1,6,14
FOR I=0 TO 8 : READ COL(I) : NEXT I
FOR I=-N1 TO N1 : A=AC+I*DELH/N1
   FOR J=0 TO N1 : B=J*DELV/N2
       U=4*(A*A+B*B) : V=U-2*A+1/4
       IF U+8*A+15/4<0 THEN L=0 : GOTO repeat
       IF V-SQR(V)+2*A-1/2<0 THEN L=0 : GOTO repeat
       X=A : Y=B : K=0
       DO
         Z=X : X=X*X-Y*Y+A : Y=2*Z*Y+B
         S=X*X+Y*Y : K=K+1
       LOOP UNTIL S>100 OR K=50
       IF K<40 THEN L=1+K MOD 8 ELSE L=0
       IF K>3 THEN
          PSET(I,J), COL(L) : PSET(I,-J), COL(L)
       END IF
repeat:
NEXT J
NEXT I
END
```

```
/MANDEL%==>Mandelbrotfractal in coloured bands
{MANDELdict begin /Courier 12 selectfont
/colours [/white /blue /lightblue /green /lightgreen
          /red /lightred /brown /yellow] def
/step .01 def /sc {100 mul} def
-2.1 step .85{/a exch def /asc a sc def
   0 step 1.35 {/b exch def /bsc b sc def
     /u 4 a dup mul b dup mul add mul def
     /v u 2 a mul sub .25 add def
     u 8 a mul add -3.75 le           %exclude cardioid
     v v sqrt sub 2 a mul add .5 le or%exclude circle
     {/l 0 def}%inside white, do nothing
     {/x a def /y b def /k 0 def
         {%loop over k
          /z x def
          /x x dup mul y dup mul sub a add def
          /y 2 z mul y mul b add def
          /s x dup mul y dup mul add def
          /k k 1 add def
           s 100 gt k 50 eq or{exit}if
         }loop%k
       k 40 lt{/l k 8 mod def}{/l 0 def}ifelse
       colours l get cvx exec
       k 3 gt{asc bsc     moveto (.) centershow
              asc bsc neg moveto (.) centershow}if
     }ifelse
   }for%j
}for%i
end}bind def
%
/MANDELdict 20 dict def
```

Variants

```
/MANDELzw%==>Mandelbrot fractal in black and white bands
{MANDELzwdict begin /Courier 12 selectfont /l 0 def
/step .01 def /sc {200 mul} def
-2.1 step .85{/a exch def /asc a sc def
   0 step 1.37 {/b exch def /bsc b sc def
     /u 4 a dup mul b dup mul add mul def
     /v u 2 a mul sub .25 add def
     u 8 a mul add -3.75 ge            %exclude cardioid
     v v sqrt sub 2 a mul add .5 ge and%exclude circle
     {/x a def /y b def /k 0 def
         {%loop over k
          /z x def
          /x x dup mul y dup mul sub a add def
          /y 2 z mul y mul b add def
          /s x dup mul y dup mul add def
          /k k 1 add def
           s 100 gt k 50 eq or{exit}if
         }loop%k
       k 40 lt{/l k 8 mod def}{/l 1 def}ifelse
       l 2 idiv 2 mul l eq{
       k 3 gt{asc bsc     moveto (.) centershow
              asc bsc neg moveto (.) centershow}if}if
     }if
   }for%j
}for%i
end}bind def
%
/MANDELzwdict 20 dict def
```

```
/MANDELzwcontour{%stack:
%==> contour mandelbrot fractal by distance formula in black
%    and white
MANDELzwcontourdict begin /Courier 12 selectfont
/pr{i j moveto (.) centershow i j neg moveto (.) centershow}def
/f .005 def/kmax 50 def /del 1.4 def /ac -.75 def
/n1 400 def /n2 300 def                  %size BB window
/thickness .8 def
n1 neg 1 n1{/i exch def
    /a ac del i n1 div mul add def
   0 1 n2{/j exch def
     /b j n2 div .85 del mul mul def
     /s1 4 a dup mul b dup mul add mul def
     /s2 s1 2 a mul sub .25 add def
     s1 8 a mul add -3.75 ge             %exclude cardioid
     s2 s2 sqrt sub 2 a mul add .5 ge and%exclude circle
     {/x a def /y b def /u 1 def /v 0 def
       /k 0 def                          %loop over k
       {/x1 x dup mul y dup mul sub a add def
        /y1 2 x mul y mul b add def
        /u1 2 u x mul v y mul sub mul 1 add def
        /v1 2 u y mul v x mul add mul def
        /w1 x1 dup mul y1 dup mul add def
        /x x1 def /y y1 def /u u1 def /v v1 def
        /dis w1 sqrt w1 log mul
            u1 dup mul v1 dup mul add sqrt div def
        w1 64 gt{dis f lt {pr} if exit}if
       /k k 1 add def k kmax eq {exit}if
      }loop%k
     }if
   }for%j
}for%i
end}bind def
%
/MANDELzwcontourdict 30 dict def
```

Appendix: MANDET into a PostScript library def and black-and-white variant

```
INPUT"x coordinate centrum = ", AC '2
INPUT"y coordinate centrum = ", BC '0
INPUT"halve breedte = ", DEL : CLS '000049
N1=200 : N2=INT(.75*N1) : KMAX=100
DIM COL(8) : DATA 0,1,9,2,10,4,12,6,14
FOR I=0 TO 8 : READ COL(I) : NEXT I
FOR I=-N1 TO N1 : A=AC+I*DELH/N1
   FOR J=-N2 TO N2 : B=BC+.75*J*DEL/N2
       X=A : Y=B: K=0
       DO
         Z=X : X=X*X-Y*Y+A : Y=2*Z*Y+B
         S=X*X+Y*Y : K=K+1
       LOOP UNTIL S>100 OR K=KMAX
       IF K<40 THEN L=1+K MOD 8 ELSE L=0
       PSET(I,J), COL(L)
repeat:
   NEXT J
NEXT I
END
```

```
/MANDET{%stack: a b del kmax
        %==> Mandelbrot fractal detail at (a,b)
MANDETdict begin /Courier 12 selectfont
/colours [/black /blue /lightblue /green /lightgreen
          /red /lightred /brown /yellow] def
/kmax exch def /del exch def /bc exch def /ac exch def
/n1 400 def /n2 n1 .75 mul cvi def %Mimics BoundingBox
n1 neg 1 n1{/i exch def
   /a ac i del mul n1 div add def
   n2 neg 1 n2{/j exch def
     /b bc j del mul n1 div add def
     /x a def /y b def /k 0 def
     {%loop over k
       /z x def
       /x x dup mul y dup mul sub a add def
       /y 2 z mul y mul b add def
       /s x dup mul y dup mul add def
       /k k 1 add def
       s 100 gt k kmax eq or{exit}if
     }loop%k
     k 40 lt{/l k 8 mod def}{/l 0 def}ifelse
     colours l get cvx exec
     i j moveto (.) centershow
   }for%j
}for%i
end}bind def
%
/MANDETdict 21 dict def
```

Black-and-white variant

-1.256 .3817 .005 50 MANDETzw

-1.749057 .000306 .05 50 MANDETzw

```
/MANDETzw{%stack: a b del kmax
        %==> mandelbrot fractal detail at (a,b)
MANDETzwdict begin /Courier 12 selectfont
/kmax exch def /del exch def /bc exch def /ac exch def
/n1 400 def /n2 n1 .75 mul cvi def
n1 neg 1 n1{/i exch def
   /a ac i del mul n1 div add def
   n2 neg 1 n2{/j exch def
     /b bc j del mul n1 div add def
     /x a def /y b def /k 0 def%loop over k
     {/z x def
      /x x dup mul y dup mul sub a add def
      /y 2 z mul y mul b add def
      /s x dup mul y dup mul add def
      /k k 1 add def
      s 100 gt k kmax eq or {exit}if
     }loop%k
     k 40 lt{/l k 8 mod def}{/l 0 def}ifelse
     l 2 idiv 2 mul l eq {i j moveto (.) centershow}if
   }for%j
}for%i
end}bind def
%
/MANDETzwdict 21 dict def
```

Appendix: Use of colours in PostScript

Simple use of colours in PostScript is explained. Colours appear opaque, not transparent. For transparent use of colours in PostScript, see Acumen Journal of Nov 2011. For colour gradients see the LRMlevel 3.

RGB-colour tables

In 2010 I found it useful to provide for color tables, such that one can see the result of each combination of colour parameters, despite the discrepancy between what is seen on the screen and what will result in print (Courtesy: Heck A (2005)). In MHO the supporting use of colours should not be that critical. For artists precise colours matter, of course.

It is curious that the LaTeX Graphics Companion does not contain colour tables, nor does mention the standard names for colours.

RGB-colour names

In `PSlib.eps` I have included the pre-settings of the rgb-colours with their names; handy. A snapshot for the parameters for `setrgbcolor` and their standard names is included below, next to an example of how to select colours by number, with irrelevant details omitted.

```
%Defs: selection of pre-settings of rgbcolors from PSlib.eps
/black    {0  0  0 setrgbcolor} def
/white    {1  1  1 setrgbcolor} def
/red      {1  0  0 setrgbcolor} def
/green    {0  1  0 setrgbcolor} def
/blue     {0  0  1 setrgbcolor} def
/greenblue{0  .7 1 setrgbcolor} def
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Selecting colours by number
%%Author: Kees van der Laan, May 2012
%%BoundingBox: 0 -5 110 15
(c:\\PSlib\\PSlib.eps) run%invoke library with colour pre-settings
/colours [/red /green /blue] def%array of colour pre-settings
%
0 0 moveto 100 0 lineto
colours 0 get cvx exec stroke showpage%Red line will show up

%%EOF
```

Use: `r g b setrsbcolor`, with each `r g b` a number in the range 0 to 1. Even simpler use: the mnemonic `greenblue`, e.g.

CMYK-colour tables

Standard CMYK-colour names

In PSlib.eps I have also included the settings of the cmyk-colours with their names; handy. A snapshot of the pre-settings for setcmykcolor and their standard names is included below.

```
% Procedures: colors
% procedures: Cmyk values  for use in PS a la pdfTeX
/cmykGreenYellow{0.15 0 0.69 0}def
...
/cmykGray{0 0 0 0.50}def
/cmykBlack{0 0 0 1}def
/cmykWhite{0 0 0 0}def
% procedures for cmyk colors
/GreenYellow{ cmykGreenYellow setcmykcolor } def
...
/Gray{ cmykGray setcmykcolor } def
/Black{ cmykBlack setcmykcolor } def
/White{ cmykWhite setcmykcolor } def
```

Use: c m y k setcmykcolor, with for each c m y k a number in the range 0 to 1. Even simpler use: the mnemonic GreenYellow, for example.

For my coloured slides, I could not precisely match the background colour in the pictures with the background colour of the slides, otherwise than with transparent pictures.

CrafTEX

Applying TEX and friends in crafts

Everything started at my job as Documentation manager in hi-tech industry: when the word processor gave up on our big fat instruction manual and the purpose-built software wasn't within budget, we ended up with ConTEXt. The transit period wasn't easy, but in time I learned to appreciate this software that does *not* make assumptions on what I'm attempting to do.

Thus, when I suddenly found myself with a textile craft book to be translated and prepared for printing, I thought of ConTEXt. Life happened and the booklet is still sitting on my desk waiting for its turn, but in the meanwhile I have learned many other things about TEX based systems and started to exploit their potential in crafts.

The experience has been mind-blowing! I come up with new implementations almost weekly, although I don't usually try things out until I have a real need for them. I am not a programmer, but I realize that a computer is especially useful in reducing the tedious repetitiveness of the planning stages. Nothing can ever prepare a crafter to what happens when you play with real yarn and real paper and glue, but the 'what if that's lighter blue' and 'I guess this really is the wrong font here' process can be sped up significantly by computer simulation.

I don't feel complete until I've shared my knowledge with others. I don't get that many face-to-face opportunities to do that, so I decided to go online instead, http://www.lucet.fi/. I haven't had the energy to fight with WordPress about the printing styles, so instead I'm planning to do printer-friendly pdf instructions with ConTEXt and MetaPost.

Besides enhancing my creativity, I also use ConTEXt to deal with the boring but necessary parts of having my own little craft business, e.g. for creating price lists and business cards. This migration is still very much in process, but eventually everything will be done with ConTEXt and possibly MetaPost and with as few different style definitions as possible.

Mari Voipio

MetaPost: PNG Output

Abstract
The latest version of Metapost (1.80x) has a third output backend: it is now possible to generate PNG bitmaps directly from within Metapost.

Introduction

For one of my presentations at EuroTEX2012 in Breskens, I wanted to create an animation in order to demonstrate a Metapost macro that uses timer variables to progress through a scene.

While working on that presentation, it quickly became obvious that the 'traditional' method of creating an animation with Metapost by using ImageMagick's `convert` to turn EPS images into PNG images was very time consuming. So much so, that I actually managed to write a new backend for Metapost while waiting for ImageMagick to complete the conversion.

Simple usage

Metapost will create a PNG image (instead of Encapsulated PostScript or Scalable Vector Graphics) by setting `outputformat` to the string `png`:

```
outputformat := "png";
outputtemplate := "%j-%c.%o";
beginfig(1);
fill fullcircle scaled 100 withcolor red;
endfig;
end.
```

This input generates a bitmap file with dimensions 100 x 100 pixels, with 8-bit/color RGBA. It shows a red dot on a transparent background.

Adjusting the bitmap size

In the simple example given above, Metapost has used the default conversion ratio where one point equals one pixel. This is not always desired, and it is tedious to have to scale the picture whenever a different output size is required.

To allow easy modification of the bitmap size independent of the actual graphic, two new internal parameters have been added: `hppp` and `vppp` (the names come from Metafont, but the actual meaning is specific to Metapost).

In Metapost, 'hppp' stands for 'horizontal points per pixel', and similarly for 'vppp'. Adding

```
hppp := 2.0;
```

to the example above changes the bitmap into 50 x 100 pixels. Specifying values less than 1.0 (but above zero!) makes the bitmap larger.

Adjusting the output options

Metapost creates a 32-bit RGBA bitmap image, unless the user alters the value of another new internal parameter: `outputformatoptions`.

The syntax for `outputformatoptions` is a space-separated list of settings. Individual settings are *keyword* + = + *value*. The only currently allowed ones are:

```
format=[rgba|rgb|graya|gray]
antialias=[none|fast|good|best]
```

No spaces are allowed on the left, nor on the right, of the equals sign inside a setting.

The assignment that would match the compiled-in default setup is:

```
outputformatoptions := "format=rgba antialias=fast";
```

however, `outputformatoptions` is initially the empty string, because that makes it easier to test whether a user-driven change has already been made.

Some notes on the different PNG output formats:

- The `rgb` and `gray` subformats have a white background. The `rgba` and `graya` subformats have a transparent background.
- The bitdepth is always 8 bits per pixel component.
- In all cases, the current picture is initially created in 8-bit RGB mode. For the `gray` and `graya` subformats, the RGB colors are reduced just before the actual PNG file is written, using a standard rule:

$$g = 0.2126 * r + 0.7152 * g + 0.0722 * b$$

- CMYK colors are always converted to RGB during generation of the output image using:

$$r = 1 - (c + k > 1?1 : c + k);$$
$$g = 1 - (m + k > 1?1 : m + k);$$
$$b = 1 - (y + k > 1?1 : y + k);$$

  If you care about color conversions, you should be doing a `within <pic>` loop inside `extra_endfig`. The built-in conversions are more of a fallback.

What you should also know

Metapost uses cairo (http://cairographics.org) to do the bitmap creation, and then uses libpng (http://www.libpng.org) to create the actual file.

Any `prologues` setting is always ignored: the internal equivalent of the `glyph of` operator is used to draw characters onto the bitmap directly.

If there are points in the current picture with negative coordinates, then the whole picture is shifted upwards to prevent things from falling outside the generated bitmap.

Taco Hoekwater

Multiple documents from one source

LaTeX for lecturers and teachers

Abstract
In general LaTeX will produce only one output document. This paradigm shifts when harnessing the power of the so-called shell escape. We will show how to produce multiple output documents with differing content from one single source document. The principle is developed step by step illustrating a typical application in academic teaching.
Focusing on mathematical problems we then explore two ways of automating calculations by integrating free software into the LaTeX run.

Keywords
mathematics, problem sheet, shell escape

The problem with problem sheets

Composing appealing problem sheets and preparing instructive solutions is a time consuming task. Further time is lost when problems need to be altered after typesetting, requiring updated plots and recalculated solutions. When LaTeX is used, we should optimize our workflow and exploit its capabilities to handle as many tedious tasks as possible.

Set up for failure: bad practice

A typical road to ruin is typesetting problems and solutions in separate files. That way it is hard to keep changes in order or notation in sync. For better comprehension we would also like to have the problem formulation above its solution, but copying them from the problem sheet source will just increase the sync problem.

The `exam` document class by Philip Hirschhorn [3] eliminates the need for two source documents. It defines a `solution` environment that can be included or hidden via a class option. The solutions are typeset in the source document right after the problem which makes for a very natural workflow and easier debugging. Changes in notation can now be handled globally using the text editors "replace all" function.

On the other hand, having just one source document requires us to pay close attention: without additional adjustments both versions will compile to the same output file name. In an inattentive moment we might accidentally upload the version with solutions to our website too early, rendering the homework assignments pointless. Also `exam.cls` may not provide the customization you need, for example if you have to implement a department's corporate design or would like to define a third version of the problem sheet containing additional notes.

Getting multiple outputs

Multiple outputs would usually be achieved by multiple invocations of LaTeX. Before each run we would need to manipulate the source document in order to have differing content. All of this can be done using shell scripts or MakeFiles, but it

- requires an additional (script or make) file
- requires non-TeX programs
- requires knowledge beyond TeX
- may not be platform independent

We will introduce a method that handles these shortcomings using only `pdflatex`.

Requirements for a unified workflow

As we have already seen, it is crucial to have just one source document to avoid incongruities and have a natural way of typesetting.

Typeset problems will most certainly be reused in future courses. Therefore we would like to be able to copy & paste parts of the exam with ease. Our approach should also apply to documents that `\input` problems from another source file.

Finally we want to produce multiple outputs differing in content by a single invocation of `pdflatex`. Our derivation will assume the following use cases:

- **student** problems only
- **tutor** problems and their solutions
- **corrector** problems, solutions and instructions on grading the students work

Designing a unified workflow

Our goal is to produce multiple output documents with differing content in a single run. Though this might seem impossible at first glance, Ulrike Fischer found a way [2] to do this using only `pdflatex`. We will introduce her approach step by step, keeping an eye on platform independence.

Since `pdflatex Sheet.tex` will only produce `Sheet.pdf` our task splits into three parts:

1. Find a way to tell LaTeX which parts of the source file to use and which to ignore
2. Change the file name of the output document in order to be able to produce multiple outputs
3. Produce all versions in just one run of `pdflatex`

Selectively in- & excluding code

We would like to wrap code into a LaTeX environment and have some kind of "switch" in the document preamble to control whether to have LaTeX process it or not. This is exactly what `comment.sty` by Victor Eijkhout [1] does.

Usage. The `comment` package provides us with two simple commands

- `\includecomment{foobar}` defines the environment `foobar` whose content will be included
- `\excludecomment{foobar}` defines the environment `foobar` whose content will be ignored

Example. A nice feature of environments defined using `comment.sty` will not break the line, so we can use them in midsentence. The minimal example

```
\includecomment{truth}
\excludecomment{nonsense}

Knuth started
\begin{truth}
developing \TeX{}
\end{truth}
\begin{nonsense}
using WinWord
\end{nonsense}
in 1977.
```

will just output

Knuth started developing TEX in 1977.

Implementing our use cases. Knowing the above, we can easily write a document that matches our use cases, but we would still have to adjust the in- and exclusions before compiling.

To reduce such modifications to a bare minimum we will assign a number to each use case. We can then define a macro `\condition` that expands to the number and use `\ifcase` to make the adjustments for corresponding case.

Listing 1. Use case implementation

```
\RequirePackage{comment}

\includecomment{problem}
\ifcase\condition
  % \condition = 0, student
  \excludecomment{solution}
  \excludecomment{howtograde}
\or % \condition = 1, tutor
  \includecomment{solution}
  \excludecomment{howtograde}
\or % \condition = 2, corrector
  \includecomment{solution}
  \includecomment{howtograde}
\fi
```

Now all it takes is defining the value of `\condition` to control the content of the output.

Coding discipline. One could of course implement the above using `\ifnum` instead. I prefer to use `\ifcase` here because the cases are "automatically numbered" in order of appearance. That way I got less confused about which number represents which use case, saving me time on debugging.

Rethinking command line calls

We want to control the version of the output without editing the source document. We could write a wrapper document that defines our `\condition` macro followed by the actual document code:

```
\gdef\condition{0}
\input Sheet.tex
```

On second thought we can also pass this code on to LaTeX directly on the command line

```
pdflatex "\gdef\condition{0} \input Sheet.tex"
```

avoiding the additional file.

Changing the output file name

By now we can produce any version of the document by altering a single value but they will still be written to the same output file, thereby overwriting the previous output.

A quick look at `pdflatex`'s manpage[1] provides

```
pdflatex --jobname="student" Sheet.tex
```

which will create `student.pdf` from `Sheet.tex`.

Escaping to the Shell

We know that LaTeX can write to auxiliary files using output stream. There is also the special stream 18 which will execute the output on the system shell.

```
\write18{ping tug.org}
```

This is called *escaping* to the shell. Used like this LaTeX will first read to the end of the document before writing to the shell. If we want the command to be executed immediately when LaTeX reaches that point in the document, we use (see [4], p. 226f)

```
\immediate\write18{ping tug.org}
```

In particular this means we can invoke the commands developed in the previous sections from within one `pdflatex` run.

Warning. Giving LaTeX access to the shell is a gateway for exploits. Hence `\write18` is disabled by default. You can temporarily enable it using

```
pdflatex --shell-escape Sheet.tex
```

or permanently by adjusting the configuration[2] of your TeX distribution.

The UniFlow principle

After introducing all the building blocks we are now able to understand Ulrike Fischer's ingenious construction [2] to produce multiple output documents in one single run.

We start off with a document skeleton to demonstrate the recursive nature of the approach.

Listing 2. General UniFlow template

```
% Beginning of Sheet.tex
\ifx\condition\undefined
  % Pseudo shell script (listing 3)
  \expandafter\stop
\fi

% Use case implementation (listing 1)

% Actual document code begins here
```

Processing this code will have `pdflatex` enter the `\ifx` block as the macro `\condition` has not been defined yet. After executing a "pseudo shell script" LaTeX will first expand the token `\fi` and then `\stop` reading. Note that this run will not produce any output.

In the pseudo shell script, we will invoke `pdflatex` again on this very same document. The file's base name is obtained from `\jobname`

```
pdflatex "\string\input\space\jobname"
```

and we told the parser to interpret `\input` as a `\string`, preventing it from expansion ([4], p. 40).

When we add a definition of `\condition`, LaTeX will ignore the `\ifx` block, apply the use case settings and output the desired version.

Considering all use cases and the change of job name we arrive at

Listing 3. Pseudo shell script in LaTeX

```
\immediate\write18{
  pdflatex
  --jobname=\jobname-student
  "\gdef\string\condition{0}
  \string\input\space\jobname"}
\immediate\write18{
  pdflatex
  --jobname=\jobname-tutor
  "\gdef\string\condition{1}
  \string\input\space\jobname"}
\immediate\write18{
  pdflatex
  --jobname=\jobname-corrector
  "\gdef\string\condition{2}
  \string\input\space\jobname"}
```

Combining the UniFlow template (listing 2) with the implementation of the use cases and the corresponding pseudo shell script (listings 1 and 3) we have constructed the single source document `Sheet.tex`. Enabling shell escape and processing it with `pdflatex` will result in the three output documents `Sheet-student.pdf`, `Sheet-tutor.pdf` and `Sheet-corrector.pdf`, each of them with the desired specific content. Therefore all of our initial requirements are met and we have developed a unified workflow.

Pitfall. Neither the wrapping `pdflatex` run nor script 3 will produce an output file named `Sheet.pdf`. This can cause error messages when using text editors with built-in PDF viewers like TeXmaker and its standard "quick build" feature.

Exercise. If you would like to check your understanding of the UniFlow principle, try to write a template for this scenario:

A school teacher always designs two slightly different versions A and B of an exam. She would like to produce the four versions A, B, A with solutions and B with solutions from a single source document.

UniFlow in action

The UniFlow principle can also serve to integrate external programs into the LaTeX run. Due to the author's background the examples are taken from mathematics.

For applications in other subjects see the PythonTeX gallery [5] or Herbert Voß's article on general source code [7].

For the sake of simplicity we now focus on having only one output document. Nevertheless we will still have to define the `\condition` macro (setting it to an arbitrary string value) whenever we want `pdflatex` to ignore the `\ifx` block. The generalization to multiple output versions is left to the reader as an exercise.

Linear Algebra using Sage

Sage is a free and open source computer algebra system. It is best used on Linux since the "Windows version" is actually an Ubuntu virtual machine image containing Sage.

We will give a tiny demonstration of the LaTeX interface called SageTeX and its implementation using the UniFlow principle. Further information on SageTeX can be found in Günter Rau's demonstration [6] or on the Sage homepage[3].

How to compile. SageTeX works similar to BibTeX: First we run LaTeX to extract the Sage commands. These are then processed externally with Sage and the results are included in the second LaTeX run.

```
# Extract Sage commands
pdflatex Example.tex
# Process Sage commands
sage Example.sagetex.sage
# Include Sage outputs
pdflatex Example.tex
```

Implementing UniFlow.

```
\ifx\condition\undefined
  \immediate\write18{
    pdflatex
    "\gdef\string\condition{0}
    \string\input\space\jobname"}
  \immediate\write18{
    sage "\jobname.sagetex.sage"}
  \immediate\write18{
    pdflatex
    "\gdef\string\condition{0}
    \string\input\space\jobname"}
  \expandafter\stop
\fi
```

Exercise. Calculate the eigenvalues of the matrix

$$A = \begin{pmatrix} 19 & 30 & -20 \\ 26 & 39 & -26 \\ 61 & 93 & -62 \end{pmatrix}$$

Solution. The characteristic polynomial of A is

$$\chi\_A(x) = x^3 + 4x^2 + 3x = x \cdot (x+1) \cdot (x+3)$$

hence its eigenvalues are

$$[0, -1, -3]$$

Using `sagetex.sty` we just needed to type

```
% 'sagesilent' returns no output
\begin{sagesilent}
A = matrix(QQ, [[19,30,-20],
    [26,39,-26], [61,93,-62]])
p = A.charpoly()
\end{sagesilent}

\[ A = \sage{A} \]
The characteristic polynomial of
    $A$ is
\[ \chi_A(x) = \sage{p} = \sage{
    factor(p)} \]
hence the eigenvalues of $A$ are
\[ \sage{A.eigenvalues()} \]
```

Note how this assures that the matrix A and the solution will always match in the output document. This is as foolproof as it gets.

Statistics using R and Sweave

Data plotting techniques play an important role in any statistics course: histograms, q-q plots, boxplots etc. are handy tools to analyze measured data.

R is a free statistics software system available for all common operating systems[4]. It comes with the plug-in Sweave which "weaves" R (the free successor to S statistics) into LaTeX documents.

We will use an easy example from elementary probability. More advanced examples can be found for example in Uwe Ziegenhagen's demonstration [8] or on Friedrich Leisch's Sweave website[5].

How to compile. As the output of Sweave will be written to `Example.tex` we change the file name of our document to `Example.Rnw` (Rnw = **R** no**w**eb). Now we can use LaTeX code as usual and insert R code as "chunks" using the noweb syntax. The document is then compiled as follows.

```
# Have R process Example.Rnw and
# create / overwrite Example.tex
R CMD Sweave Example.Rnw
pdflatex Example.tex
```

Implementing UniFlow.

```
% Beginning of Example.Rnw
\ifx\condition\undefined
  \immediate\write18{
    R CMD Sweave \jobname.Rnw}
  \immediate\write18{
    pdflatex
    "\gdef\string\condition{0}
    \string\input\space\jobname"}
  \expandafter\stop
\fi
```

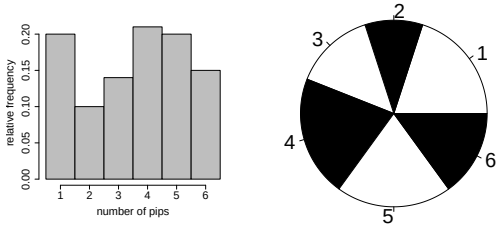
Editing the `Example.Rnw` as source file and using the above code, the correct command line call is

```
pdflatex --shell-escape Example.Rnw
```

If you like to use the tab completion feature of your system shell, it will probably only offer you the `.tex` file. Observe that this will generate the same output because both execute the same pseudo shell script.

Exercise. Roll a dice 100 times in a row recording the number of pips each time. Visualize their relative frequency as a histogram and a pie chart.

Solution. Since this exercise depends on probability, everyone will have a different result. Mine looks like this:



These diagrams where of course generated at compile time from the following code snippet.

```
# Relevant part of Example.Rnw
<< echo=FALSE, fig=TRUE >>=
par(ps=20)
pips <- sample(1:6,100,replace=
    TRUE)
hist(pips, breaks=c(0.5, 1.5,
    2.5, 3.5, 4.5, 5.5, 6.5), col=
    "gray", freq=FALSE, main="",
    xlab="number of pips", ylab="
    relative frequency")
@

<< echo=FALSE, fig=TRUE >>=
pie(table(pips), col=c("white", "
    black"), cex=2)
@
```

Here, due to the use of **sample**(), the output will be different after every compile run.

Aftermath

Of course we could have achieved all of this in a one-call fashion using some kind of shell script, `make`[6] or its LaTeX analogs `latexmk`[7] or `rubber`[8]. On the other hand the UniFlow principle provides a platform independent, script-like alternative without additional (Make)files or non-TeX executables.

The future of UniFlow

To enable anyone to implement the UniFlow principle with ease I will work on developing it into a LaTeX package.

Versatility is UniFlow's biggest asset and every reader will by now have his or her special use case in mind – and most certainly be struggling with the inconvenient syntax of the corresponding `\write18` statement. Hence designing an intuitive command structure will be key and your TeXnical comments and pieces of advice are always welcome.

One step further we could think about a unified interface to integrate virtually any program into the LaTeX run. Herbert Voß [7] already showed how general source code can be extracted from a document and reincluding the output after processing. His approach works with any kind of batching method, allowing for an integration into UniFlow (once developed).

Acknowledgments

Many people have directly or indirectly contributed in the development of the UniFlow principle.

First and foremost I want to thank Ulrike Fischer who provided the core concepts in her short and effective post on StackExchange.

When I was struggling with selectively showing or hiding content, Rolf Niepraschk pointed out the fascinating simplicity of `comment.sty` to me.

Marcus Bitzl is an avid reader of "Die TeXnische Komödie" and drew my attention to the articles on the integration of free math software. He was also the first to encourage the development of a UniFlow package.

The articles by Günter Rau (SageTeX) and Uwe Ziegenhagen (Sweave) were invaluable primers to me and made the vivid demonstrations of UniFlow in action possible.

Finally I would like to express my gratitude to the EuroTeX 2012 organizers for arranging a wonderful and inspiring conference and giving me the opportunity to present the UniFlow principle.

References

[1] V. Eijkhout. comment.sty: Selectively include / excludes portions of text. CTAN:macros/latex/contrib/comment: http://www.ctan.org/tex-archive/macros/latex/contrib/comment.

[2] U. Fischer. Answer to "Can one TeX file output to multiple PDF files?" http://tex.stackexchange.com/a/5265.

[3] P. Hirschhorn. exam.cls: Package for typesetting exam scripts. CTAN:macros/latex/contrib/exam: http://www.ctan.org/tex-archive/macros/latex/contrib/exam.

[4] D. E. Knuth. *The TEXbook*. Addison-Wesley, Eighth printing, August 1986.

[5] G. Poore. PythonTEX: Fast Access to Python from within LaTEX. github:gpoore/pythontex: https://github.com/gpoore/pythontex.

[6] G. Rau. SageTEX. *Die TEXnische Komödie*, 1/2011: http://archiv.dante.de/DTK/PDF/komoedie_2011_1.pdf:17–21.

[7] H. Voß. Einlesen und Ausführen von Quellcode. *Die TEXnische Komödie*, 1/2011: http://archiv.dante.de/DTK/PDF/komoedie_2011_1.pdf:40–54.

[8] U. Ziegenhagen. Datenanalyse mit Sweave, LaTEX und R. *Die TEXnische Komödie*, 4/2010: http://www.dante.de/DTK/Ausgaben/dtk104.pdf:35–45.

Weblinks

1. http://linux.die.net/man/1/pdflatex
2. http://wiki.contextgarden.net/Write18
3. http://www.sagemath.org
4. http://www.r-project.org
5. http://www.statistik.lmu.de/~leisch/Sweave
6. http://www.gnu.org/software/make
7. CTAN:support/latexmk: http://ctan.org/tex-archive/support/latexmk/
8. https://launchpad.net/rubber

Leo Arnold
tex@arney.de

Database publishing with LuaTEX and the speedata Publisher

Database Publishing is the repetitive (semi) automatic transformation from a data source to some kind of output (HTML, PDF, epub,...). A common task is to have an excel sheet, a product information management system or a webshop database and generate reports, data sheets, product catalogs or other kind of PDF documents out of it. Database publishing is often equal to 'InDesign Publishing' with the help of some plugin that automates the task of pulling data from the database into the document. The user can (and must) make the resulting document more beautiful.

There are several alternatives to this approach, especially when you need 100% unattended workflows. Each alternative has advantages and of course drawbacks. 1) ConTEXt fills this gap nicely, but requires a very knowledgable programmer. 2) Many times users write some perl or python scripts that reads the database contents and produces some kind of output, perhaps LaTEX code that must be run with PdfLaTEX. This is a fast approach, but tends to get very hackerish after some time. 3) There is a standardized way of transforming XML to PDF called XSL-FO. This w3c standard has the big advantage that many tools exist to help the user in the task of publishing. But XSL-FO is very limited in its capabilities to produce reasonable documents.

A common demand in high volume output is to optimize page usage. As an example: imagine you have six products in a group but a page only fits five. The software system should be able to re-arrange the products and change a few parameters (image size, text size, text length), so that all six products fit on the same page and thus a whole page saved. The aforementioned systems are either very demanding on the programming side or just not capable of optimizations like these.

The product of our company is filling in this gap. It provides a way to transform XML (and thus any data) to PDF. It has a specialized input language designed for the purpose of laying out elements on a page, and it has all functionality of a modern programming language (variables, loops, if-then-else switches). It can put text and graphical elements on a virtual page that is used for any kind of layout optimization. These virtual pages can be removed and re-typeset with different parameters and only the 'best' page will make it to the PDF. As there is no control language for this kind of application yet, the system is inspired by the standards HTML (table layout), XPath (accessing the data and running specialized functions) and XSLT (accessing document nodes, programming constructs).

The software (called 'speedata Publisher') is written in Lua and makes heavy use of the LuaTEX engine. We use TEX to break paragraphs into lines, arrange the programmatically created boxes and glue for layout of complex tables and to write clean PDF. The publisher is open source software (AGPL) and runs under the three major operating systems (Linux, Windows, Mac OS X). The documentation is mostly still in German, although we are currently translating the documentation into english.

Patrick Gundlach

MetaPost path resolution isolated

Abstract
A new interface in MPLib version 1.800 allows one to resolve path choices programmatically, without the need to go through the MetaPost input language.

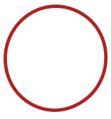
Metapost path solving

As we all know, MetaPost is pretty good at finding pleasing control points for paths. What all of you may know is that besides drawing on a picture, MetaPost can also display the found control points in the log file.

Some illustration at this point is useful. Here is the MetaPost path input source of a very simple path (as well as a visualisation of the path):

```
tracingchoices := 1;
path p;
p := (0,0) ..(10,10) ..(10,-5) ..cycle;
```

And here is what MetaPost outputs in the log file:



```
Path at line 5, before choices:
(0,0)
 ..(10,10)
 ..(10,-5)
 ..cycle

Path at line 5, after choices:
(0,0)..controls (-1.8685,6.35925) and (4.02429,12.14362)
 ..(10,10)..controls (16.85191,7.54208) and (16.9642,-2.22969)
 ..(10,-5)..controls (5.87875,-6.6394) and (1.26079,-4.29094)
 ..cycle
```

A more complex path of course creates more output, so:

```
p := (0,0)..(2,20){curl1}..{curl1}(10, 5)..controls (2,2) and (9,4.5)..
  (3,10)..tension 3 and atleast 4 .. (1, 14){2,0} .. {0,1}(5,-4) ;
```

produces:

```
Path at line 7, before choices:
(0,0){curl 1}
 ..{curl 1}(2,20){curl 1}
 ..{curl 1}(10,5)..controls (2,2) and (9,4.5)
 ..(3,10)..tension 3 and atleast4.5
 ..{4096,0}(1,14){4096,0}
 ..{0,4096}(5,-4)

Path at line 7, after choices:
(0,0)..controls (0.66667,6.66667) and (1.33333,13.33333)
 ..(2,20)..controls (4.66667,15) and (7.33333,10)
 ..(10,5)..controls (2,2) and (9,4.5)
 ..(3,10)..controls (2.34547,10.59998) and (0.48712,14)
 ..(1,14)..controls (13.40117,14) and (5,-35.58354)
 ..(5,-4)
```

But what if ...

But what if you want to use that functionality outside of MetaPost, for instance in a C program?

> You will have to compile MPLib into your program; then create a Metapost language input string; execute it; and parse the log result.

All of that is not very appealing. It would be much better ...

> if you could compile MPLib into your program; create a path programmatically; and then run the Metapost path solver directly; automatically updating the original path.

And that is what the current version of MPLib will allow you to do.

How it works

Once again, it is easiest to show you what to do by using a source code example:

```
#include "mplib.h"

int main (int argc, char ** argv)  {
        MP mp ;
        MP_options * opt = mp_options () ;
        opt -> command_line = NULL;
        opt -> noninteractive = 1 ;
        mp = mp_initialize ( opt ) ;
        my_try_path(mp);
        mp_finish ( mp ) ;
        free(opt);
        return 0;
}
```

Most of the example code above is exactly what one needs to do anything with MPlib programmatically. The only new line is the line calling `my_try_path(mp)`:

```
void my_try_path(MP mp) {
        mp_knot first, p, q;
        first = p = mp_append_knot(mp,NULL,0,0);
        q = mp_append_knot(mp,p,10,10);
        p = mp_append_knot(mp,q,10,-5);
        mp_close_path_cycle(mp, p, first);
        if (mp_solve_path(mp, first)) {
            mp_dump_solved_path(mp, first);
        }
        mp_free_path(mp, first);
}
```

This function uses a new type (`mp_knot`) as well as a bunch of new library functions in MPlib that exist since version 1.800.

- `mp_append_knot()` creates a new knot, appends it to the path that is being built, and returns it as the new tail of the path.
- `mp_close_path_cycle()` mimics `cycle` in the Metapost language.
- `mp_solve_path()` finds the control points of the path. `solve_path` does not alter the state of the used MPlib instance in away, it only modifies its argument path.
- `mp_dump_solved_path()` *user defined function, see below for its definition*
- `mp_free_path()` releases the used memory.

`mp_dump_solved_path` uses even more new functions. First let us look at its definition:

```
#define SHOW(a,b) mp_number_as_double(mp,mp_knot_##b(mp,a))
void mp_dump_solved_path (MP mp, mp_knot h) {
    mp_knot p, q;
    p = h;
    do {
        q = mp_knot_next(mp,p);
        printf ("(%g,%g)..controls (%g,%g) and (%g,%g)",
                SHOW(p,x_coord), SHOW(p,y_coord), SHOW(p,right_x),
                SHOW(p,right_y), SHOW(q,left_x), SHOW(q,left_y));
        p = q;
        if (p != h || mp_knot_left_type(mp,h) != mp_endpoint)
            printf ("\n ..");
    } while (p != h);
    if (mp_knot_left_type(mp,h) != mp_endpoint)
        printf("cycle");
    printf ("\n");
}
```

Somewhat hidden in the source above is that there is another new type: `mp_number`, the data structure representing a numerical value inside MPlib.

The used MPlib library functions are as follows:

- `mp_knot_next()` move to the next knot in the path.
- `mp_knot_x_coord()`, `mp_knot_y_coord()`, `mp_knot_right_x()`, `mp_knot_right_y()`, `mp_knot_left_x()`, `mp_knot_left_y()` return the value of a knot field, as a `mp_number` object (the calls to these functions are hidden inside the definition of the `SHOW` macro).
- `mp_knot_left_type()` returns the type of a knot, normally either `mp_endpoint` or `mp_open`.
- `mp_number_as_double()` converts a `mp_number` to `double`.

To satisfy our curiosity, here is the actual output of the example program listed above:

```
(0,0)..controls (-1.8685,6.35925) and (4.02429,12.1436)
 ..(10,10)..controls (16.8519,7.54208) and (16.9642,-2.22969)
 ..(10,-5)..controls (5.87875,-6.6394) and (1.26079,-4.29094)
 ..cycle
```

And that is almost exactly the same as in the log file:

```
(0,0)..controls (-1.8685,6.35925) and (4.02429,12.14362)
 ..(10,10)..controls (16.85191,7.54208) and (16.9642,-2.22969)
 ..(10,-5)..controls (5.87875,-6.6394) and (1.26079,-4.29094)
 ..cycle
```

The output is not perfectly the same because MetaPost itself does not use `mp_number_as_double` and `%g` for printing the `scaled` values that are by default used to represent numerical values.

The difference is not really relevant, since any programmatic use of the path solver should not have to be 100% compatible with the MetaPost programming language.

More complex paths

Of course there are also new functions to the create more complex paths that make use of `curl`, `tension` and/or `direction` specifiers.

Here is how to encode the second MetaPost path from the earlier example:

```
first = p = mp_append_knot(mp,NULL,0,0);
q = mp_append_knot(mp,p,2,20);
```

```
p = mp_append_knot(mp,q,10,5);
if (!mp_set_knotpair_curls(mp, q,p, 1.0, 1.0))
    exit ( EXIT_FAILURE ) ;
q = mp_append_knot(mp,p,3,10);
if (!mp_set_knotpair_controls(mp, p,q, 2.0, 2.0, 9.0, 4.5))
    exit ( EXIT_FAILURE ) ;
p = mp_append_knot(mp,q,1,14);
if (!mp_set_knotpair_tensions(mp,q,p, 3.0, -4.0))
    exit ( EXIT_FAILURE ) ;
q = mp_append_knot(mp,p,5,-4);
if (!mp_set_knotpair_directions(mp, p,q, 2.0, 0.0, 0.0, 1.0))
   exit ( EXIT_FAILURE ) ;
mp_close_path(mp, q, first);
```

Elaborate documentation for these extra functions (and a few more) is in `mplibapi.tex` which is included in the MetaPost distribution.

Lua interface

There is also a Lua interface for use in LuaTEX, which is a bit higher-level

```
<boolean> success = mp.solve_path(<table> knots, <boolean> cyclic)
```

This modifies the `knots` table (which should contain an array of points in a path, with the substructure explained below) by filling in the control points. The boolean `cyclic` is used to determine whether the path should be the equivalent of `--cycle`. If the return value is `false`, there is an extra return argument containing the error string.

On entry, the individual knot tables can contain the six knot field values mentioned above (but typically the `left_{x,y}` and `right_{x,y}` will be missing). `{x,y}_coord` are both required. Also, some extra values are allowed:

`left_tension`	number	A tension specifier
`right_tension`	number	like `left_tension`
`left_curl`	number	A curl specifier
`right_curl`	number	like `left_curl`
`direction_x`	number	x displacement of a direction specifier
`direction_y`	number	y displacement of a direction specifier

Issues to watch out for

All the 'normal' requirements for MetaPost paths still apply using this new interface. In particular

- A knot has either a direction specifier, or a curl specifier, or a tension specification, or explicit control points, with the additional note that tensions, curls and control points are split in a left and a right side (directions apply to both sides equally).
- The absolute value of a tension specifier should be more than 0.75 and less than 4096.0, with negative values indicating '`atleast`'.
- The absolute value of a direction or curl should be less than 4096.0.
- If a tension, curl, or direction is specified, any existing control points will be replaced by the newly computed value.

Taco Hoekwater

Parsing PDF content streams with LuaTEX

Abstract
The new pdfparser library in LuaTEX allows parsing of external pdf content streams directly from within a LuaTEX document. This paper explains its origin and usage.

Background

Docwolves main product is an infrastructure to facilitate paperless meetings. One part of the functionality is handling meeting documents, and to do so it offers the meeting participants a method to distribute, share, and comment on such documents by means of an intranet application as well as an iPad App.

Meeting documents typically consist of a meeting agenda, followed by included appendices, combined into a single pdf file. Such documents can have various revisions, for example if a change has been made to the agenda or if an appendix has to be added or removed. After such a change, a newly combined pdf document is re-distributed.

Annotations can be made on these documents and these can then be shared with other meeting participants, or just communicated to the server for safe keeping. Like documents, annotations can be updated as well.

All annotations are made on the iPad, with an (implied) author and an intended audience. Annotations apply to a specific part of the source text, and come in a few types (highlight, sticky note, freehand drawing). The iPad App communicates with a network server to synchronize shared annotations between meeting participants.

The annotation update problem

The server-client protocol aims to be as efficient as possible, especially in the case of communication with the iPad app, since bandwidth and connection costs can be an issue.

This means that for any annotation on a referenced document, only the document's internal identification, the (pdf) page number, and the beginning and end word indices on the page and are communicated back and forth. This is quite efficient, but gives rise to the following problem:

> When a document changes, e.g. if an extra meeting item is added, all annotations following that new item have to be updated because their placement is off.

The actual update process is quite complicated, but the issue this paper deals with is that the server software needs to know what words are on any pdf page, as well as their location on that page, and therefore its text extraction process has to agree with the same process on the iPad.

pdf text extraction

Text extraction is a two-step process. The actual drawing of a pdf page is handled by PostScript-style postfix operators. These are contained in objects that are called page content streams.

After decompression of the pdf, the beginning of a content stream could look like this:

```
59 0 obj
<< /Length 4013 >>
stream
0 g 0 G
1 g 1 G
q
0 0 597.7584 448.3188 re f
Q
0 g 0 G
1 0 0 1 54.7979 44.8344 cm
...
```

Here `g`, `G`, `q`, `re`, `f`, `Q`, and `cm` are all (postfix) operators, and the numeric quantities are all arguments. As you see, not all operators take the same amount of arguments (`g` takes one, `q` zero, and `re` four). Other operators may take for instance string-valued arguments instead of numeric ones. There are a little over half a dozen different types.

To process such a stream easily, it is best to separate the task (at least conceptually) into two separate tasks. First there is a the lexing stage, which entails converting the actual bytes into combinations of values and types (tokens) that can be acted upon.

Separate from that, there is the interpretation

stage, where the operators are actually executed with the tokenized arguments that have preceded it.

pdf text extraction on the iPad
It is very easy on an iPad to display a representation of a pdf page, and Apple also provides a convenient interface to do the actual lexing of pdf content streams that is the first step in getting the text from the page. But to find out where the actual pdf objects are, one has to interpret the pdf document stream oneself, and that is the harder part of the text extraction operation.

pdf text extraction on the server
On the server side, there is a similar problem at a different stage: displaying a pdf is easy, and even literal text extraction is easy (with tools like `pdftotext`). However, that does not give you the location of the text on the page. On the server, Apple's lexing interface is not available, and the available pdf library (`libpoppler`) does not offer similar functionality.

Our solution

We needed to write text extraction software that can be used on both platforms, to ensure that the same releases of server and iPad software always agreed perfectly on the what and where of the text on the pdf page.

Both platforms use a stream interpreter written by ourselves in C, with the iPad sofware starting from the Apple lexer, and the server software starting from a new lexer written from scratch.

The prototype and first version of the newly created stream interpreter as well as the server-side lexer were written in Lua. LuaTEX's `epdf` libpoppler bindings to Lua were a very handy tool at that stage (see below). The code was later converted back to C for compilation into a server-side helper application as well as the iPad App, but originally it was written als a TEXLua script.

A side effect of this development process is that the lexer could be offered as a new LuaTEX extension, and that is exactly what we did.

About the 'epdf' library

This library is written by Hartmut Henkel, and it provides Lua access to the `poppler` library included in LuaTEX. For instance, it is used by ConTEXt for keeping links in external pdf figures.

The library is fairly extensive, but a bit low-level, because it closely mimics the `libpoppler` interface. It is fully documented in the LuaTEX reference manual, but here is a small example that extracts the page cropbox information from a pdf document:

```
local function run (filename)
   local doc = epdf.open(filename)
   local cat = doc:getCatalog()
   local numpages = doc:getNumPages()
   local pagenum  = 1
   print ('Pages: ' .. numpages)
   while pagenum <= numpages do
      local page = cat:getPage(pagenum)
      local cbox = page:getCropBox()
      print (string.format(
           'Page %d: [%g %g %g %g]',
           pagenum, cbox.x1, cbox.y1,
           cbox.x2, cbox.y2))
      pagenum = pagenum + 1
   end
end
run(arg[1])
```

Lexing via poppler

As said, a lexer converts bytes in the input text stream into tokens, and such tokens have types and values. `libpoppler` provides a way to get one byte from a stream using the `getChar()` method, and it also applies any stream filters beforehand, but it does not create actual tokens.

Poppler limitations
There is no way to get the full text of a stream immediately, it has to be read byte by byte.

Also, if the page content consists of an array of content streams instead of a single entry, the separate content streams have to be manually concatenated.

And content streams have to be 'reset' before the first use.

Here is a bit of example code for reading a stream, using the `epdf` library:

```
function parsestream(stream)
  local self = { streams = {} }
  local thetype = type(stream)
  if thetype == 'userdata' then
    self.stream = stream:getStream()
  elseif thetype == 'table' then
    for i,v in ipairs(stream) do
      self.streams[i] = v:getStream()
    end
    self.stream = table.remove(
                      self.streams,1)
  end
  self.stream:reset()
  local byte = getChar(self)
  while byte >= 0 do
    ...
    byte = getChar(self)
  end
```

```
  if self.stream then
    self.stream:close()
  end
end
```

In the code above, any interesting things you want to happen have to inserted at the ... line. The example makes use of one helper function (getChar) and that looks like this:

```
local function getChar(self)
  local i = self.stream:getChar()
  if (i<0) and (#self.streams>0) then
    self.stream:close()
    self.stream = table.remove(
                    self.streams, 1)
    self.stream:reset()
    i = getChar(self)
  end
  return i
end
```

Our own lexer: 'pdfscanner'

The new lexer we wrote does create actual tokens. Its Lua interface accepts either a poppler stream, or an array of such streams. It puts pdf operands on an internal stack and then executes user-selected operators.

The library pdfscanner has only one function, scan(). Usage looks like this:

```
require 'pdfscanner'
function scanPage(page)
  local stream = page:getContents()
  local ops = createOperatorTable()
  local info = createParserState()
  if stream then
    if stream:isStream()
       or stream:isArray() then
      pdfscanner.scan(stream, ops,
                      info)
    end
  end
end
```

The functions createOperatorTable() and createParserState() are helper functions that create arguments of the proper types.

The scan() function

As you can see, the function takes three arguments:

The first argument should be either a pdf stream object, or a pdf array of pdf stream objects (those options comprise the possible return values of <Page>:getContents() and <Object>:getStream() in the epdf library).

The second argument should be a Lua table where the keys are pdf operator name strings and the values are Lua functions (defined by you) that are used to process those operators. The functions are called whenever the scanner finds one of these pdf operators in the content stream(s).

Here is a possible definition of the helper function createOperatorTable():

```
function createOperatorTable()
  local ops = {}
  -- handlecm is listed below
  ops['cm'] = handlecm
  return ops
end
```

The third argument is a Lua variable that is passed on to provide context for the processing functions. This is needed to keep track of the state of the pdf page since pdf operators, and especially those that change the graphics state, can be nested.[1]

In its simplest form, its creation looks like this:

```
function createParserState()
  local stack = {}
  stack[1] = {}
  stack[1].ctm =
    AffineTransformIdentity()
  return stack
end
```

Internally, pdfscanner.scan() loops over the input stream content bytes, creating tokens and collecting operands on an internal stack until it finds a pdf operator. If that pdf operator's name exists in the given operator table, then the associated Lua function is executed. After that function has run (or when there is no function to execute) the internal operand stack is cleared in preparation for the next operator, and processing continues.

The processing functions are called with two arguments: the scanner object itself, and the info table that was passed are the third argument to pdfscanner.scan.

The scanner argument to the processing functions is needed because it offers various methods to get the actual operands from the internal operand stack.

1. In Lua this could actually have been handled by upvalues or global variables. The third argument was initially a concession made to the planned conversion to C.

Extracting tokens from the scanner
The most low-level function in scanner is scanner:pop() which pops the top operand of the internal stack, and returns a lua table where the object at index one is a string representing the type of the operand, and object two is its value.

The list of possible operand types and associated lua value types is:

integer	<number>
real	<number>
boolean	<boolean>
name	<string>
operator	<string>
string	<string>
array	<table>
dict	<table>

In case of integer or real, the value is always a Lua (floating point) number.

In case of name, the leading slash is always stripped.

In case of string, please bear in mind that pdf actually supports different types of strings (with different encodings) in different parts of the pdf document, so you may need to reencode some of the results; pdfscanner always outputs the byte stream without reencoding anything. pdfscanner does not differentiate between literal strings and hexidecimal strings (the hexadecimal values are decoded), and it treats the stream data for inline images as a string that is the single operand for EI.

In case of array, the table content is a list of pop return values.

In case of dict, the table keys are pdf name strings and the values are pop return values.

While parsing a pdf document that is known to be valid, one usually knows beforehand what the types of the arguments will be. For that reason, there are few more scanner methods defined:

- popNumber() takes a number object off the operand stack.
- popString() takes a string object off the operand stack.
- popName() takes a name object off the operand stack.
- popArray() takes an array object off the operand stack.
- popDict() takes a dictionary object off the operand stack.
- popBool() takes a boolean object off the operand stack.

A simple operator function could therefore look like this (The Affine... functions are left as an exercise to the reader):

```
function handlecm (scanner, info)
  local ty = scanner:popNumber()
  local tx = scanner:popNumber()
  local d  = scanner:popNumber()
  local c  = scanner:popNumber()
  local b  = scanner:popNumber()
  local a  = scanner:popNumber()
  local t =
    AffineTransformMake(a,b,c,d,tx,ty)
  local stack = info.stack
  local state = stack[#stack]
  state.ctm =
    AffineTransformConcat(state.ctm,t)
end
```

Finally, there is also the scanner:done() function which allows you to abort processing of a stream once you have learned everything you want to learn. This comes in handy while parsing /ToUnicode, because there usually is trailing garbage that you are not interested in. Without done, processing only ends at the end of the stream, wasting CPU cycles.

Summary

The new pdfparser library in LuaTEX allows parsing of external pdf content streams directly from within a LuaTEX document. While this paper explained its usage, the formal documentation of the new library is the LuaTEX reference manual. Happy LuaTEX-ing!

Taco Hoekwater
Docwolves B.V.

MFLua: Instrumentation of MF with Lua

Abstract
We present MFLua, a METAFONT version which is capable of code instrumentation and has an embedded Lua interpreter that allows glyphs curves extraction and post-processing. We also show and discuss an example of a METAFONT source processed by MFLua to output an OpenType font.

1 Introduction

MFLua is a version of METAFONT, Knuth's program (Knuth, 1986b) designed to draw fonts. MFLua has an embedded Lua interpreter, as well as the capability of the Pascal-WEB code instrumentation to output information about bitmaps and curves used in glyphs drawing. The latest capability is known as *code tracing*. MFLua's main goal is to ease the production of vector fonts which source code is a METAFONT code. MFLua doesn't extend the METAFONT language in any way (i.e., it's not possible to embed Lua code in a METAFONT source file), so that a METAFONT source file is fully compatible with MFLua and vice versa. MFLua won't be extended like LuaTEX extends pdfLaTEX. The code instrumentation is a facility to gather and manage information collected in the log file when METAFONT `tracing` instructions are enabled. MFLua automatically saves data into Lua tables using external Lua scripts. Therefore a programmer can manage these tables according to his needs, i.e. extracting a glyph vector outline(s). Rephrasing the previous statements, MFLua is a (bitmap) *tracing* program that knows curves in advance instead of determining them from the bitmap. Please notice that this is only possible when the data have been gathered.

The paper has the following structure: after explaining what *code instrumentation* is (section 2), it shows the components used to trace a glyph (section 3) and finally two different approaches to manage curves (section 4).

As a final remark, we consider MFLua as being in a state between a proof of concept and alpha and it's not (yet) too user-friendly. Its code is hosted at `https://github.com/luigiScarso/mflua`.

2 Code instrumentation

METAFONT is written in Pascal-WEB (a programming language by Donald Knuth to have a real literate programming tool. As the name suggests, it's a subset of Pascal) and is automatically translated into C by `tangle` and `web2c`. Instrumentation is the capability to add *trace statements* (a.k.a. *sensors*) in strategic points of the code to register current state information and pass it to the Lua interpreter. A typical sensor has the `mflua` prefix. We can see some sensors in the following chunk of code, the main body of METAFONT (slightly refolded to fit the printing area).

```
@p begin @!{|start_here|}
mflua_begin_program;
{in case we quit during initialization}
history:=fatal_error_stop;
t_open_out; {open the terminal for output}
if ready_already=314159 then goto start_of_MF;
@<Check the ``constant'' values...@>@;
if bad>0 then
  begin wterm_ln('Ouch---my internal constants
     have been clobbered!',
    '---case ',bad:1);
@.Ouch...clobbered@>
  goto final_end;
  end;
{set global variables to their starting values}
initialize;
@!init if not get_strings_started then
  goto final_end;
init_tab; {initialize the tables}
init_prim; {call |primitive| for each primitive}
init_str_ptr:=str_ptr; init_pool_ptr:=pool_ptr;@/
max_str_ptr:=str_ptr; max_pool_ptr:=pool_ptr;
fix_date_and_time;
tini@/
ready_already:=314159;
mfluaPRE_start_of_MF;
start_of_MF: @<Initialize the output routines@>;
@<Get the first line of input and prepare
  to start@>;
history:=spotless; {ready to go!}
mflua_initialize;
if start_sym>0 then
  {insert the `\&{everyjob}' symbol}
  begin cur_sym:=start_sym; back_input;
  end;
mfluaPRE_main_control;
main_control; {come to life}
mfluaPOST_main_control;
final_cleanup; {prepare for death}
mfluaPOST_final_cleanup;
end_of_MF: close_files_and_terminate;
final_end: ready_already:=0;
end.
```

We're going to examine the role of the `mflua_begin_program` sensor. The Pascal-into-C translator, `web2c`, is smart enough to distinguish a symbol already present in the Pascal source from an external symbol (i.e., a symbol defined in another file). In the latter case, the programmer has to register that symbol into the file `texmf.defines` if the symbol is related to an argumented procedure: the translator will manage properly the arguments translation. The translated code will contain the C form of the sensor symbol, which will be resolved at compile-time — i.e., we need an object file that contains that symbol. Each sensor is stored into `mflua.h` and `mflua.c`. The first one lists the symbol:

```
#include "lua51/lua.h"
#include "lua51/lualib.h"
#include "lua51/lauxlib.h"
#include <kpathsea/c-proto.h>
#include <web2c/config.h>

extern lua_State* Luas[];
extern int mfluabeginprogram();
```

while the second one contains the corresponding function source code:

```
int mfluabeginprogram()
{
  lua_State *L = luaL_newstate();
  luaL_openlibs(L);
  Luas[0] = L;
   /* execute Lua external "begin_program.lua" */
  const char* file = "begin_program.lua";
  int res = luaL_loadfile(L, file);
  if ( res==0 ) {
      res = lua_pcall(L, 0, 0, 0);
    }
  priv_lua_reporterrors(L, res);
  return 0;
}
```

The above function initializes the Lua interpreter, stores its state in the array `Luas[]` (it would be possible to have more than one interpreter but this feature is currently neglected) and then executes the external script `begin_program.lua` calling `lua_pcall(L, 0, 0, 0)`. This call protects the interpreter from errors. Every time we run `mf` (the METAFONT program), it loads and executes the Lua script `begin_program.lua`, customizable by programmers.

We surely need to pay attention to some issues. The first one is that literate programming style allows to collect every changes we make in a source file into a *change* file (`mf.ch` in our case), which is then merged into a Pascal program by `tangle`. This means that inserting a sensor can interfere with the change file. In this case we also have to insert the sensor into the change file as we do, for instance, with `mfluaPRE_make_ellipse(major_axis, minor_axis,theta,tx,ty,0)`. Of course the right solution is directly inserting the sensors in the change file. Unfortunately it's usually faster discovering where to insert a sensor in the source file and then managing conflicts in the change file: source files have a context — the source itself — that change file don't. The second issue is the need to export some METAFONT variables and constants to the Lua interpreter. An easy way to accomplish this task is inspecting the translated C code to realize how those variables and constants are managed. For example, to make Lua read the `charcode` variable, which contains the index of the current glyph, we need to know that it's stored into the `internal` array at index 18 (the index is from the METAFONT WEB source) so that we can write a wrapper function like the following one:

```
#define roundunscaled(i) (((i>>15)+1)>>1)
EXTERN scaled internal[maxinternal + 1]  ;
static int
 priv_mfweb_LUAGLOBALGET_char_code(lua_State *L)
{
  integer char_code=18;
  integer p=roundunscaled(internal[char_code])%256;
  lua_pushnumber(L,p);
  return 1;
}
```

and then make it available to the Lua interpreter as the `LUAGLOBALGET_char_code` variable:

```
int mfluainitialize()
{
  /* execute Lua external "mfluaini.lua" */
  lua_State *L = Luas[0];
  /* register lua functions */
:
lua_pushcfunction(L,
       priv_mfweb_LUAGLOBALGET_char_code);
lua_setglobal(L,"LUAGLOBALGET_char_code");
:

/* execute Lua external "mfluaini.lua" */
  const char* file = "mfluaini.lua";
  int res = luaL_loadfile(L, file);
  if ( res==0 ) {
      res = lua_pcall(L, 0, 0, 0);
  }
  priv_lua_reporterrors(L, res);
  return 0;
}
```

Users can read and set this variable though the set value won't be passed to METAFONT in order to interfere as little as possible with its state. That's why we prefer to inspect the translated C code, which quality depends on the translation performed at compile-time. A clean solution should only depend on the WEB source. For historical reasons, translating code from Pascal into C outputs two files (`mf0.c` and `mf1.c`). Finding a symbol implies searching in two files, which hardens the process.

There are currently 24 sensors, 33 global variables and 15 scripts, though it's possible to increase these quantities if we discover that tracing a specific function inside METAFONT is better than reimplementing it in Lua. While it's easy to implement the algorithm to draw a curve in Lua, it's slightly harder to implement the algorithm to fill a region. Whatever, the main goal is to keep the number of sensors as low as possible. Notice that MFLua currently reads scripts from the current folder and doesn't use the standard TEX folders.

The counterpart of `mflua_begin_program` is `mflua_end_program`, which calls the `end_program.lua` script. It contains all the functions used to transform the components of a glyph, the subject of the next section.

3 The components of a glyph

METAFONT mainly manages Bézier cubic curves (see fig. 1).[1] This curve is completely described by four *controls points*: $\mathbf{p}$ (called the *starting point*), $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{q}$ (known as the *ending point*). The METAFONT command to draw such a curve is

```
draw p .. controls c1 and c2 .. q;
```

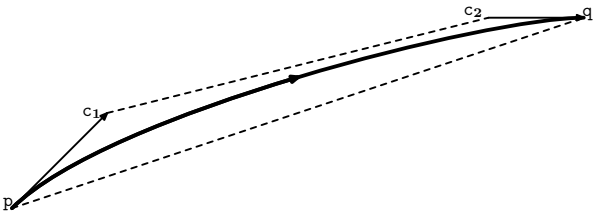


Figure 1. A cubic Bézier curve and its convex hull.

This curve lies on the XY plane and its *parametric form* is quite simple:

$$\mathbf{B}(t) = (1-t)^3\mathbf{p} + 3(1-t)^2 t\mathbf{c_1} + 3(1-t)t^2\mathbf{c_3} + t^3\mathbf{q} \qquad \forall t \in [0,1] \quad (1)$$

The corrensponding algebraic expression, the *closed form*, is more complex but it's useful to quickly test whether a point belongs to the curve or not.

Equation (1) has first derivatives $\mathbf{B}'(0) = 3(\mathbf{c_1} - \mathbf{p})$ and $\mathbf{B}'(1) = 3(\mathbf{q} - \mathbf{c_2})$ respectively when $t = 0$ and $t = 1$. We can easily calculate them when we know $\mathbf{p}$, $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{q}$. An important property is that a Bézier cubic curve is completely contained in the polygon $\mathbf{p}\,\mathbf{c_1}\,\mathbf{c_2}\,\mathbf{q}\,\mathbf{p}$ (the convex hull) and this immediately leads to the conclusion that the intersection of two curves is empty if and only if the intersection of their convex hulls is empty. Another important property is the existence of the *De Casteljau's algorithm*, very easy to implement: given the four control points of a curve and a time t_1, it returns the point $(x_1, y_1) = \mathbf{B}(t_1)$ on the curve and the two series of control points, one for the curve $\mathbf{B_l}(t) = \mathbf{B}(t), t \in [0, t_1]$ (the *left side*) and one for the curve $\mathbf{B_r}(t) = \mathbf{B}(t), t \in [t_1, 1]$ (the *right side*). It recursively reduces the curve $\mathbf{B}(t)$, $t \in [0,1]$ tracing to the tracing of the left side $\mathbf{B_l}(t) = B(t)$, $t \in [0, 1/2]$ and the right side $\mathbf{B_r}(t) = B(t)$, $t \in [1/2, 1]$ (the recursion ends when the distance between two points (x_j, y_j) and (x_{j+1}, y_{j+1}) is less than a pixel).

The De Casteljau's algorithm is useful because it estimates how long a curve is counting the number of pixels covered by the curve. It also finds intersections of two curves $\mathbf{B}(t)$ and $\mathbf{C}(t)$ reducing this problem to the problem of calculating the intersection of four curves: left and right side of $\mathbf{B}(t)$ and left and right side of $\mathbf{C}(t)$ for $t = 1/2$. The algorithm keeps working when one curve degenerates into a segment (i.e., if $\mathbf{p} = \mathbf{c_1}$ and $\mathbf{c_2} = \mathbf{q}$) or when it degenerates into a point ($\mathbf{p} = \mathbf{c_1} = \mathbf{c_2} = \mathbf{q}$). Therefore it can be used to find an intersection between a line and a curve and to test if a point belongs to a curve. A set of curves $\{\mathbf{B}_1, \mathbf{B}_2 \ldots \mathbf{B}_n\}$ where $\mathbf{q}_{j-1} = \mathbf{p}_j$ and $\mathbf{q}_n = \mathbf{p}_0$ is a simple cycle if the only intersection is $(x, y) = \mathbf{q}_n = \mathbf{p}_0$. Simple cycles are the building blocks of a glyph: a simple cycle can be filled or unfilled and, according to METAFONT's point of view, a glyph is a set of cycles filled and/or unfilled at the right moment.

A normal METAFONT designer doesn't care about these details because METAFONT has a high level language to describe curves, points, lines, intersections, filled and unfilled cycles and, most important, pens. The listed entities produce a combination of two different basic draws: regions (un)filled by a *contour* and regions (un)filled by the stroke of a pen, i.e., the *envelope of a pen*. Both are simple cycles, but their origin is very different.

Let's consider the code of the glyph 0 from the file `xmssdc10.mf`:

```
cmchar "The numeral 0";
beginchar("0",9u#,fig_height#,0);
italcorr fig_height#*slant-.5u#;
adjust_fit(0,0);
penpos1(vair,90);
penpos3(vair,-90);
penpos2(curve,180);
penpos4(curve,0);
if not monospace:
 interim superness:=sqrt(more_super*hein_super);
fi
x2r=hround max(.7u,1.45u-.5curve);
x4r=w-x2r; x1=x3=.5w;
y1r=h+o; y3r=-o;
y2=y4=.5h-vair_corr;
y2l:=y4l:=.52h;
penstroke pulled_arc.e(1,2) & pulled_arc.e(2,3)
 & pulled_arc.e(3,4)
 & pulled_arc.e(4,1) & cycle;  % bowl
penlabels(1,2,3,4);
endchar;
```

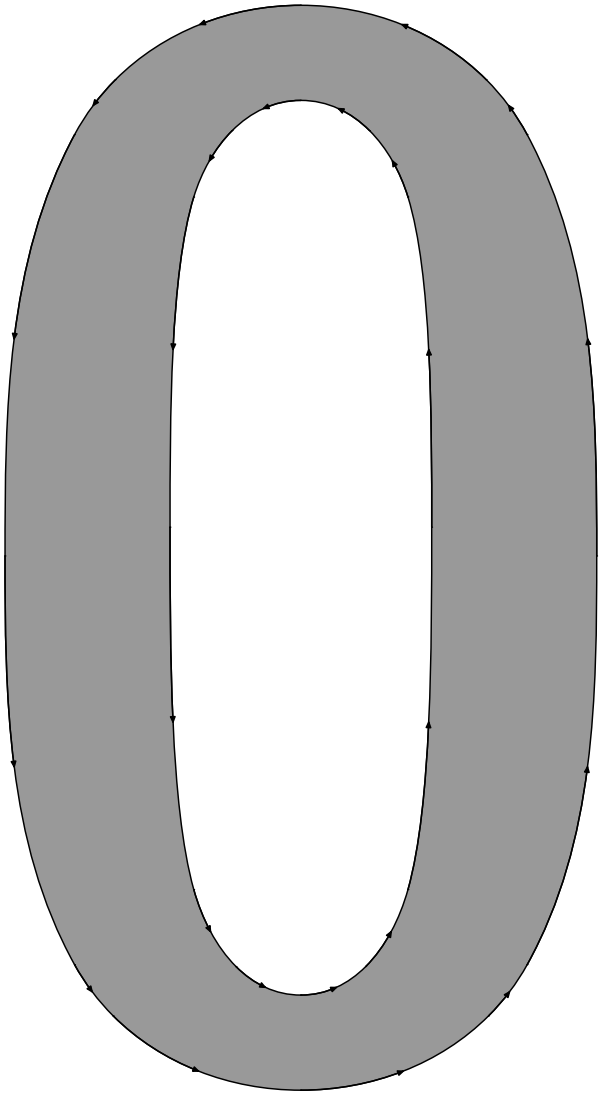


Figure 2. The glyph of the numeral 0 in `xmssdc10.mf` font.

Fig. 2 shows a glyph only made by two contours which are the result of `penpos` and `penstroke` macros. Of course we could obtain the same result drawing 24 curve sections (12 for the outer contour, 12 for the inner one) but it should be clear that the METAFONT description is much more straight or, at least, "typographic".

Things completely change when we consider the numeral 2:

```
cmchar "The numeral 2";
beginchar("2",9u#,fig_height#,0);
italcorr fig_height#*slant-.5u#;
adjust_fit(0,0);
numeric arm_thickness, hair_vair;
hair_vair=.25[vair,hair];
arm_thickness=
Vround(if hefty:slab+2stem_corr
       else:.4[stem,cap_stem] fi);
pickup crisp.nib;
pos7(arm_thickness,-90); pos8(hair,0);
bot y7r=0; lft x7=hround .9u; rt x8r=hround(w-.9u);
y8=good.y(y7l+beak/2)+eps;
arm(7,8,a,.3beak_darkness,beak_jut);%arm and beak
pickup fine.nib; pos2(slab,90);
pos3(.4[curve,cap_curve],0);
top y2r=h+o; x2=.5(w-.5u);
rt x3r=hround(w-.9u); y3+.5vair=.75h;
if serifs:
 numeric bulb_diam;
 bulb_diam=hround(flare+2/3(cap_stem-stem));
 pos0(bulb_diam,180); pos1(cap_hair,180);
 lft x1r=hround .9u; y1-.5bulb_diam=2/3h;
 (x,y2l)=whatever[z1l,z2r];
 x2l:=x; bulb(2,1,0);  % bulb and arc
else: x2l:=x2l-.25u; pos1(flare,angle(-9u,h));
 lft x1r=hround .75u; bot y1l=vround .7h;
 y1r:=good.y y1r+eps; x1l:=good.x x1l;
 filldraw stroke term.e(2,1,left,.9,4);
fi  % terminal and arc
pos4(.25[hair_vair,cap_stem],0);
pos5(hair_vair,0);
pos6(hair_vair,0);
y5=arm_thickness; y4=.3[y5,y3];
top y6=min(y5,slab,top y7l);
lft x6l=crisp.lft x7;
z4l=whatever[z6l,(x3l,bot .58h)];
z5l=whatever[z6l,z4l];
erase fill z4l--
 z6l--lft z6l--
 (lft x6l,y4l)--cycle;%erase excess at left
filldraw stroke z2e{right}..tension
  atleast .9 and atleast 1
 ..z3e{down}..{z5e-z4e}z4e--z5e--z6e;%stroke
penlabels(0,1,2,3,4,5,6,7,8);
endchar;
```

As we can see in fig. 3, there are both a contour and envelopes of more than a pen; there are intersections between the contour the envelopes and the pens, and some curves are outside the glyph (some of these curves are used to delete unwanted black pixels). There are also some unexpected straight lines and small curves. The number of curves looks quite large, which is not what we desire as we want to obtain the outline depicted in fig. 4.

Unfortunately, things are even different and it's necessary to describe how METAFONT calculates pen envelopes to go on. This is explained in the book "METAFONT: The Program" (KNUTH, 1986a) at the "Polygonal pens" part, chapter 469, that we briefly quote with a slightly modified notation:

> Given a convex polygon with vertices $\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{w}_n = \mathbf{w}_0$ a in *counterclockwise* order ...(and a curve $\mathbf{B}(t)$) the envelope is obtained if we offset $\mathbf{B}(t)$ by $\mathbf{w}_k$ when the curve is travelling in a direction $\mathbf{B}'(t)$ ly-

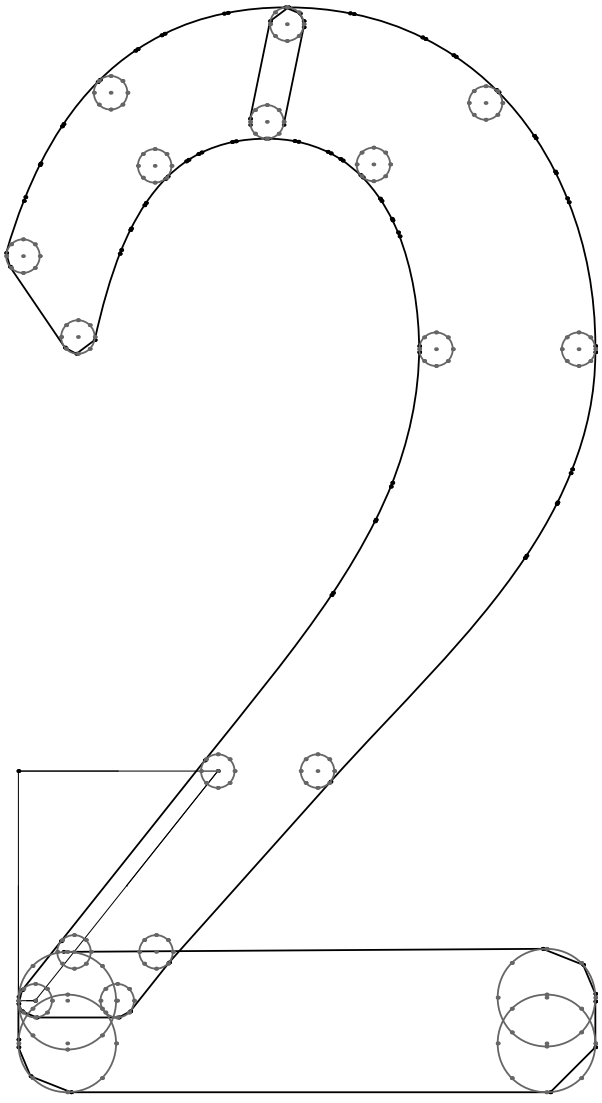


Figure 3. The glyph of the numeral 2 in `xmssdc10` font. We can see envelopes and pens (thick curves) and a contour (thin curve).

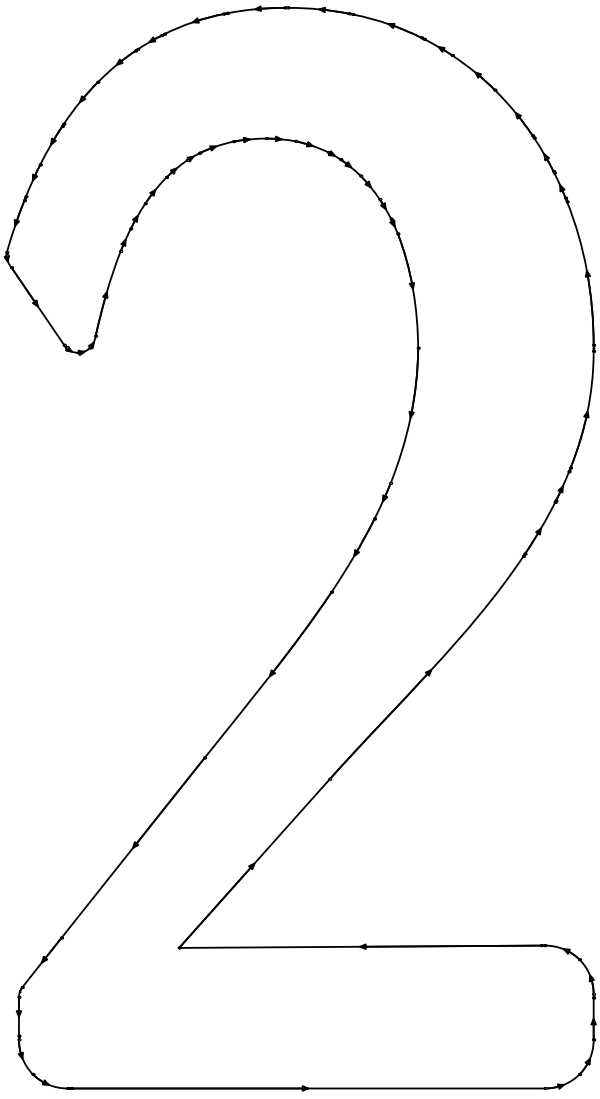


Figure 4. An outline of the numeral 2 in `xmssdc10.mf` font.

ing between the directions $\mathbf{w}_k - \mathbf{w}_{k-1}$ and $\mathbf{w}_{k+1} - \mathbf{w}_k$. At times t when the curve direction $\mathbf{B}'(t)$ increases past $\mathbf{w}_{k+1} - \mathbf{w}_k$, we temporarily stop plotting the offset curve and we insert a straight line from $\mathbf{B}(t) + \mathbf{w}_k$ to $\mathbf{B}(t) + \mathbf{w}_{k+1}$; notice that this straight line is tangent to the to the offset curve. Similarly, when the curve direction decreases past $\mathbf{w}_k - \mathbf{w}_{k-1}$, we stop plotting and insert a straight line from $\mathbf{B}(t) + \mathbf{w}_k$ to $\mathbf{B}(t) + \mathbf{w}_{k+1}$; the latter line is actually a "retrograde" step which will not be part of the final envelope under the METAFONT's assumptions. The result of this construction is a continuous path that consist of alternating curves and straight line segments.

This explains why the number of the curves is large and why there are small curves, but says nothing about those circular curves that we can see in fig. 4: METAFONT indeed converts an elliptical pen into a polygonal one and then applies the algorithm. The conversion is accurate enough to guarantee that the envelope is correctly filled with the right pixels. This is a key point to understand: *METAFONT's main task is to produce the best bitmap of a glyph, not the best outline.*

The role of the sensors is to gather as much information as possible about pixels, contours, the polygonal

version of the pens, envelopes and their straight lines and then store these information (basically the edge structure of the pixels and Bézier curves with an eventual offset) into appropriate Lua tables. As METAFONT halts, the Lua interpreter calls `end_program.lua` and let the programmer manage these tables: sometimes, as we have seen in the numeral 0 case, the post-process can be quite simple, sometimes not. MFLua doesn't automatically output a glyph outline because it's the programmer who has to implement the best strategy according to his experience.

4 Two different strategies for post-processing the curves

The Concrete Roman 10 pt

The first use of MFLua has been the post-processing of Concrete Roman 10 pt to obtain an OpenType version of it. This font is described in the file `ccr10.mf`. As we previously said, sensors collect the data into Lua tables and `end_program.lua` post-processes them at the end of the execution (we could even choose to execute the no-more post-process during the execution). The script `end_program.lua` defines the global array `chartable[index]` that contains the data for the glyph with char code `index`: we have the edge structure that allows the program to calculate the pixels of the glyph as well as the three arrays `valid_curves_c`, `valid_curves_e` and `valid_curves_p` that gather the data of contours, envelopes and the polygonal version of the pens. Each array contains the array of the control points `{p,c1,c2,q}` stored as a string `"(<x>,<y>)"`, where <x> and <y> are the coordinates of the point. With fig. 3 as a reference, we can see that when we draw a glyph with a pen it usually has overlapping strokes. Along with the curves of the pen(s), these overlaps create curves inside or outside the glyph that must be deleted. Having the pixels of the glyph, we can use the parametric form (1) to check if a point (x, y) (or better, a neighborhood with center (x, y)) is inside or outside. If all the points of the curve are inside or outside, we can delete them. The drawback is that while time t goes linearly in $\mathbf{B}(t)$, the points $(x(t), y(t))$ follow a cubic (i.e., not linear) law in case the curve is not a straight line. Hence, they are not equally spaced — this means that we can jump over some critical points. Using the same time interval steps for each curve means that short curves are evaluated in times where the points can differ less than a pixel — a useless evaluation. Of course not all the curves are inside the glyph: there are curves on the border and curves partially lying on the border and partially inside (or outside). In the latter case the result of evaluation is an array of time intervals where the curve crosses the border.

Once we have deleted the curves that are completely inside (or outside) the glyph, the next step is to merge all the curves and split them using the previously seen time interval (this is done by a Lua implementation of the De Casteljau's algorithm). Now we have a set of curves that are on the border or "near" it (i.e., partially on the border). We can delete those curves having only one intersection (a *pending curve*), supposing that each curve of the final outline has exactly 2 intersections at times $t_0 = 0$ and $t_1 = 1$.

To calculate all the intersections we use the following trick: if we have n curves, we produce a METAFONT file that contains the code that calculates the intersection between p_i and p_j for $1 \leq j \leq n$ and $j < i \leq n$ (given that $p_j \cap p_i = p_i \cap p_j$) and then we parse the log file with Lua. For example if

```
p1={"(57.401,351.877)", "(57.401,351.877)",
    "(57.901,349.877)", "(57.901,349.877)"}
```

and

```
p2={"(56.834,356.5)",   "(56.834,354.905)",
    "(57.031,353.356)", "(57.401,351.877)"}
```

then we have

```
batchmode;
message "BEGIN i=2,j=1";
path p[];
p1:=(57.401,351.877) ..
 controls (57.401,351.877) and (57.901,349.877) ..
   (57.901,349.877);
p2:=(56.834,356.5) ..
 controls (56.834,354.905) and (57.031,353.356) ..
    (57.401,351.877);
numeric t,u;
(t,u) = p1 intersectiontimes p2;
show t,u;
message "" ;
```

and the log

```
BEGIN i=2,j=1
>> 0
>> 0.99998
```

If the result is $(-1, -1)$ the intersection is empty. There are two problems with this approach: the first one shows when a curve crosses the border and time intervals can generate two curves, one completely outside and one completely inside — hence deleting an intersection. To avoid this issue we must adjust the intervals moving the extremes a bit. We have the second problem when there can be curves with three or more intersections — i.e., we can have *loops*. Opening a loop can be a difficult task: e.g., if the curve p_a intersects $\{p_b, p_c, p_d\}$ at the same time t_a and p_b intersects $\{p_a, p_c, p_d\}$ at t_b then $I_a = \{p_a\} \cup \{p_b, p_c, p_d\}$ is equal to $I_b = \{p_b\} \cup \{p_a, p_c, p_d\}$ and we can delete p_a and p_b because p_c and p_d stay connected. But with more than three intersections things become more complex.

ff fi fl ffi ffl Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam nisl urna, eleifend vel mollis quis, facilisis vel dolor. Sed auctor nibh eu magna vulputate vulputate. Curabitur ante mauris, pretium eu laoreet at, venenatis et neque. Vestibulum ante quam, tristique in posuere eu, pulvinar vel neque. Nam faucibus, neque ut commodo luctus, lacus risus accumsan felis, a feugiat lorem justo venenatis dui. Aenean bibendum tincidunt enim ac cursus. Vivamus a arcu a augue auctor consectetur nec sed augue. Quisque dignissim felis imperdiet mi lacinia suscipit. Maecenas nunc tortor, congue nec posuere sit amet, ultricies vel diam. In aliquam arcu eu lacus congue eget rutrum justo volutpat. Quisque ac nisi vitae leo fringilla lobortis.

Curabitur rhoncus lobortis ante, eget euismod magna blandit nec. Praesent non sem nulla. Sed congue magna sit amet libero sodales eu ultrices orci posuere. Suspendisse sed nibh a tortor fermentum ornare. Suspendisse vel felis eget tellus gravida rhoncus. Ut vel magna lacus, placerat semper enim. Vestibulum rutrum condimentum neque et adipiscing. Duis nulla enim, euismod a cursus id, ornare vel tellus. Vestibulum lobortis metus egestas velit euismod pellentesque. Praesent elit ante, consequat at posuere a, rhoncus id magna. Phasellus ut nisl orci, ac molestie eros. Suspendisse potenti. Suspendisse ac porttitor lorem. Curabitur eu elit sed neque placerat accumsan. Cras eu odio diam. Nunc lorem ligula, interdum eget consequat non, laoreet eget magna.

Maecenas consequat ultrices est, vitae rutrum nulla egestas sed. Proin rutrum lorem in sem posuere pretium. Cras accumsan euismod quam eget pulvinar. Maecenas eget posuere sem. Nulla sit amet luctus elit. Nulla vel ligula velit. Nunc consectetur orci a odio venenatis facilisis. Integer venenatis commodo nibh sed gravida. Ut ornare arcu in mi eleifend convallis. Quisque tincidunt, tellus et sodales interdum, nulla massa suscipit ante, non tincidunt ligula diam id nunc. In eu justo at lectus pulvinar accumsan. Vivamus convallis sodales ligula, ut gravida elit consectetur at. Ut in augue nec tortor vehicula vehicula eu eu lorem. Vivamus tristique neque ut tellus tristique aliquet.

Proin quis augue a elit convallis venenatis. Quisque scelerisque dictum augue condimentum rutrum. Integer nec dignissim nisl. Aenean vitae justo lectus, eu vulputate ipsum. Sed porttitor dapibus arcu sed faucibus. Sed vitae arcu eu quam ultrices ornare. In in est nec purus consequat vehicula. Integer ut fermentum dolor. Vivamus neque quam, cursus at viverra

Figure 5. The ConcreteOT font produced by MFLua from `ccr10.mf`.

To solve these cases, `end_program.lua` has a series of *filters*. A filter acts on a specific glyph and typically removes unwanted curves and/or adjusts the control points to ensure that a curve joins properly with its predecessors and successor. Of course this means that the programmer inspects each glyph separately, which is reasonable when we are designing the font — less reasonable when we convert it.

We can call this approach *per-font and per-glyph*: `end_program.lua` is a Lua script valid only for a specific font and which has filters for each glyph.

The script `end_program.lua` also has some functions to convert the outlines (with the correct turning number) of each glyph into a SVG font: this font format can be imported into FontForge and, usually after re-editing the glyphs (typically simplifying the curves), it can be saved as an OpenType CFF. In fig. 5 we can see an example of this font.

The Computer Modern Sans Serif Demibold Condensed 10 pt

We now approach a more "geometric" strategy. We don't want to output an OpenType font but to find an `end_program.lua` more *universal and per-glyph* and less *per-font and per-glyph*. Our experience with `ccr10.mf` make us believe that is always possibile to write a METAFONT program that outputs a nice bitmap of a glyph using a very complex set of curves. This is especially true when we use pens and the need to manually correct every error arises. Up to now we only made few outlines of numerals.

There are new functions to trace a curve and to calculate the intersections between two cubics (both based on De Casteljau's bisection algorithm, an application of De Casteljau's algorithm) so the parametric form and the trick to calculate the intersections are not needed anymore. We also keep contours, envelopes and pens apart almost until the end of the process, when we first merge envelopes and pens and then, at last, contours. The most important enhancement is probably the replacement of the polygonal version of a pen with an elliptical one. METAFONT generates a polygonal getting an ideal ellipse with major axis, minor axis and the angle of rotation from the pen specifications and then calls `make_ellipse`. Putting a sensor around helps us store the axis and theta into a Lua table, to be read

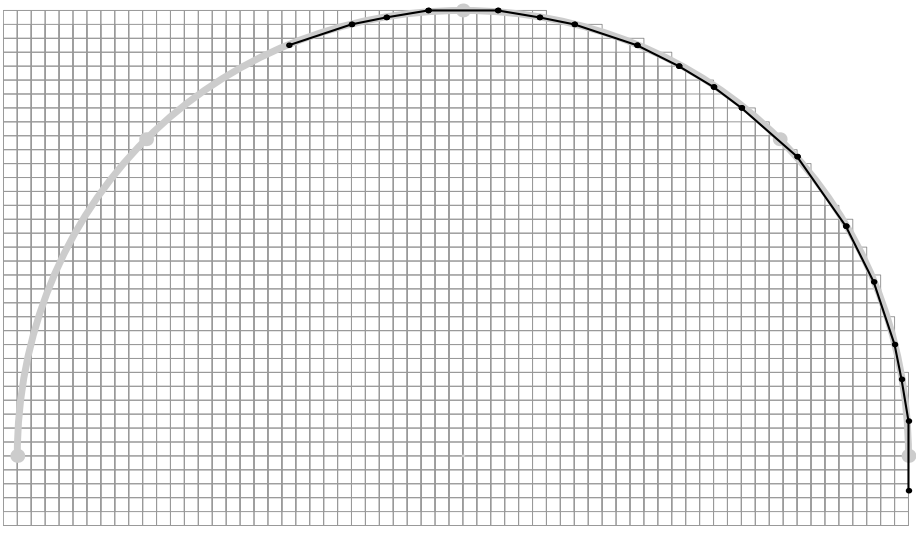


Figure 6. Real polygonal pen (black) vs. calculated elliptical pen (gray). Square boxes are the pixels. It's the bottom right part of fig. 3 .

later from `end_program.lua`. The next step is a trick again: we call MFLua with the following file:

```
batchmode;
fill fullcircle
    xscaled (majoraxis)
    yscaled (minoraxis)
    rotated (theta) shifted (0,0);
shipit;
bye.
```

where `majoraxis`, `minoraxis` and `theta` get the ellipse data. MFLua then saves the outlines of the filled ellipse into another file, from which they can be read by `end_program.lua`. This script then saves each elliptical pen in a table, with `p..c1..c2..q` as a key to be reused later, instead of the polygonal one. We can see the result in fig. 6: the approximation is quite good. This reduces the total number of curves and gives the glyph a more natural look.

5 Conclusion

We believe that MFLua is an interesting tool for font designers because too many fonts (if not all) are currently designed using contours. In this case `end_program.lua` should be simple (less or even no intersections, compared to the METAFONT technique, see the numeral 0 of `xmssdc10.mf`). On the other side, using the pens shows that extracting an outline is a difficult task. It's almost impossible to find an always valid script. The outlines from an envelope usually have a large number of curves, which is not a good feature, and this is a METAFONT property: we can always implement routines to simplify them, though FontForge already does it.

The work will continue on `xmssdc10.mf` to find an `end_program.lua` modular and flexible enough for a wide application.

References

Knuth, D. E. (1986a). *METAFONT: The Program*. Addison-Wesley, Massachusetts, 1ª edizione.

— (1986b). *The METAFONTbook*. Addison-Wesley, Massachusetts, 1ª edizione. — with final correction made in 1995 —.

Marsh, D. (2005). *Applied Geometry for Computer Graphics and CAD*. Springer, London, 2ª edizione.

Notes

1. I borrow notation from Marsh (2005), where points and functions in the Bézier curves section are represented by bold, upright letters.

Luigi Scarso
luigi dot scarso at gmail dot com

Conference portfolio

(Workshop)

Abstract
In accordance to the conference's theme, a workshop for making a portfolio binder has been held. The portfolio was made so it could carry the papers for the conference, such as preprints of the proceedings, additional papers and the carpenter's pencil given to each participant. The construction is made from a single sheet of cardboard with folded flaps along three sides, so that it completely envelopes the content. The portfolio is held closed by a black elastic band.

Introduction

A portfolio is a practical solution to keep all information gathered during a meeting or conference neatly in one place. Most portfolios are made from strong material, for instance manilla cardboard. Normally they are built from several pieces, i.e. the flaps are glued onto the back-cover. In this workshop an intriguing design is used, which allows to prepare the complete portfolio from a single sheet of cardboard without the need for glueing.

Basic design

In order to understand the mechanics of this type of construction, it is a good idea to make a blueprint first. Although the same principles apply to the version made with cardboard, it is important to understand that the drawing is only in two dimensions. In other words, it does not include compensation for the thickness of the content and the material used for the portfolio. When making the actual portfolio, we will have to compensate for this.

The size of your portfolio is dictated by its intended content. Therefore, the height of your blank cardboard sheet should be height (h) + width (w) of the content. The width is calculated by adding twice the width of the content (w) to the width of the fold-in flap (f).

For the real world portfolio, we will take into account the thickness of the content. This adds 2 × the spine width to the width of the sheet.

The conference portfolio

In order to determine the size of the portfolio, one first needs to know the dimensions of the content which it should be able to carry.

For the preprints of the proceedings these dimensions are:

- Height = 265 mm
- Width = 210 mm
- Thickness = 9 mm

The space inside the portfolio should always be slightly larger than the actual dimensions of the content, not to mention we have to leave room for our carpenter's pencil.

The final dimensions of the portfolio are set to:

- Height = 275 mm
- Width = 225 mm
- Spine width = 10 mm

When choosing a material for the portfolio, it is important to check that the grain of the material is in the direction **width** axis of the material. The cardboard size should be 500 × 700 mm (width × length).

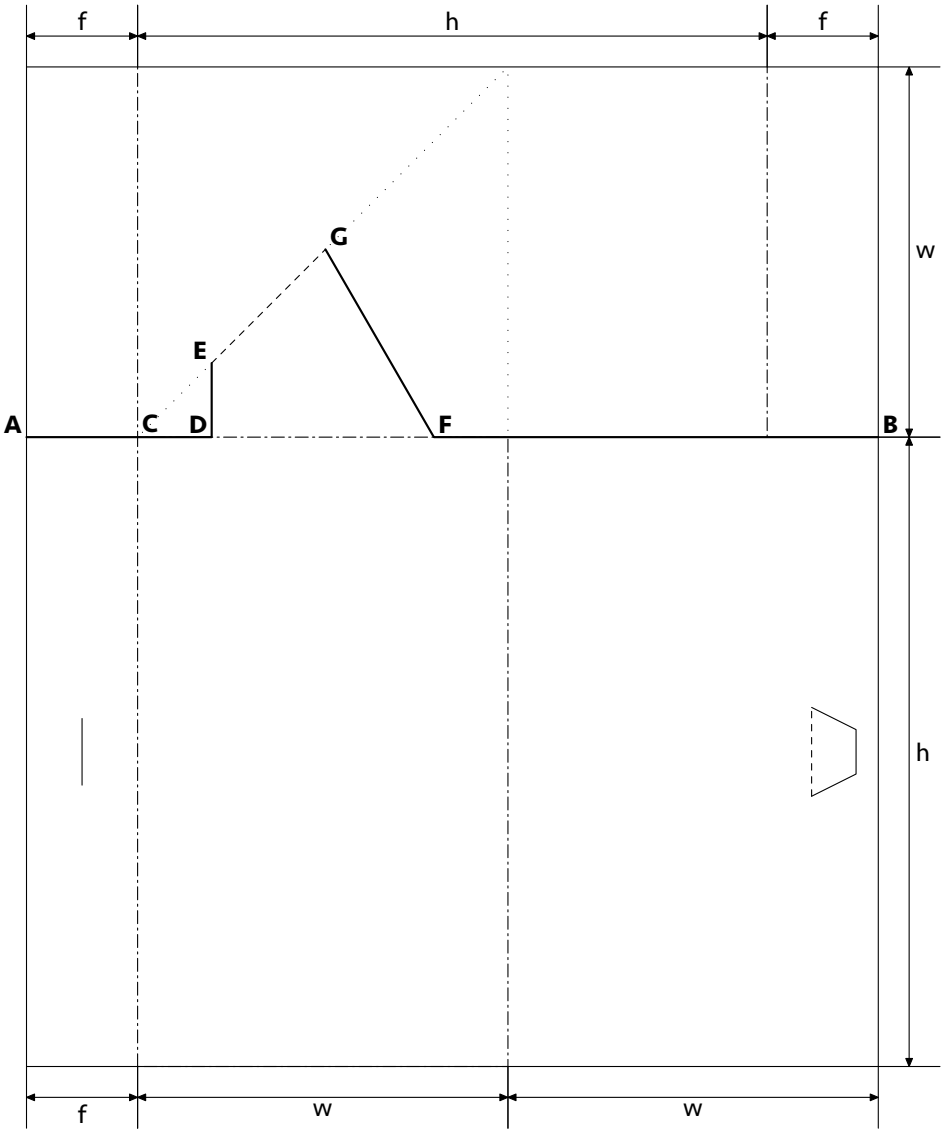


Figure 1. The blueprint for the portfolio

Making the portfolio

In order to get good results, make sure to measure and mark precisely ("measure twice, cut once"). For creasing the folds a fairly sharp bone-folder is advised. For cutting, a cutter with snap-off blade is most suitable. The ruler used should be made of steel. Ones made from hard woods such as beechwood are also usable, however one should be careful not to cut into the wood. To make rulers more steady when drawing and cutting, a strip of sandpaper may be glued to the back of them.

When looking at the blueprint discussed earlier, we see we need to compensate for the thickness of the content (in our case, 10mm). In addition to that, we also have to take into account the thickness of the material. The latter is reflected by the fact that point **C** in the drawing 2 is moved to the right from the inner line of the spine by about 2 mm. Thickness of the material affects overall thickness of the portfolio, and also things such as corner radii, as stiffer materials do not fold as easily.

Correctly creasing the line between points **E** and **G** is crucial. This line must be made at an angle of exactly 45°.

After folding, folds **E** – **G** and **D** – **F** should be sharpened with a bone-folder.

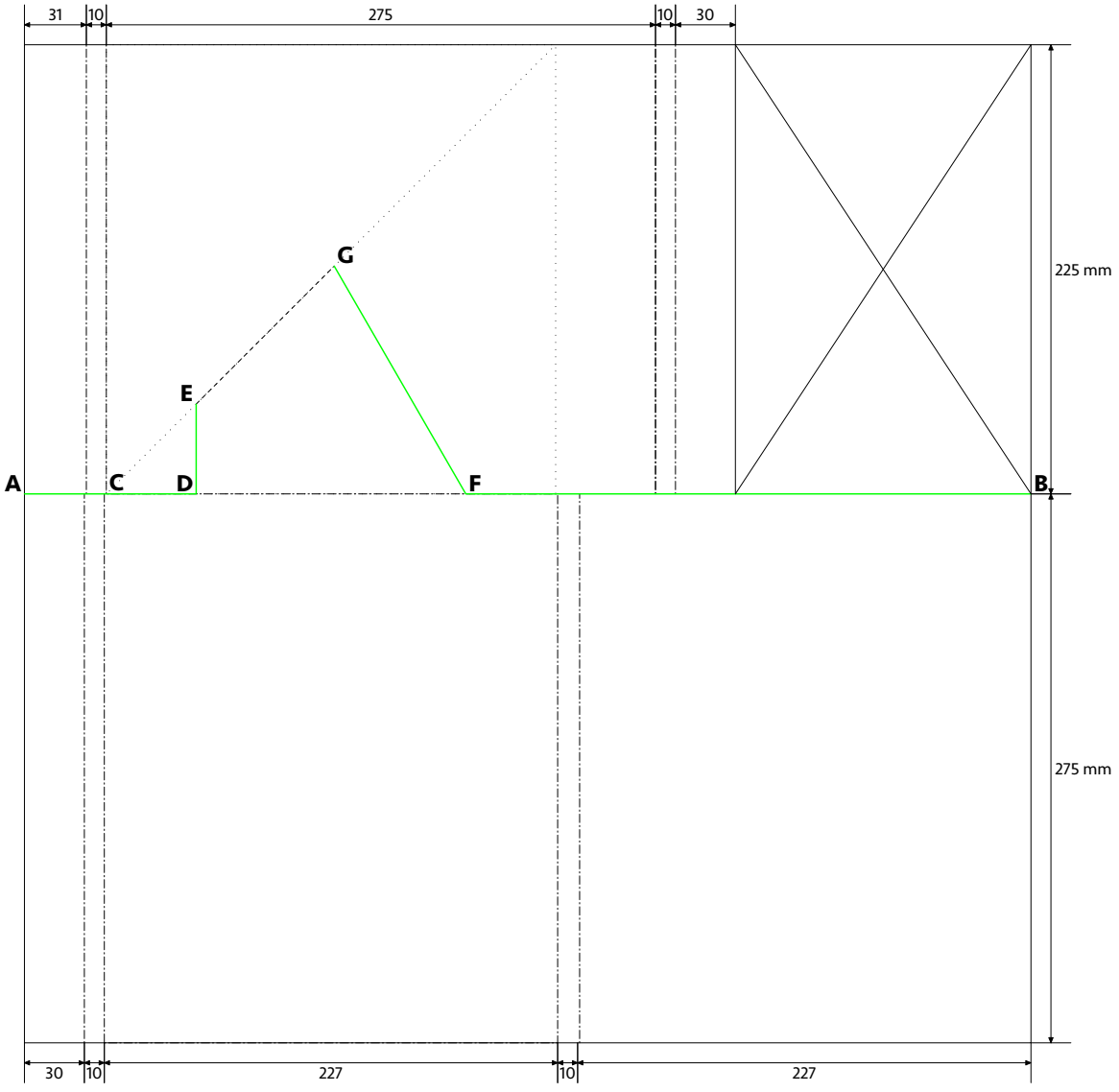


Figure 2. Conference portfolio layout

For folding narrow spines, it is easiest to put a ruler on the inside along the line you wish to fold. Use a bone-folder to follow the edge of the ruler on the outside. As you drag it along the ruler, you push the material upwards into a straight, clean fold.

The spines are just wide enough to place another crease between the outer spine folds. It is a little more work, but allows the spine to flex if the portfolio contains less paper than the spine width allows for. This guarantees a tight fit when the elastic band is used.

To make the slots used to insert a postcard into the front cover, it is best to prepare a template. The template should be 210 × 160 mm and can be made from a piece of discarded cardboard. Draw a rectangle of 150 × 100 mm, which is offset by 40 mm from the bottom and the right edge. Mark at each angle two points e.g. 15 mm offset along the rectangle's frame. Cut the drawn triangles out of the template or mark the 8 points by punching little holes with a sharp awl. By cutting the triangle a fraction of a millimeter larger than drawn, you can then insert a 150 × 100 mm postcard with ease.

For best results, make your pencil marks as lightly as possible or alternatively use the tip of the bone-folder.

When the portfolio is finished, you could cut the short edges of the fold-in flaps with a bevelled edge.

The steps for making the porfolio

☐ Place the sheet in front of you so the widest side is parallel to the edge of the working table.
☐ Mark and crease a horizontal line at 275 mm from the bottom of the sheet (A – B).
☐ Mark and crease a vertical line 30 mm from the left edge extending to the already creased horizontal line (A – B).
☐ Mark and crease a vertical line 40 mm from the left edge extending up to the already creased horizontal line (A – B).
☐ Point C is 2 mm right of the last creased line. Draw a line upwards at an angle of 45°starting from C.
☐ Draw a vertical line 80 mm from the left edge on the horizontally creased line (A – B) (D), until it intersects with the diagonal. This intersection point is point E.
☐ Draw a line starting 210 mm from the left edge on the horizontally creased line (A – B) (F) upwards until it intersects with the diagonal (the angle is not important) (G).
☐ Cut lines A – D, D – E, F – B and F – G.
☐ Crease E – G on the **outside** of the cover. Be very careful when doing this crease, or the portfolio will not fold correctly.
☐ Fold D – F precisely and sharpen the fold with the bone-folder.
☐ Fold E – G while turning the strip, fold down after precise positioning of the strip with the bone-folder.
☐ Mark the fold-in flaps at the bottom and top so that they are approximately 2 mm inside the cover after folding.
Mark two lines 2 and 12 mm to the right of the fold-in (direction of the height of the portfolio).
☐ Completely unfold the portfolio.
Mark and crease a line 10 mm to the left of the bottom turn in. Mark and crease a line 10 mm to the right of the top fold-in.
Crease also the two lines in direction of the height of the portfolio.
☐ Cut the top fold-in to the width of the bottom fold-in flap (30 mm).
☐ There are now four small strips of 10 mm marked by creases. Place another crease between the creased lines if you want to give the portfolio rounded spines.
☐ All creased lines need to be folded now. Use the ruler and the bone-folder for this task.
☐ Refold the portfolio and fold the flaps towards the inside.
☐ Close the portfolio. Mark the width of the cover. Open the cover and check that the marks are at equal distance from the front edge.
☐ Cut the front cover to size. Alternatively you can also fold the oversized flap toward the inside of the front cover. By glueing the edges it forms a pouch.

Making the closing:
For the closing an elastic band is fixed to the portfolio with two eyelets.

☐ Punch two holes in the back, approximately 70 mm from the left edge of the cardboard sheet (this is where the longest fold-in flap is).
☐ Insert the elastic band from the outside inward. Make sure you have about 10 mm of elastic band to spare on the inside.
☐ Insert an eyelet into the pliers and insert the pin with eyelet from the outside through the hole. Arrange the elastic band so it aligns with the long side of the portfolio. Press firmly in place with pliers.
☐ Repeat the procedure for the other hole.

Make the cuts for fixing a postcard (150 × 100 mm)

☐ Open the front cover and flip the portfolio outside up, the front cover being on the right side.
☐ Place the template on the lower right edge of the front cover.
☐ Mark the 4 short diagonal lines. It works best with a sharp awl.
☐ Remove the template and cut the lines precisely.
☐ Insert the corners of the card.

Consideration

The article provides two drawings which enable you to make such portfolios in other sizes. Consider e.g. making a nice wrapping for a present or an invitation. It is also fine to experiment with closings. E.g. a cord could be fixed to the front cover instead of the elastic band and turned around the portfolio. The end is just tucked under the turns of the cord ...

The workshop with the participants was a lot of fun, and very recreational as intended by the theme of this year's EuroTEX.

Willi Egger
w.egger at boede.nl

Oriental TEX: optimizing paragraphs

Introduction

One of the objectives of the Oriental TEX project has always been to play with paragraph optimization. The original assumption was that we needed an advanced non-standard paragraph builder to Arabic done right but in the end we found out that a more straightforward approach is to use a sophisticated OpenType font in combination with a paragraph postprocessor that uses the advanced font capabilities. This solution is somewhat easier to realise than a complex paragraph builder but still involves quite some juggling.

At the June 2012 meeting of the ntg there was a talk about typesetting Devanagari and as fonts are always a nice topic (if only because there is something to show) it made sense to tell a bit more about optimizing Arabic at the same time. In fact, that presentation was already a few years too late because a couple of years back, when the oriental TEX project was presented at tug and Dante meetings, the optimizer was already part of the ConTEXt core code. The main reason for not advocating it was the simple fact that no font other than the (not yet finished) Husayni font provided the relevant feature set.

The lack of advanced fonts does not prevent us from showing what we're dealing with. This is because the ConTEXt mechanisms are generic in the sense that they can also be used with regular Latin fonts, although it does not make that much sense. Anyhow, in the next section we wrap up the current state of typesetting Arabic in ConTEXt. We focus on the rendering, and leave general aspects of bidirectional typesetting and layouts for another time.

This article is written by Idris Samawi Hamid and Hans Hagen and is typeset by ConTEXt MkIV which uses LuaTEX. This program is an extension of TEX that uses Lua to open op the core machinery. The LuaTEX core team consists of Taco Hoekwater, Hartmut Henkel and Hans Hagen.

Manipulating glyphs

When discussing optical optimization of a paragraph, a few alternatives come to mind:

- One can get rid of extensive spaces by adding additional kerns between glyphs. This is often used by poor man's typesetting programs (or routines) and can be applied to non-connecting scripts. It just looks bad. Of course, for connected scripts like Arabic, inter-glyph kerning is not an option, not even in principle.
- Glyphs can be widened a few percent and this is an option that LuaTEX inherits from its predecessor pdfTEX. Normally this goes unnoticed although excessive scaling makes things worse, and yes, one can run into such examples. This strategy goes under the name hz-optimization (the hz refers to Hermann Zapf, who first came up with this solution).[1]
- A real nice solution is to replace glyphs by narrower or wider variants. This is in fact the ideal hz solution —including for Arabic-script as well— but for it to happen

one not only needs needs fonts with alternative shapes, but also a machinery that can deal with them.

□ An already old variant is the one first used by Gutenberg, who used alternative cuts for certain combinations of characters. This is comparable with ligatures. However, to make the look and feel optimal, one needs to analyze the text and make decisions on what to replace without losing consistency.

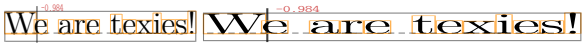
The solution described here does a bit of everything. As it is mostly meant for a connective script, the starting point is how a scribe works when filling up a line nicely. Depending on how well one can see it coming, the writing can be adapted to widen or narrow following words. And it happens that in Arabic-script there are quite some ways to squeeze more characters in a small area and/or expand some to the extreme to fill up the available space. Shapes can be wider or narrower, they can be stacked and they can get replaced by ligatures. Of course there is some interference with the optional marks on top and below but even there we have some freedom. The only condition is that the characters in a word stay connected.[2]

So, given enough alternative glyphs, one can imagine that excessive interword spacing can be avoided. However, it is non-trivial to check all possible combinations. Actually, it is not needed either, as carefully chosen aesthetic rules put some bounds on what can be done. One should more think in terms of alternative strategies or solutions and this is the terminology that we will therefore use.

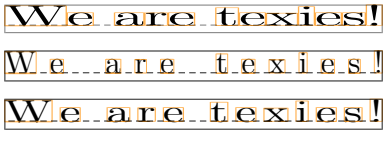
Scaling glyphs horizontally is no problem if we keep the scale factor very small, say percentages. This also means that we should not overestimate the impact. For the Arabic script we can stretch more —using non-scaling methods— but again there are some constraints, which we will discuss later on.

In the next example, we demonstrate some excessive stretching:

In practice, fonts can provide intercharacter kerning, which is demonstrated next:



Some poor man's justification routines mess with additional inter-character kerning. Although this is, within reasonable bounds, ok for special purposes like titles, it looks bad in text. The first line expands glyphs and spaces, the second line expands spaces and adds additional kerns between characters and the third line expands and adds extra kerns.



Unfortunately we see quite often examples of the last method in novels and even scientific texts. There is definitely a down side to advanced manipulation.

Applying features to Latin-script

It is easiest to start out with Latin, if only because it's more intuitive for most of us to see what happens. This is not the place to discuss all the gory details so you have to take some of the configuration options on face value. Once this mechanism is stable and used, the options can be described. For now we stick to presenting the idea.

Let's assume that you know what font features are. The idea is to work with combinations of such features and figure out what combination suits best. In order not to clutter a document style, these sets are defined in so called goodie files. Here is an excerpt of `demo.lfg`:

```
return {
  name = "demo",
  version = "1.01",
```

```
  comment = "An example of goodies.",
  author = "Hans Hagen",
  featuresets = {
    simple = {
      mode   = "node",
      script = "latn"
    },
    default = {
      mode   = "node",
      script = "latn",
      kern   = "yes",
    },
    ligatures = {
      mode   = "node",
      script = "latn",
      kern   = "yes",
      liga   = "yes",
    },
    smallcaps = {
      mode   = "node",
      script = "latn",
      kern   = "yes",
      smcp   = "yes",
    },
  },
  solutions = {
    experimental = {
      less = {
        "ligatures", "simple",
      },

      more = {
        "smallcaps",
      },
    },
  },
}
```

We see four sets of features here. You can use these sets in a ConTEXt feature definition, like:

```
\definefontfeature
  [solution-demo]
  [goodies=demo,
   featureset=default]
```

You can use a set as follows:

```
\definefont
  [SomeTestFont]
  [texgyrepagellaregular*solution-demo at 10pt]
```

So far, there is nothing special or new, but we can go a step further.

```
\definefontsolution
  [solution-a]
  [goodies=demo,
   solution=experimental,
   method={normal,preroll},
   criterium=1]

\definefontsolution
```

```
[solution-b]
[goodies=demo,
 solution=experimental,
 method={normal,preroll,split},
 criterium=1]
```

Here we have defined two solutions. They refer to the `experimental` solution in the goodie file `demo.lfg`. A solution has a `less` and a `more` entry. The featuresets mentioned there reflect ways to make a word narrower or wider. There can be more than one way to do that, although it comes at a performance price. Before we see how this works out we turn on a tracing option:

```
\enabletrackers
  [builders.paragraphs.solutions.splitters.colors]
```

This will color the words in the result according to what has happened. When a featureset out of the `more` category has been applied, the words turn green, when `less` is applied, the word becomes yellow. The `preroll` option in the `method` list makes sure that we do a more extensive test beforehand.

```
\SomeTestFont \startfontsolution[solution-a]
\input zapf \par
\stopfontsolution
```

In Figure 1 we see what happens. In each already split line words get wider or narrower until we're satisfied. A criterium of 1 is pretty strict[3]. Keep in mind that we use some arbitrary features here. We try removing kerns to get narrower although there is nothing that guarantees that kerns are positive. On the other hand, using ligatures might help. In order to get wider we use smallcaps. Okay, the result will look somewhat strange but so does much typesetting nowadays.

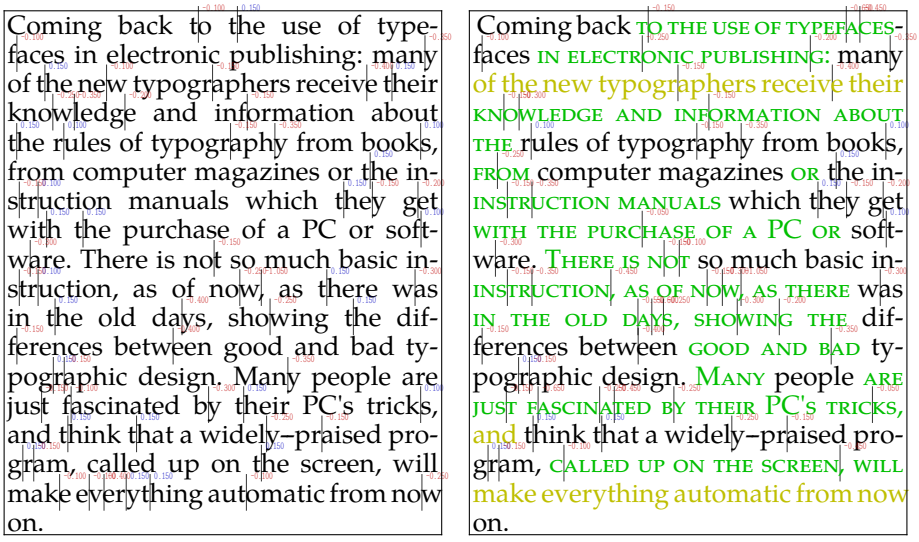


Figure 1. Solution a.

There is one pitfall here. This mechanism is made for a connective script where hyphenation is not used. As a result a word here is actually split up when it has discretionaries and of course this text fragment has. It goes unnoticed in the rendering but is of course far from optimal.

```
\SomeTestFont \startfontsolution[solution-b]
\input zapf \par
\stopfontsolution
```

In this example (Figure 2) we keep words as a whole but as a side effect we skip words that are broken across a line. This is mostly because it makes not much sense to implement it as Latin is not our target. Future versions of ConTEXt might get more sophisticated font machinery so then things might look better.

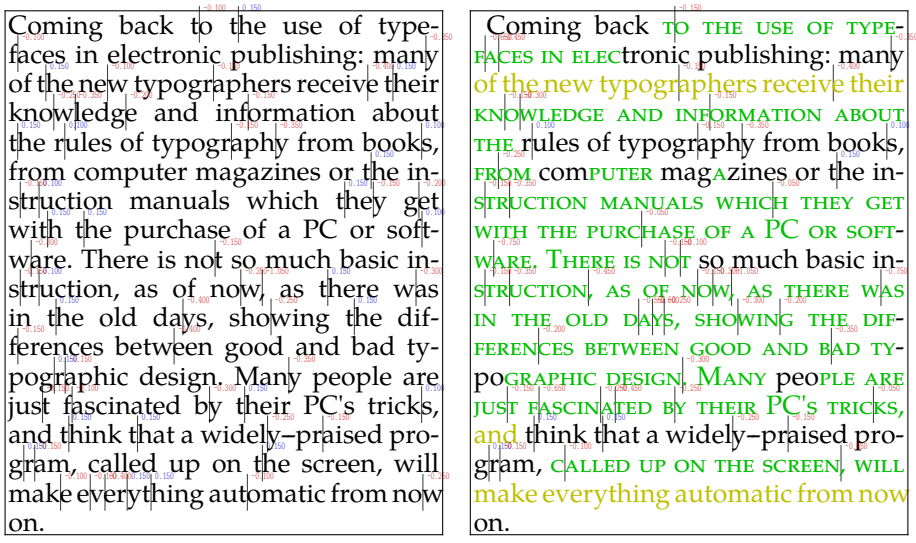


Figure 2. Solution b.

We show two more methods:

```
\definefontsolution
  [solution-c]
  [goodies=demo,
   solution=experimental,
   method={reverse,preroll},
   criterium=1]

\definefontsolution
  [solution-d]
  [goodies=demo,
   solution=experimental,
   method={random,preroll,split},
   criterium=1]
```

In Figure 3 we start at the other end of a line. As we sort of mimick a scribe, we can be one who plays safe at the start of corrects at the end.

In Figure 4 we add some randomness but to what extent this works well depends on how many words we need to retypeset before we get the badness of the line within the constraints.

Salient features of Arabic-script

Before applying the above to Arabic-script, let's discuss some salient aspects of the problem. As a cursive script, Arabic is extremely versatile and the scribal calligraphy tradition reflects that. Digital Arabic typography is only beginning to catch up with

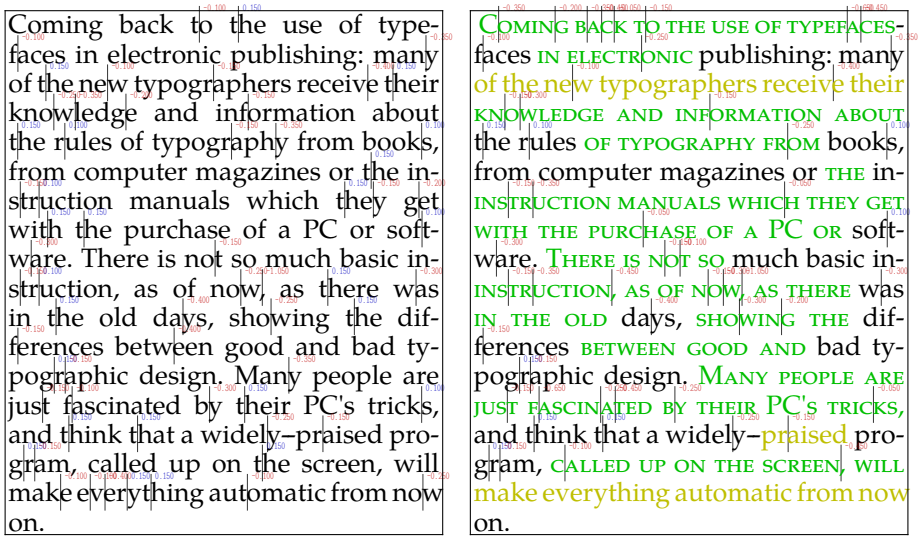


Figure 3. Solution c.

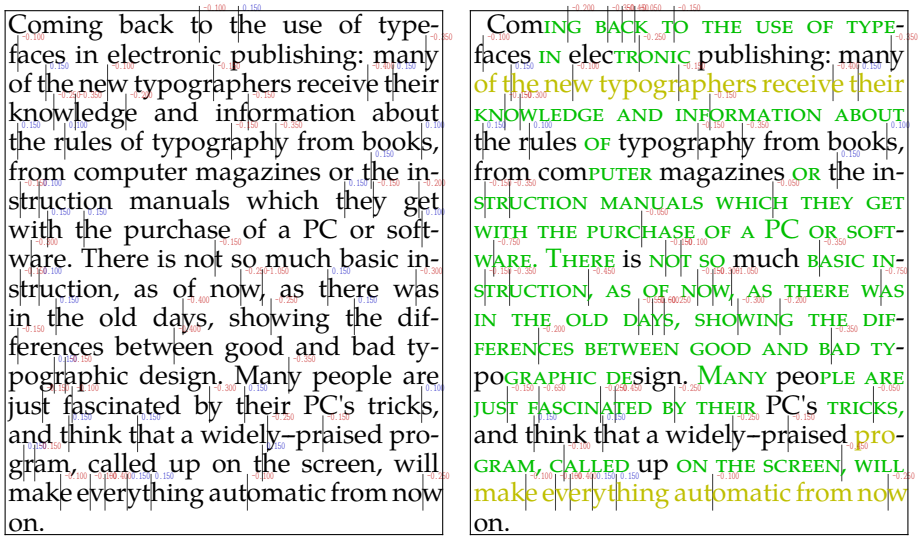


Figure 4. Solution d.

the possibilities afforded by the scribal tradition. Indeed, early lead-punch typography and typesetting of Arabic-script was more advanced than most digital typography even up to this day. In any case, let us begin to organize some of that versatility into a taxonomy for typography purposes.

What's available?

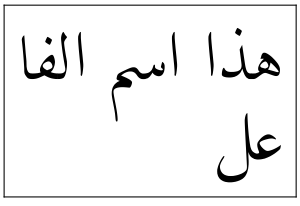
We have to work within the following parameters:

□ No hyphenation ever (well, almost never)
It is commonly pointed out that there is no hyphenation in Arabic. This is something of a half-truth. In the manuscript tradition one actually does find something akin to hyphenation. In the ancient Kufic script, breaking a word across lines is

actually quite common. But even in the more modern Naskh script, the one most normal Arabic text fonts are based on, it does occur, albeit rarely and presumably when the scribe is out of options for the line he is working on. Indeed, one could regard it as a failure on the part of the scribe once he reaches the end of the line.[4]

But there is still an important rule, regardless of whether we use Naskh, Kufic, or any other Arabic script. Consider the word below:

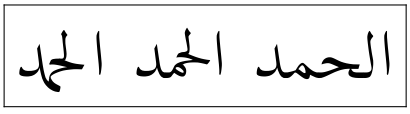
It is a single word composed of two cursive strings. One could actually hyphenate it, with our rule being to break it at the end of the first cursive string and before the beginning of the second cursive string:



Again, it's a rare phenomenon and hardly ever occurs in modern typesetting, lead-punch or digital, if at all. On the other hand, it could have some creative uses in future Arabic-script typography.

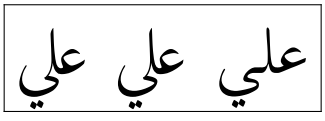
□ Macrotypography (aesthetic features)

In Arabic there are often numerous aesthetic ways of writing out the exact same semantic string:[5]



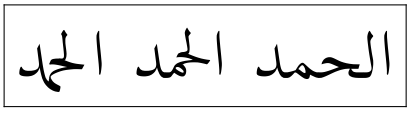
Normally we combine OpenType features into feature sets that are each internally and aesthetically coherent. So in the above example we have used three different sets, reading from right to left. We'll call them simple, default, and dipped.

Just as Latin typography uses separate fonts to mark off different uses of text (bold, italic, etc.), an advanced Arabic font can use aesthetic feature sets to similar effect. This works best on distinguishing long streams of text from one another, since the differences between feature sets are not always noticeable on short strings. That is, two different aesthetic sets may type a given short string, such as a single word, exactly the same way. Consider the above three sets (simple, default, and dipped) once more:



For the above string the default and dipped aesthetic sets (middle and left) give the exact same result, while the basic one (right) remains, well, quite basic.

Let's go back to our earlier example:



Note that the simple version is wider than the default, and the dipped version is (slightly) thinner than the default. This relates to another point: An aesthetic feature set can serve two functions:

1. It can serve as the base aesthetic style.
2. It can serve as a resource for glyph substitution for a given string in another base aesthetic style.

This brings us back to our main topic.

□ Microtypography (paragraph optimization features)
Here our job is to optimize the paragraph for even spacing and aesthetic viewing. It turns out that there are a number of ways to look at this issue, and we will begin exploring these in the next subsection.

Two approaches

Let us start off with a couple of samples. Qurʾānic transcription has always been the gold standard of Arabic-script. In Figure 5 we see a nice example of scribal optimization. The scribe here is operating under the constraint that each page ends with the

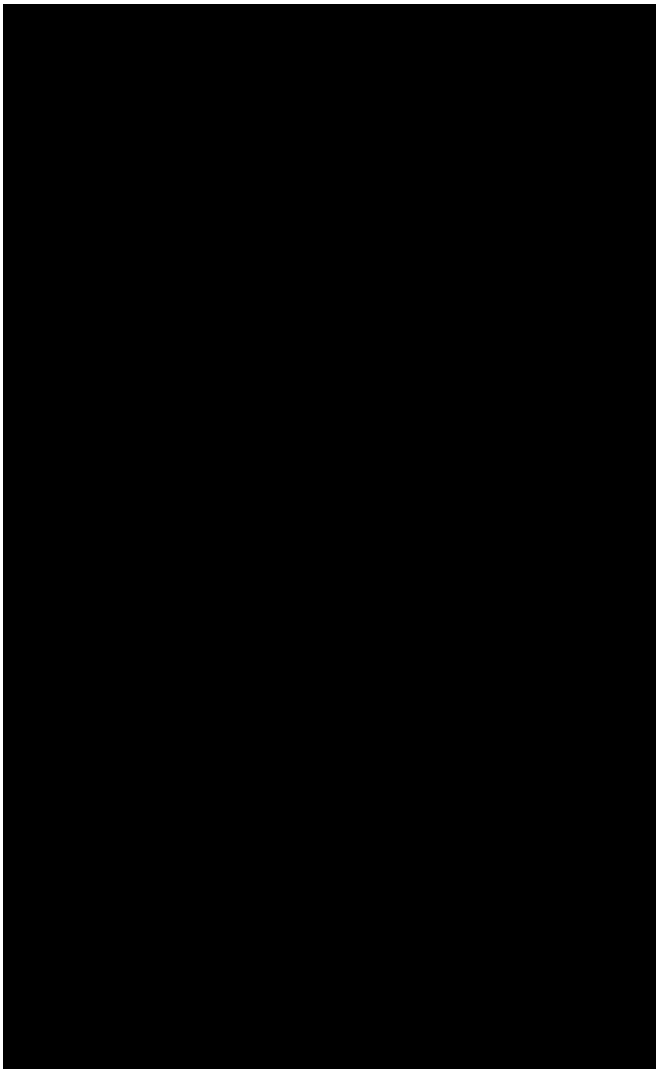


Figure 5. Scribal Optimization. Scribe: *ᶜUthmān Ṭāhā*. Qurʾān, circa 1997.

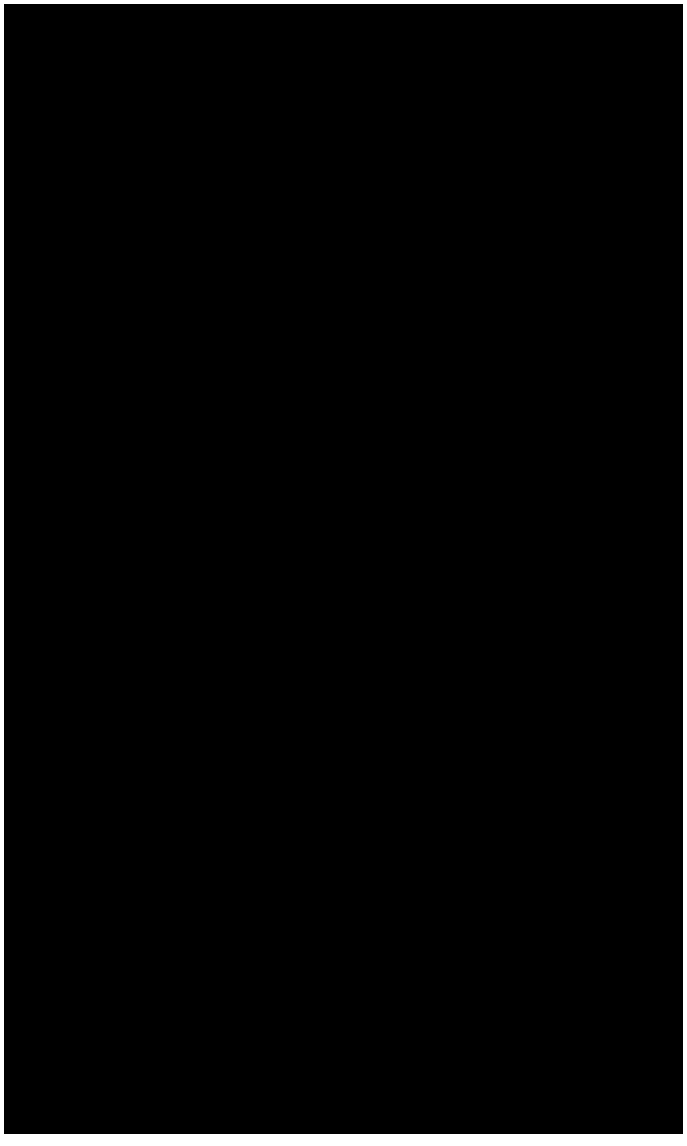


Figure 6. Alternate Fixed Glyphs. From the *al-Husayni Muṣḥaf* of the Qurʾān, 1923.

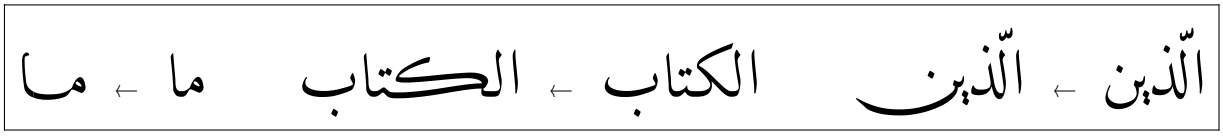
end of a Qurʾānic verse (designated by the symbol U+06DD ۝). That is, no verse is broken across pages. That constraint, which is by no means mandatory or universal, gives the scribe lots of space for optimization, even more than normal.

In Figure 6 we have a page of the famous *al-Husayni Muṣḥaf* of 1919–1923, which remains up to this day the only typeset copy of the Qurʾān to attain general acceptance in the Muslim world. Indeed, it remains the standard 'edition' of the Qurʾān and even later scribal copies, such as the one featured in Figure 5 are based on its orthography. Unlike the scribal version, the typesetters of the *al-Husayni Muṣḥaf* did not try to constrain each page to end with the end of a Qurʾānic verse. Again, that is a nice feature to have as it makes recitation somewhat easier but it is by no means a mandatory one.

In any case, both samples share verses 172–176 in commmon, so there is lots to compare and contrast. We will also use these verses as our main textual sample for paragraph optimization.

Using Figure 5 and Figure 6 as benchmarks, we can begin by analyzing the approaches to paragraph optimization in Arabic-script typography into two kinds:

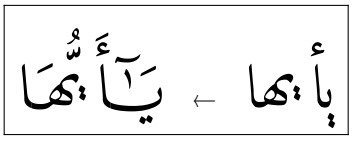
□ Alternate glyphs
Much of pre-digital Arabic typography uses this method. Generally, a wide variant of a letter is used to take up the space which would normally get absorbed by hyphenation in Latin. Here are examples of three of the most common substitutions, again, reading from right to left:



Each of the six strings above occurs in Figure 6. Identifying them is an exercise left to the reader. We call these kinds of alternate glyphs *alternate-shaped* glyphs.

The three substitutions above are the most common alternate-glyph substitutions found in pre-digital Arabic-script typography, including some contextual variants (initial, medial, final, and isolated) where appropriate. (The scribal tradition contains a lot more alternate-shaped glyphs. A few lead-punch fonts implement some of them, and we have implemented many of these in our Husayni font.) The results generally look quite nice and much more professional than most digital Arabic typography, which generally dispenses with these alternates.

But one also finds attempts at *extending* individual characters without changing the shape very much. One finds this already in Figure 6. We call these kinds of alternate glyphs *naturally curved widened* glyphs, or just *naturally widened* glyphs for short. Sometimes this is done for the purpose of making enough space for the vowels (which in Arabic take the form of diacritic characters). For example:



As you can see, there are two letters that have been *widened* for vowel accommodation. In Figure 6 there are some good but near-clumsy attempts at this. We say, 'near-clumsy' because the typographers and typesetters mix natural, curved, *widened* variants of letters with flat, horizontal, *extended* versions. One reason for this is that a full repertoire of naturally curved glyph alternates would be much too unwieldy for even the best lead-punch typesetting machines and their operators. Even with these limitations one can find brave examples of lead-based typesetting that do a good job of sophisticated paragraph optimization via glyph alternates, both widened and alternate-shaped. Figure 7 is a representative example (in the context of columns).

Careful examination of this two-column sample will reveal the tension between naturally *widened* and horizontally *extended* glyphs in the execution of paragraph optimization. On the other hand, there is one apparent 'rule' that one finds in this and other examples of lead-punch Arabic-script typesetting:

Generally, there is only one naturally widened character per word or one alternate-shaped character per word.

In Figure 5 one can see that this 'rule' is not always observed by scribes, see, e.g., the middle word in line 9 from the top, which uses two of the alternate-shaped characters we encountered above (can you identify that word?). But we still need

كالسِّلْسِلة . وشَيْءٌ (مسَلْسَلٌ) مُتَّصِلٌ بَعْضُه ببَعْض ومنه (سِلْسِلة) الحَديد .

* س ل م — (سَلْم) آسم رجُلٍ و (سَلْمَى) آسم آمرأة . و (سَلْمَانُ) آسم جَبَل وآسم رَجُلٍ . و (سَالِم) آسم رجل . و (السَّلَمُ) بفتحتين السَّلَف . والسَّلَمَ أيضا (الآسْتِسْلام) . و (السَّلَمَ) أيضا شَجَرٌ من العِضَاهِ الواحدة سَلَمة . و (سَلَمَةُ) أيضا آسم رَجُل . و (السُّلَّم) بفتح اللام وَاحِدُ (السَّلَالِيم) التى يُرْتَقَى عليها . و (السِّلْم) السَّلَام . وقرأ أبو عَمْرو : « أُدْخُلوا فى السِّلْم كافَّةً » وذهب بمعناها إلى الإسلام . و (السِّلْم) الصُّلْح بفتح السين وكسرها يُذَكَّر ويؤنَّث . والسِّلْم المُسَالِم تقولُ أنَا سِلْمٌ لمن سَالَمَنِي . و (السَّلامُ السَّلَامة) . و (السَّلامُ) الآسْتِسلام . والسَّلام الآسمُ من التسليم . والسَّلام آسمٌ من أسْماءِ الله تعالى . والسَّلامُ البراءةُ منَ العُيُوب فى قَوْل أُمَيَّة .

وقرئ « وَرَجُلًا سَلَمًا » و (السَّلامَيَاتُ) بفتح المِيم عِظَام الأصَابِع واحدها (سُلَامَى) وهو آسم للواحد والجمع أيضا . و (السَّلِيم) اللَّديغ كأنهم تَفَاءَلُوا له بالسَّلَامة وقيل لأنه أُسْلِم لِمَا به . وقَلْبٌ سَلِيم أى سَالِم . و (سَلِم) فلان من الآفات بالكسر (سَلَامةً) و (سلَّمه) اللهُ منها . و (سَلَّم) إليه الشَّيْءَ (فَتَسَلَّمه) أى أخذه . و (التَّسْلِيم) بَذْل الرِّضَا بالحُكْم . والتَّسْلِيم أيضا السَّلام . و (أَسْلَمَ) فى الطَّعام أسْلَفَ فيه . وأَسْلَمَ أمْرَه إلى الله أى سَلَّم . وأَسْلَمَ دَخَل فى (السَّلَم) بفتحتين وهو الآستِسْلام و (أَسْلَمَ) مِن الإسلام . وأَسْلَمَه خَذَله . و (التَّسَالُمُ) التَّصالُحُ . و (المُسالَمَة) المُصالَحَة . و (آسْتَلَمَ) الحَجَرَ لَمَسَه إما بالقُبْلة أو[1] باليَد ولا يُهْمَز وبعضُهم يَهمِزه . و (آسْتَسْلَمَ) أى آنْقادَ .

* س ل ا — (سَلَا) عنه من باب سَمَا و (سَلِيَ) عنه بالكسر (سُلِيًّا) مثله .

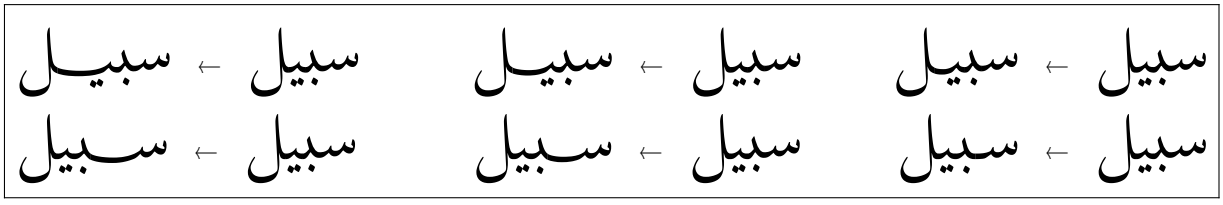
(١) ذكر فى مادة — أ م ا — صفحة ٢٧ ، أنه لابدّ من تكرار « إمّا »

Figure 7. Mixed Alternate Glyphs in Two Columns. From the classical dictionary *Mukhtār al-Ṣiḥāḥ*.

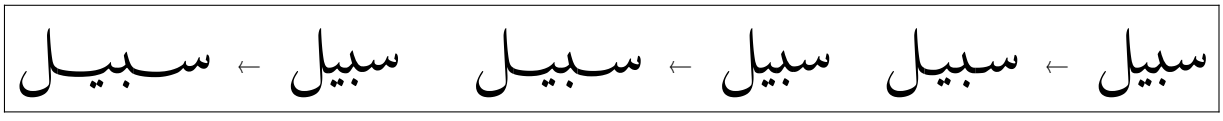
some constraints for decent-looking typesetting, and the above tentative rule is a good place to start the analysis. For widened characters in particular we see that even the scribe (Figure 5) closely approximates this rule. So let's begin improving on our tentative rule somewhat, and expand it into a number of possibilities. Let's look at the naturally-widened-glyph case first:

Generally, there is only one naturally widened character allowed per word. However, two extended non-consecutive characters may be allowed. (The logic of the experimental font Husayni already has contraints that prevent *consecutive* curved widened characters).

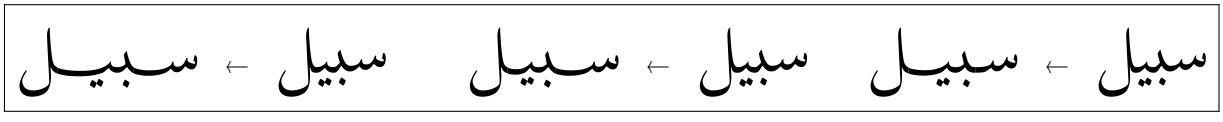
For example, we prefer to get widening like the following:



But as, e.g., a last resort or for stylistic purposes we can also do



Or even better, we mix it up a bit. That is, if there is more than one widened character, one should be longer than the other, e.g.:



One will notice that the middle substitution (where the first widened character is longer than the second) does not look as good as the two outer ones (where the second is longer than the first). These kinds of aesthetic issues can be formalized for future work. In the meantime, here is a working modified version of the rule for naturally-widened-glyphs:

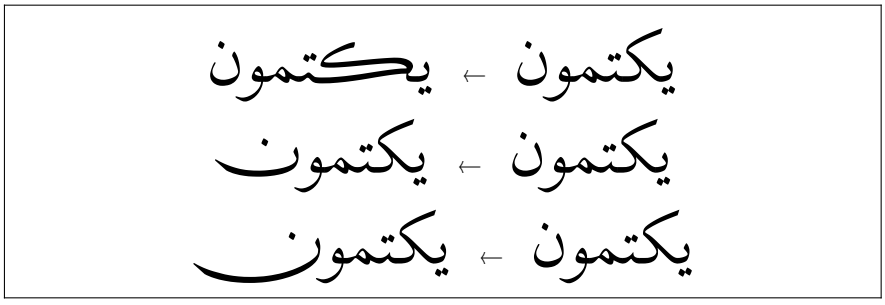
Generally, there is only one naturally widened character allowed per word. However, two non-consecutive widened characters may be allowed. In that case, the second widened character should be longer than the first.

One case where cases of two naturally widened characters will be common is in poetry, which involves wide lines. We'll say more about this in the section on flat extending.

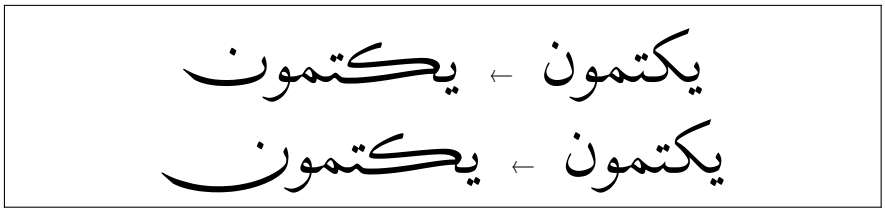
Now let's look at the alternate-shaped case:

Generally, there is only one alternate-shaped character allowed per word. However, two non-consecutive alternate-shaped characters may be allowed.

So we prefer, e.g.,



but we could have, e.g, as a last resort or as a stylistic option,



Again, in poetry this kind of multiple substitution within a single word could occur frequently. A challenge will be to develop a system of parameters where we can almost predict which kinds of substitution will happen under a given set of values of those parameters.

□ Flat extending

In the transition from lead-punch to digital typography, alternate-glyph substitution largely vanished.[6] The problem of spacing remained, and a simple yet inelegant solution was adopted: flat, horizontal extending of characters. Now this solution did have *some* precedent in pre-digital Arabic typography, as you can see in Figure 6 and Figure 7. This solution had the advantage that it required only a single character: a simple horizontal bar called a *taṭwīl* or more commonly a *kashidah* (U+0640). This character could then be repeated as often as necessary to fill any extra space.

Now an examination of pre-digital books shows a (rather wise) reticence to using this method too slavishly. That reticence has now been thrown to the winds. This can be seen by looking at the standard implementation of flat extending as provided by Microsoft Word. This program provides three levels of extending that it calls 'justification'. See Figure 9 for examples of all three. The minimum level is actually very close to the default (i.e., no-justification) level. Note that the sample text used in Figure 9 is the same as that used in the earlier samples from the Qurʾān.

Older implementations of Arabic-script within TEX, such as ArabTEX and Omega/Aleph, also provided facilities for flat extending. The most common use was in poetry, which requires a fixed width for each stanza.

In Omega/Aleph, a method based on `\xleaders` was used, based on a very thin *taṭwīl* glyph (much thinner than U+0640) that could be used for very fine extending optimization based on TEX's badness parameter. One nice application is in marginal notes: See Figure 10, where the marginal note on the right is zoomed in. On the other hand, we see that the leaders method creates extending that may be considered too perfectly even: Do we want to impose the rule that only one character should be extended per word (or at most two non-consecutive characters)? I have seen a lot of older digital Arabic typography that does even extending, including the poetry in the ArabTEX sample in figure 8. Compare this with the Microsoft Word method (Figure 9). The method used in Microsoft Word, with only one extension per word, seems to be the current standard for flat-extending justification.

فَإِنْ كَانَ ٱلْقَوِيُّ ٱلْوُجُودَ، إِطْمَأَنَّتِ ٱلنَّفْسُ وَ كَانَتْ أُخْتُ ٱلْعَقْلِ. وَ رَقَّتِ ٱلْمَاهِيَّةُ وَ شَابَهَتِ [٢٠٢] ٱلْوُجُودَ، كَٱلْحَدِيدَةِ ٱلْمُحَمَّاةِ فِي ٱلنَّارِ. فَلَا فَرْقَ فِي ٱلْفِعْلِ بَيْنَهُمَا، وَ إِنْ كَانَ مَا بِهَا بِٱلْعَرَضِ، كَٱلْحَدِيدِ. قَالَ ٱلشَّاعِرُ:

رَقَّ ٱلزُّجَاجُ وَ رَقَّتِ ٱلْخَمْرُ فَتَشَاكَلَا وَ تَشَابَهَ ٱلْأَمْرُ
فَكَأَنَّمَا خَمْرٌ وَ لَا قَدَحٌ وَ كَأَنَّمَا قَدَحٌ وَ لَا خَمْرُ

وَ إِنْ كَانَ ٱلْقَوِيُّ ٱلْمَاهِيَّةَ كَانَ ٱلْأَمْرُ عَلَى ٱلْعَكْسِ. وَ كُلُّ وَاحِدٍ مِنْهُمَا إِنَّمَا يَسْتَمِدُّ وَ يَقْوِي بِمَدَدٍ مِنْ جِنْسِهِ إِذْ لَا يَسْتَمِدُّ ٱلشَّيْءُ مِنْ نَحْوِ مَا هُوَ مِنْ ضِدِّهِ. فَلَا يَسْتَمِدُّ ٱلنُّورُ مِنَ ٱلظُّلْمَةِ وَ لَا ٱلْعَكْسُ مِنْ حَيْثُ هُوَ كَذٰلِكَ. وَ مَيْلُ ٱلْآخَرِ مَعَهُ، إِنَّمَا هُوَ لِبَقَآئِهِمَا.

Figure 8. Poetry Justification in ArabTEX.

يَا أَيُّهَا الَّذِينَ آمَنُوا كُلُوا مِنْ طَيِّبَاتِ مَا رَزَقْنَاكُمْ وَاشْكُرُوا لِلَّهِ إِنْ كُنتُمْ إِيَّاهُ تَعْبُدُونَ ١٧٢ إِنَّمَا حَرَّمَ عَلَيْكُمُ الْمَيْتَةَ وَالدَّمَ وَلَحْمَ الْخِنزِيرِ وَمَا أُهِلَّ بِهِ لِغَيْرِ اللَّهِ ۖ فَمَنِ اضْطُرَّ غَيْرَ بَاغٍ وَلَا عَادٍ فَلَا إِثْمَ عَلَيْهِ ۚ إِنَّ اللَّهَ غَفُورٌ رَحِيمٌ ١٧٣ إِنَّ الَّذِينَ يَكْتُمُونَ مَا أَنزَلَ اللَّهُ مِنَ الْكِتَابِ وَيَشْتَرُونَ بِهِ ثَمَنًا قَلِيلًا ۙ أُولَٰئِكَ مَا يَأْكُلُونَ فِي بُطُونِهِمْ إِلَّا النَّارَ وَلَا يُكَلِّمُهُمُ اللَّهُ يَوْمَ الْقِيَامَةِ وَلَا يُزَكِّيهِمْ وَلَهُمْ عَذَابٌ أَلِيمٌ ١٧٤ أُولَٰئِكَ الَّذِينَ اشْتَرَوُا الضَّلَالَةَ بِالْهُدَىٰ وَالْعَذَابَ بِالْمَغْفِرَةِ ۚ فَمَا أَصْبَرَهُمْ عَلَى النَّارِ ١٧٥ ذَٰلِكَ بِأَنَّ اللَّهَ نَزَّلَ الْكِتَابَ بِالْحَقِّ ۗ وَإِنَّ الَّذِينَ اخْتَلَفُوا فِي الْكِتَابِ لَفِي شِقَاقٍ بَعِيدٍ ١٧٦

يَا أَيُّهَا الَّذِينَ آمَنُوا كُلُوا مِنْ طَيِّبَاتِ مَا رَزَقْنَاكُمْ وَاشْكُرُوا لِلَّهِ إِنْ كُنْتُمْ إِيَّاهُ تَعْبُدُونَ ١٧٢ إِنَّمَا حَرَّمَ عَلَيْكُمُ الْمَيْتَةَ وَالدَّمَ وَلَحْمَ الْخِنْزِيرِ وَمَا أُهِلَّ بِهِ لِغَيْرِ اللَّهِ ۖ فَمَنِ اضْطُرَّ غَيْرَ بَاغٍ وَلَا عَادٍ فَلَا إِثْمَ عَلَيْهِ ۚ إِنَّ اللَّهَ غَفُورٌ رَحِيمٌ ١٧٣ إِنَّ الَّذِينَ يَكْتُمُونَ مَا أَنزَلَ اللَّهُ مِنَ الْكِتَابِ وَيَشْتَرُونَ بِهِ ثَمَنًا قَلِيلًا ۙ أُولَٰئِكَ مَا يَأْكُلُونَ فِي بُطُونِهِمْ إِلَّا النَّارَ وَلَا يُكَلِّمُهُمُ اللَّهُ يَوْمَ الْقِيَامَةِ وَلَا يُزَكِّيهِمْ وَلَهُمْ عَذَابٌ أَلِيمٌ ١٧٤ أُولَٰئِكَ الَّذِينَ اشْتَرَوُا الضَّلَالَةَ بِالْهُدَىٰ وَالْعَذَابَ بِالْمَغْفِرَةِ ۚ فَمَا أَصْبَرَهُمْ عَلَى النَّارِ ١٧٥ ذَٰلِكَ بِأَنَّ اللَّهَ نَزَّلَ الْكِتَابَ بِالْحَقِّ ۗ وَإِنَّ الَّذِينَ اخْتَلَفُوا فِي الْكِتَابِ لَفِي شِقَاقٍ بَعِيدٍ ١٧٦

يَا أَيُّهَا الَّذِينَ آمَنُوا كُلُوا مِنْ طَيِّبَاتِ مَا رَزَقْنَاكُمْ وَاشْكُرُوا لِلَّهِ إِنْ كُنْتُمْ إِيَّاهُ تَعْبُدُونَ ١٧٢ إِنَّمَا حَرَّمَ عَلَيْكُمُ الْمَيْتَةَ وَالدَّمَ وَلَحْمَ الْخِنْزِيرِ وَمَا أُهِلَّ بِهِ لِغَيْرِ اللَّهِ ۖ فَمَنِ اضْطُرَّ غَيْرَ بَاغٍ وَلَا عَادٍ فَلَا إِثْمَ عَلَيْهِ ۚ إِنَّ اللَّهَ غَفُورٌ رَحِيمٌ ١٧٣ إِنَّ الَّذِينَ يَكْتُمُونَ مَا أَنزَلَ اللَّهُ مِنَ الْكِتَابِ وَيَشْتَرُونَ بِهِ ثَمَنًا قَلِيلًا ۙ أُولَٰئِكَ مَا يَأْكُلُونَ فِي بُطُونِهِمْ إِلَّا النَّارَ وَلَا يُكَلِّمُهُمُ اللَّهُ يَوْمَ الْقِيَامَةِ وَلَا يُزَكِّيهِمْ وَلَهُمْ عَذَابٌ أَلِيمٌ ١٧٤ أُولَٰئِكَ الَّذِينَ اشْتَرَوُا الضَّلَالَةَ بِالْهُدَىٰ وَالْعَذَابَ بِالْمَغْفِرَةِ ۚ فَمَا أَصْبَرَهُمْ عَلَى النَّارِ ١٧٥ ذَٰلِكَ بِأَنَّ اللَّهَ نَزَّلَ الْكِتَابَ بِالْحَقِّ ۗ وَإِنَّ الَّذِينَ اخْتَلَفُوا فِي الْكِتَابِ لَفِي شِقَاقٍ بَعِيدٍ ١٧٦

Figure 9. Flat justification from Microsoft Word 2010.

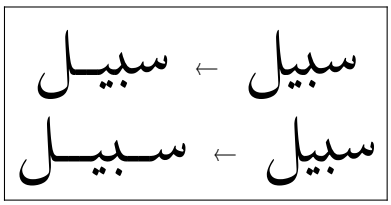
On the other hand, the justification used in Microsoft Word is not particularly aesthetically pleasing. The answer will lie, again, in parameterization of some sort

Figure 10. Marginal-note justification in Omega/Aleph.

to be determined. As TEXies, we want to be able to have fine control over this kind of behavior in any case. In the meantime, we mirror the same rule we arrived at for naturally-widened-glyphs:

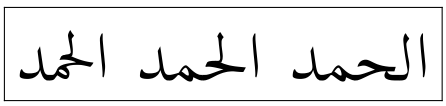
> *Generally, there is only one flat extended character allowed per word. However, two non-consecutive extended characters may be allowed. In that case, the second extended character should be longer than the first.*

For example:

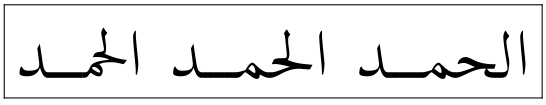


In accordance with our working rule, the top substitution uses only one flat extended character. The bottom uses two, but the second is longer than the first.

In our own estimation, the smaller the type, as in, e.g., footnotes and marginal notes, the less aesthetic variants that are needed. And the less aesthetic variants needed, the better that flat extending will work as a solution. Consider another example of the same word processed in three different variants:

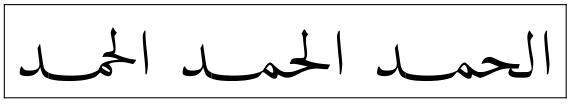


In this case our default is on the left. The variant on the right is about as basic as one can get; the default on the left is a sophisticated aesthetic variant. The middle one is, well, in between. Let's try them with flat extending, using only one extended character per word:



On the left, we have an aesthetic combination of letters followed by a flat *taṭwīl*. This is what Microsoft Word would give us, and the result is aesthetically distasteful. In the word on the right, however, the flat extension fits well with the basic nature of the feature set. As for the middle one, it could go either way and we leave it to the reader to decide what one thinks.

Now let's repeat with more naturally curved widening:



Here, the variant on the left comes out much nicer. The one on the right looks okay with curved widening, although one could arguably do better with flat extending, at least in some contexts. The middle one, again, could go either way, though we think it does somewhat better with curved widening compared to the one on the right. The variant on the left only works well with curved widening.

Towards a ConTEXt solution

In what follows, we will focus on a solution to the problem of paragraph optimization via alternate glyphs (including alternately-shaped and naturally-widened variants). It turns out that the `\xleaders` method used by Omega/Aleph does not work in LuaTEX, so flat extending could not be naively implemented that way. At the moment flat extending is yet to be implemented in ConTEXt.

Since flat extending is so ubiquitous in current Arabic-script typography, and since it does have important applications (poetry and small font sizes where one prefers simpler aesthetic variants), one could ask why this was not implemented first. In part, this is because the immediate priority of the Oriental TEX project has been top-notch, unparalleled aesthetic sophistication of the script. As we noted above, flat extending does not work so well with sophisticated aesthetic variation. So although the flat-extending problem is apparently simpler, it is understandable that we have focused on the more difficult problem first. A clear understanding of the issues and challenges involved with the more general alternate glyph method will help us implement a solution to the the flat-extended problem as a special case. We will come back to this issue towards the end.

Let us now consider the current experimental ConTEXt setup for paragraph optimization for Arabic-script.

Applying Features to Arabic-script

We're now ready for the real thing: Arabic script. The initial setup is not that different from the Latin-script case.

```
\definefontfeature
  [husayni-whatever]
  [goodies=husayni,
   featureset=default]

\definefontsolution
  [FancyHusayni]
  [goodies=husayni,
   solution=experimental]

\definefont
  [FancyHusayni]
  [file:husayni*husayni-whatever at 24pt]
```

But here the definitions in the goodies file look way more complex. Here we have only one shrink set but multiple expansion sets.

```
local yes = "yes"
local basics = {
    analyze  = yes,
    mode     = "node",
    language = "dflt",
    script   = "arab",
}
local analysis = {
    ccmp = yes,
    init = yes, medi = yes, fina = yes,
}
local regular = {
   rlig = yes, calt = yes, salt = yes, anum = yes,
   ss01 = yes, ss03 = yes, ss07 = yes, ss10 = yes, ss12 = yes,
   ss15 = yes, ss16 = yes, ss19 = yes, ss24 = yes, ss25 = yes,
   ss26 = yes, ss27 = yes, ss31 = yes, ss34 = yes, ss35 = yes,
   ss36 = yes, ss37 = yes, ss38 = yes, ss41 = yes, ss42 = yes,
   ss43 = yes, js16 = yes,
}
local positioning = {
   kern = yes, curs = yes, mark = yes, mkmk = yes,
}
local minimal_stretching = {
    js11 = yes, js03 = yes,
}
local medium_stretching = {
    js12=yes, js05=yes,
}
local maximal_stretching= {
    js13 = yes, js05 = yes, js09 = yes,
}
local wide_all = {
    js11 = yes, js12 = yes, js13 = yes, js05 = yes, js09 = yes,
}
local shrink = {
    flts = yes, js17 = yes, ss05 = yes, ss11 = yes, ss06 = yes,
    ss09 = yes,
}
local default = {
    basics, analysis, regular, positioning,
}
```

```
return {
  name = "husayni",
  version = "1.00",
  comment = "Goodies that complement the Husayni font by prof.Hamid.",
  author = "Idris Samawi Hamid and Hans Hagen",
  featuresets = {
    default = {
      default,
    },
    minimal_stretching = {
      default,
      js11 = yes, js03 = yes,
    },
    medium_stretching = {
      default,
      js12=yes, js05=yes,
    },
    maximal_stretching= {
      default,
      js13 = yes, js05 = yes, js09 = yes,
    },
    wide_all = {
      default,
      js11 = yes, js12 = yes, js13 = yes, js05 = yes, js09 = yes,
    },
    shrink = {
      default, flts = yes,
      js17 = yes,
      ss05 = yes, ss11 = yes, ss06 = yes, ss09 = yes,
    },
   },
  solutions = {
    experimental = {
      less = {
        "shrink",
      },
      more = {
        "minimal_stretching", "medium_stretching", "maximal_stretching",
        "wide_all"
      },
    },
  },
  ...
}
```

There are some 55 stylistic and 21 justification features. Not all make sense when optimizing. We predefine some Lua tables to make the sets and solutions easier to understand. The default rendering looks as follows:

```
\FancyHusayni
\righttoleft
\definefontfeature[rasm][script=arab,ss05=yes,js06=no,ss55=yes]
\addff{rasm}
\getbuffer[sample]
\par
```

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ كُلُواْ مِن طَيِّبَٰتِ مَا رَزَقْنَٰكُمْ وَٱشْكُرُواْ لِلَّهِ
إِن كُنتُمْ إِيَّاهُ تَعْبُدُونَ ﴿١٧٢﴾ إِنَّمَا حَرَّمَ عَلَيْكُمُ ٱلْمَيْتَةَ وَٱلدَّمَ وَلَحْمَ
ٱلْخِنزِيرِ وَمَآ أُهِلَّ بِهِۦ لِغَيْرِ ٱللَّهِ ۖ فَمَنِ ٱضْطُرَّ غَيْرَ بَاغٍ وَلَا عَادٍ فَلَآ
إِثْمَ عَلَيْهِ ۚ إِنَّ ٱللَّهَ غَفُورٌ رَّحِيمٌ ﴿١٧٣﴾ إِنَّ ٱلَّذِينَ يَكْتُمُونَ مَآ أَنزَلَ
ٱللَّهُ مِنَ ٱلْكِتَٰبِ وَيَشْتَرُونَ بِهِۦ ثَمَنًا قَلِيلًا ۙ أُوْلَٰٓئِكَ مَا يَأْكُلُونَ فِى
بُطُونِهِمْ إِلَّا ٱلنَّارَ وَلَا يُكَلِّمُهُمُ ٱللَّهُ يَوْمَ ٱلْقِيَٰمَةِ وَلَا يُزَكِّيهِمْ وَلَهُمْ
عَذَابٌ أَلِيمٌ ﴿١٧٤﴾ أُوْلَٰٓئِكَ ٱلَّذِينَ ٱشْتَرَوُاْ ٱلضَّلَٰلَةَ بِٱلْهُدَىٰ وَٱلْعَذَابَ
بِٱلْمَغْفِرَةِ ۚ فَمَآ أَصْبَرَهُمْ عَلَى ٱلنَّارِ ﴿١٧٥﴾ ذَٰلِكَ بِأَنَّ ٱللَّهَ نَزَّلَ ٱلْكِتَٰبَ
بِٱلْحَقِّ ۗ وَإِنَّ ٱلَّذِينَ ٱخْتَلَفُواْ فِى ٱلْكِتَٰبِ لَفِى شِقَاقٍۭ بَعِيدٍ ﴿١٧٦﴾

Note that we already have a degree of widened substitution in this example. This is all for the accommodation of vowels, and is defined entirely in the OpenType tables of the font. We also added some special orthography (the `rasm` font feature to get the Qur'ānic features just right). You can also do this by adding the feature to the `lfg` file (`local regular = `). There is no paragraph optimization as yet, although the default LuaTEX engine does a good job to start with.

Next we show a more optimized result:

```
\setupfontsolution
  [FancyHusayni]
  [method={preroll,normal},
   criterium=1]

\startfontsolution[FancyHusayni]
  \FancyHusayni
  \righttoleft
\definefontfeature[rasm][script=arab,ss05=yes,js06=no,ss55=yes]
\addff{rasm}
  \getbuffer[sample]
  \par
\stopfontsolution
```

يَـٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ كُلُواْ مِن طَيِّبَـٰتِ مَا رَزَقْنَـٰكُمْ وَٱشْكُرُواْ لِلَّهِ
إِن كُنتُمْ إِيَّاهُ تَعْبُدُونَ ﴿١٧٢﴾ إِنَّمَا حَرَّمَ عَلَيْكُمُ ٱلْمَيْتَةَ وَٱلدَّمَ وَلَحْمَ
ٱلْخِنزِيرِ وَمَآ أُهِلَّ بِهِۦ لِغَيْرِ ٱللَّهِ ۖ فَمَنِ ٱضْطُرَّ غَيْرَ بَاغٍ وَلَا عَادٍ فَلَآ
إِثْمَ عَلَيْهِ ۚ إِنَّ ٱللَّهَ غَفُورٌ رَّحِيمٌ ﴿١٧٣﴾ إِنَّ ٱلَّذِينَ يَكْتُمُونَ مَآ أَنزَلَ
ٱللَّهُ مِنَ ٱلْكِتَـٰبِ وَيَشْتَرُونَ بِهِۦ ثَمَنًا قَلِيلًا ۙ أُو۟لَـٰٓئِكَ مَا يَأْكُلُونَ فِى
بُطُونِهِمْ إِلَّا ٱلنَّارَ وَلَا يُكَلِّمُهُمُ ٱللَّهُ يَوْمَ ٱلْقِيَـٰمَةِ وَلَا يُزَكِّيهِمْ وَلَهُمْ
عَذَابٌ أَلِيمٌ ﴿١٧٤﴾ أُو۟لَـٰٓئِكَ ٱلَّذِينَ ٱشْتَرَوُا۟ ٱلضَّلَـٰلَةَ بِٱلْهُدَىٰ وَٱلْعَذَابَ
بِٱلْمَغْفِرَةِ ۚ فَمَآ أَصْبَرَهُمْ عَلَى ٱلنَّارِ ﴿١٧٥﴾ ذَٰلِكَ بِأَنَّ ٱللَّهَ نَزَّلَ ٱلْكِتَـٰبَ
بِٱلْحَقِّ ۗ وَإِنَّ ٱلَّذِينَ ٱخْتَلَفُوا۟ فِى ٱلْكِتَـٰبِ لَفِى شِقَاقٍۭ بَعِيدٍ ﴿١٧٦﴾

Now let's see what happens when `\parfillskip= 0pt`, i.e., the last line has no extra space after the end of the paragraph. This is important for getting, e.g., the last line of the page to end with the end of a verse as we discussed earlier:

```
\setupfontsolution
  [FancyHusayni]
  [method={preroll,normal},
   criterium=1]

\startfontsolution[FancyHusayni]
  \FancyHusayni
  \righttoleft
\definefontfeature[rasm][script=arab,ss05=yes,js06=no,ss55=yes]
\addff{rasm}
  \parfillskip=0pt
  \getbuffer[sample]
  \par
\stopfontsolution
```

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ كُلُواْ مِن طَيِّبَٰتِ مَا رَزَقْنَٰكُمْ وَٱشْكُرُواْ لِلَّهِ
إِن كُنتُمْ إِيَّاهُ تَعْبُدُونَ ﴿١٧٢﴾ إِنَّمَا حَرَّمَ عَلَيْكُمُ ٱلْمَيْتَةَ وَٱلدَّمَ وَلَحْمَ
ٱلْخِنزِيرِ وَمَآ أُهِلَّ بِهِۦ لِغَيْرِ ٱللَّهِ ۖ فَمَنِ ٱضْطُرَّ غَيْرَ بَاغٍ وَلَا عَادٍ فَلَآ
إِثْمَ عَلَيْهِ ۚ إِنَّ ٱللَّهَ غَفُورٌ رَّحِيمٌ ﴿١٧٣﴾ إِنَّ ٱلَّذِينَ يَكْتُمُونَ مَآ أَنزَلَ
ٱللَّهُ مِنَ ٱلْكِتَٰبِ وَيَشْتَرُونَ بِهِۦ ثَمَنًا قَلِيلًا ۙ أُوْلَٰٓئِكَ مَا يَأْكُلُونَ فِى
بُطُونِهِمْ إِلَّا ٱلنَّارَ وَلَا يُكَلِّمُهُمُ ٱللَّهُ يَوْمَ ٱلْقِيَٰمَةِ وَلَا يُزَكِّيهِمْ وَلَهُمْ
عَذَابٌ أَلِيمٌ ﴿١٧٤﴾ أُوْلَٰٓئِكَ ٱلَّذِينَ ٱشْتَرَوُاْ ٱلضَّلَٰلَةَ بِٱلْهُدَىٰ وَٱلْعَذَابَ
بِٱلْمَغْفِرَةِ ۚ فَمَآ أَصْبَرَهُمْ عَلَى ٱلنَّارِ ﴿١٧٥﴾ ذَٰلِكَ بِأَنَّ ٱللَّهَ نَزَّلَ ٱلْكِتَٰبَ
بِٱلْحَقِّ ۗ وَإِنَّ ٱلَّذِينَ ٱخْتَلَفُواْ فِى ٱلْكِتَٰبِ لَفِى شِقَاقٍۭ بَعِيدٍ ﴿١٧٦﴾

Just as the effects are more visible in the `\parfillskip=0pt` case, the impact is much larger when the available width is less. In figures 11, 12, 13, 14 and 15 we can see the optimizer in action when that happens.

In our estimation, the current experimental solution works best for alternate-shaped glyphs, although there is some success with naturally widened characters. Clearly, some widened substitutions work better than others. A lot of fine tuning is needed, both within the OpenType features as well as the optimization algorithm.

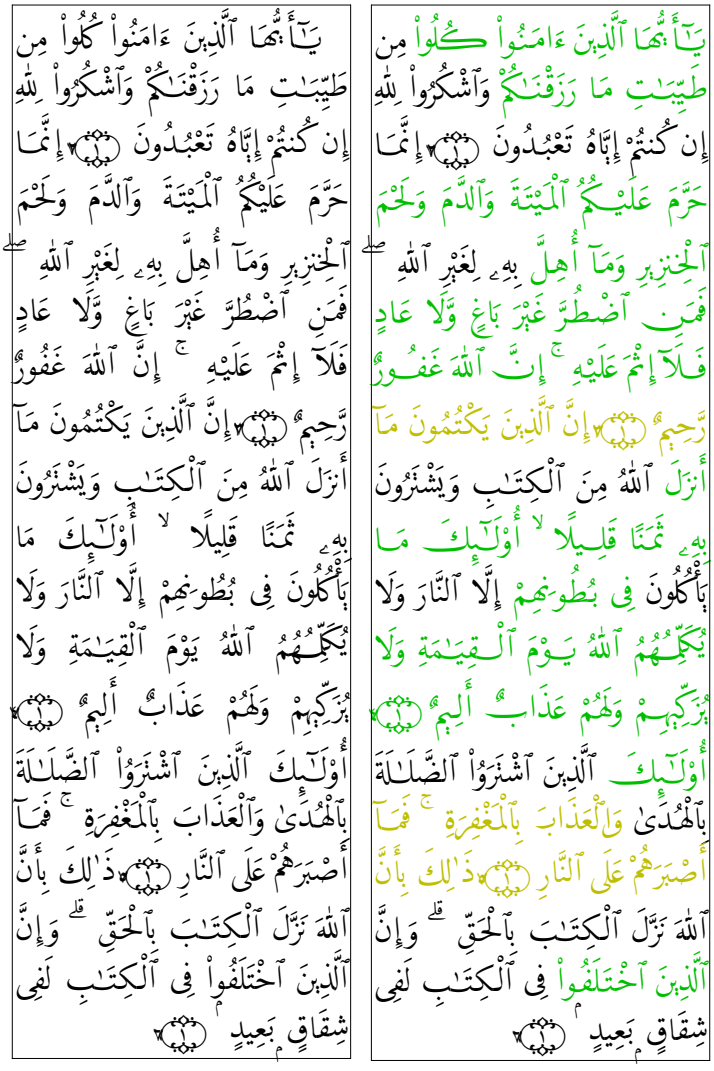

normal narrow

Figure 11. A narrower sample (a).

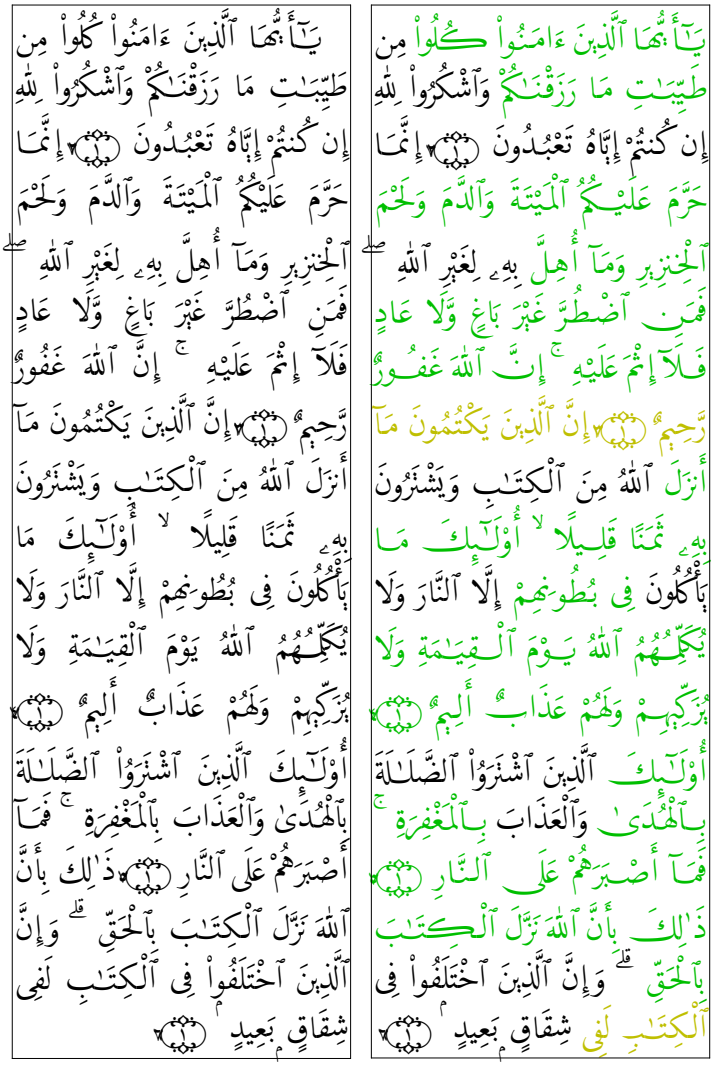


Figure 12. A narrower sample with no parfillskip (b).

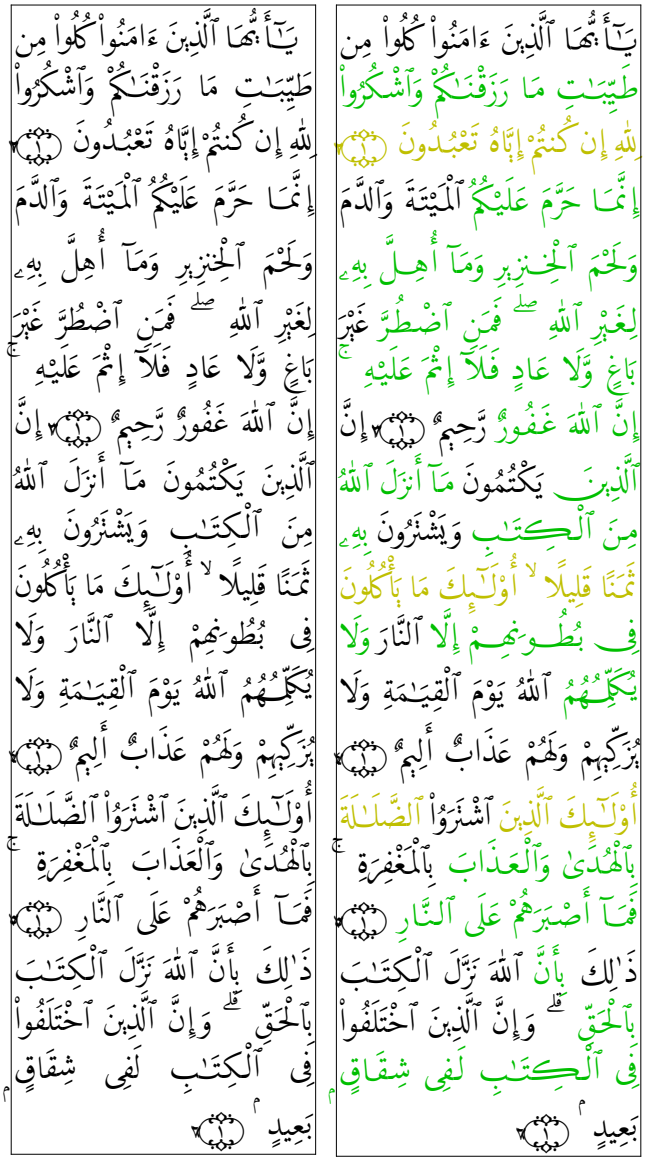


Figure 13. An even narrower sample (c).

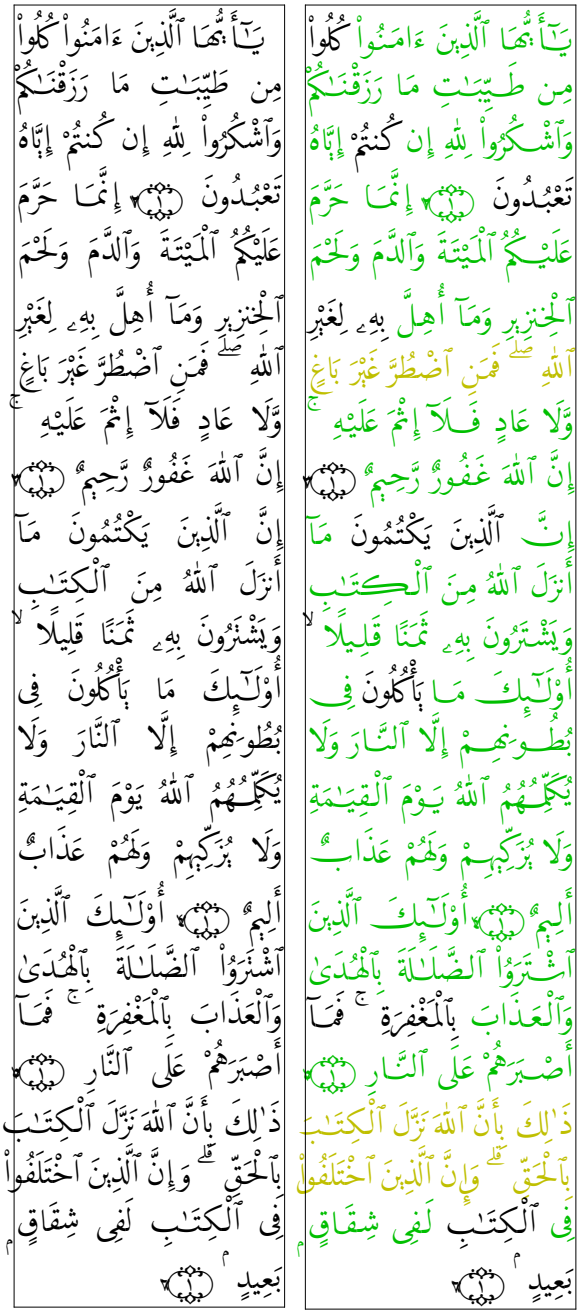


Figure 14. An even narrower sample (d).

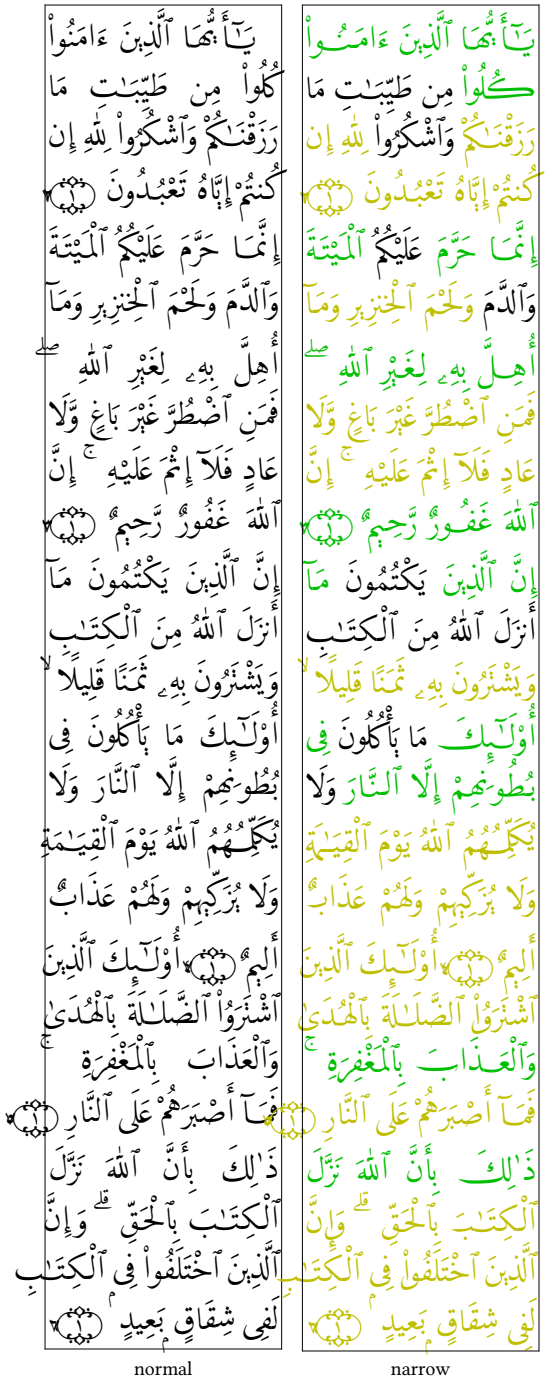


Figure 15. An even narrower sample (e).

Without going into a detailed analysis at the moment, we restrict ourselves to two critical observations.

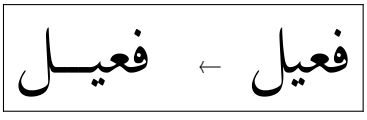
First, in our tests one will notice that the glyph substitutions tend to take place on the right side of the line. They should be more evenly distributed throughout each line.

Second, we can say that the current method works better for alternate-shaped glyph substitution than it does for naturally-widened glyph substitution. This leads us to the next step in this research project:

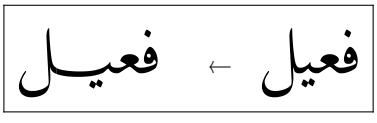
Within the Husayni font there is now a mapping between flat extending via *taṭwīl* and curved widening via alternate glyphs. Consider the following manually typed utf text using the *taṭwīl* character (U+0640):

فعيـــــل \ARROW\ فعيل

In flat-extended typography that comes out like this:



Husayni, through the optional Stylistic Alternates feature (salt) will map the flat *taṭwīl*-extended characters to curved widened characters. So with salt=yes selected in ConTEXt we get



This opens up a way to connect a forthcoming solution to the flat *taṭwīl*-extended character method with the curved widened-glyph method. A future version of the optimizer may be able to optimize the paragraph in terms of the *taṭwīl* character and a set of rules along the lines we discussed earlier. Then we can simply convert the result to curves using the *taṭwīl* character. At least this is one possibility.

In any case, the current paragraph optimizer, even in its experimental status at the moment, represents one of the greatest and most important steps in the evolution of digital Arabic-script typography. Its potential impact on for Arabic-script typesetting is immense, and we excitedly look forward to its completion.

Notes

1. Sometimes hz-optimization also goes under the rubric of 'Semitic justification'. See, e.g., Bringhurst in pre-3rd editions of his *Elements of Typographic Style*. This technique does not work well for Arabic script in general because glyphs are connected in two dimensions. On the other hand, a certain basic yet ubiquitous Semitic justification *can* be achieved by using the *taṭwīl* character, commonly called the *kashīdah* (U+0640). We will discuss this later in this article.
2. Much of this is handled within the GPOS features of the OpenType font itself (e.g., mark and mkmk)
3. This number reflects the maximum badness and future versions might have a different measure with more granularity.
4. Indeed, even Latin hyphenation, when it occurs, can be considered a 'failure' of sorts.
5. This five character string can be represented in Latin by the five character string '*al-ḥmd*' (not including the '-'). It is pronounced '*al-ḥamdu*'. Note that Arabic script is mainly consonantal: pure vowels are not part of the alphabet and are, instead, represented by diacritics.
6. Indeed, as was the case with Latin typography, Arabic-script typography took a sharp turn for the worse with the advent of digital typography. On the other hand, Latin typography recovered much more quickly, in large part thanks to Knuth's development of TEX.

MlbibTEX and Its New Extensions

Dedication
I dedicate this article to my late father (1922–2012). When I was a child, he introduced me to the joy of reading. He was himself an avid reader; I surely share this feature with him.

Abstract
These last years, MlbibTEX's kernel functions have been reused and extended in order to put new programs about bibliographies into action. Examples are the `hal` program, allowing an open archive site to be populated, the `mlbiblatex` program, building bibliographies suitable for the biblatex package, the `mlbibcontext` program, doing the same task for ConTEXt documents. We show how all these programs are organised, and explain how some operations can be refined or extended. For a point of view related to efficiency, the programs `mlbiblatex` and `mlbibcontext` are written using Scheme only, so they are more efficient than analogous programs that would interpret a .bst bibliography style of bibTEX.

Keywords
bibTEX, MlbibTEX, `mlbibtex2xml`, `mlbiblatex`, `mlbibcontext`, LATEX, ConTEXt MkII, ConTEXt MkIV, LuaTEX, biblatex package, `bib` module

Introduction

LATEX [23] is rightly viewed as a wonderful word processor for typesetting written documents. Besides, it is assisted by other programs like bibTEX [24] as bibliography processors which generate 'References' sections (.bbl files), or other graphical tools [4]. As a proof that TEX's community of developers is very dynamic, many programs—including LATEX itself—have evolved and been improved for many years. Other formats based on TEX or engines related to it have come out: e.g., XeTEX [19], LuaTEX [7]. We can observe analogous dynamism about graphical tools: compare the two editions of *The LATEX Graphics Companion*, [5] and [4].

As we mentioned in [16], bibTEX was unrivalled as the bibliography processor usually associated with LATEX for a long time. Besides, bibTEX is stable for many years. In fact, some slight extensions, built out of bibTEX's source files, have been designed, e.g., bibTEX8 [23, § 13.1.1] and bibTEXu [29, § 4.3] (see [16] for more details). The difficulty of writing a new bibliography processor from scratch is mainly related to bibliography database files. Many LATEX users have a *huge* number of .bib files, according to the format used by bibTEX. So a new bibliography processor designed to work in conjunction with LATEX should be able to deal with this format. At first glance, it is not very complicated, entries' metadata are given using the syntax '`KEY = value`', as you can see in Fig. 1. In reality, this format is more subtle. For example, values may be surrounded by double quotes:

```
@BOOK{holmstrom2011,
        AUTHOR = {Darwin Holmstrom},
        TITLE = {Toxic Terrain},
        SERIES = {Don Pendleton's The
                  Executioner},
        NUMBER = 390,
        PUBLISHER = {Gold Eagle},
        TOTALPAGES = 192,
        YEAR = 2011,
        MONTH = may}
```

Figure 1. Example using bibTEX's format.

```
TITLE = "Villa Vortex"
```

in which case a double quote character used within such a value must be surrounded by braces:

```
TITLE = "Die Energiej{\"a}ger"
```

Values may also be surrounded by braces[1]:

```
TITLE = {Grande Jonction}
```

in which case a double quote character can be used alone within such a value:

```
TITLE = {Murcos Verm\"achtnis}
```

The syntax for person names—see [10] for more details—is accurate for simple cases, but may be surprising in such a case:

```
AUTHOR = {Jean {Le Clerc de la Herverie}}
```

(if you remove the braces surrounding 'Le Clerc de la Herverie', that causes 'Herverie' to be viewed as the

```
<book id="holmstrom2011" from="mb.bib">
  <author>
   <name>
    <personname>
     <first>Darwin</first>
     <last>Holmstrom</last>
    </personname>
   </name>
  </author>
  <title>Toxic Terrain</title>
  <publisher>Gold Eagle</publisher>
  <number>390</number>
  <series>
   Don Pendleton's The Executioner
  </series>
  <totalpages>192</totalpages>
  <year>2011</year>
  <month><may/></month>
</book>
```

(The `from` attribute of the `book` element is set to the base name of the .bib file originally containing this entry.)

Figure 2. Fig. 1's example given using XML syntax.

last name, 'Jean Le Clerc' as the first name, and 'de la' as a particle). In addition, many users get used to insert LaTeX commands inside values of bibTeX fields:

```
TITLE = {\em Babylon Babies}
```

what would be difficult to interpret for a converter into a language used to put Web pages into action. Moreover, such a declaration:

```
TITLE = {\emph{CosmosIncorporated}}
```

yields a title's specification which would be correctly interpreted by LaTeX, but ConTeXt [6] would not recognise the `\emph` command.

In other words, it is quite easy to transform the syntax '`KEY = value`' into '`<KEY>value</KEY>`' if we adopt XML[2]-like syntax, or '`(KEY value)`' if Lisp[3]-like syntax is preferred. On the contrary, destructuring fields' values may be more complicated. That is why you can find many converters from .bib files into other formats, but at the first level. Roughly speaking, only a few programs run the risk of analysing the contents of fields' values.

Let us recall that we have developed MlbibTeX[4] [9], as a 'better' bibTeX with particular focus on multilingual features. As part of this task, we put into action an analysis of the values associated with bibTeX fields, as deeply as possible. We have precisely designed an internal format for bibliographical items. Later, we were asked for a program populating an open-archive site from the entries of .bib files [14,15].Although this program needed conventions more precise than usually about .bib files, we succeeded in developing it quickly. More precisely, they have many fragments in common, and the different parts were easily assembled. We decided to do again this kind of experiment... and succeeded again. First we explain how MlbibTeX can be extended. Second we recall some advantages of using MlbibTeX's kernel. Then we sketch the variants of MlbibTeX out.

MlbibTeX's extensibility

When MlbibTeX's parser processes a .bib file, we can consider that it builds an XML tree of this file. More precisely, this program written using Scheme [18] builds expressions according to the SXML[5] format [20]. For example, Fig. 1's entry is translated to the XML tree given in Fig. 2. We can see that the author's name has been split into these components. Likewise, LaTeX commands—e.g., `\em` or `\emph`—are recognised and replaced by XML tags.

When bibTeX users begin to run MlbibTeX, the most surprising feature is that the latter performs a more precise analysis of .bib files. When a field name is not recognised, a warning message is emitted[6]. By default, the fields subject to additional check are:

- the standard fields `AUTHOR`, `EDITOR`, `MONTH`, `PAGES`, and `YEAR`;
- the field `DAY`, used by numerous styles[7];
- the fields `GENDER` and `TOTALPAGES`, used by the bibliography styles associated with the jurabib package [23, § 12.5.1];
- two special fields used by MlbibTeX: `LANGUAGE` [9] and `LASTSORTKEY` [12].

The second extension of MlbibTeX—as abovementioned, the `hal` program, populating an open-archive site from the entries of .bib files [14]—needs additional check about the `ADDRESS` field of an entry being type `@INPROCEEDINGS`: we have to extract the country of the corresponding conference, and optionally the town. In addition, the name of such a country is to be checked, because we have to give its ISO[8] code. So we have decided to accept declarations like:

```
ADDRESS = {Breskens, The Netherlands}
```

or '`ADDRESS = {The Netherlands}`'. If the country is not given—e.g., in '`ADDRESS = {New-York}`' or:

```
ADDRESS = {Paris, Texas}
```

—an error has to be reported[9]. So we implemented a switch mechanism that allowed us to perform a

'classical' check about this ADDRESS field when 'original' MlbibTEX was running, and 'complete' check when this program related to open archives was used[10]. Symmetrically, disabling some check procedures would be possible within other variants. When MlbibTEX's functions work in interpreted mode, such a switch can be controlled by means of Scheme functions.

Later, we noticed the *modus operandi* of the biblatex package [21]: .bbl files only contain *structures*, and formatting 'References' sections is entirely deferred to LATEX. That is why there is no need of a \bibliographystyle command. If bibTEX is used, there is only one suitable bibliography style written using bibTEX's language. Another bibliography processor, biber [1], has come out: it builds only .bbl files suitable for biblatex. Let us consider the example of a LATEX document using this biblatex package given in Fig. 3. The corresponding .bbl file looks like Fig. 4, and the bibliography will be formatted w.r.t. the author-date style [23, § 12.3], because of the bibstyle option of the biblatex package.

```
\documentclass{article}

\usepackage[bibstyle=authoryear]{biblatex}
\addbibresource{mb.bib}  % The suffix is needed.

\begin{document}

Did you read \citetitle*{holmstrom2011}? This is
a thriller written by \citeauthor{holmstrom2011}.

\printbibliography

\end{document}
```

Figure 3. Using the biblatex package.

```
\entry{holmstrom2011}{book}{}
  \name{author}{1}{}{%
    {{uniquename=0}{Holmstrom}{H.}{Darwin}%
     {D.}{}{}{}{}}%
  }%
  \field{title}{Toxic Terrain}%
  \list{publisher}{1}{{Gold Eagle}}%
  \field{number}{390}%
  \field{series}{Don Pendleton's The Executioner}%
  \field{totalpages}{192}%
  \field{year}{2011}%
  \field{month}{05}%
\endentry
```

Figure 4. Reference used by the biblatex package.

The biblatex package's conceptor introduced new entry types a bibliography processor should be able to process. On the contrary, these new types are unknown in standard bibliography styles. Again, a switch mechanism allows us to recognise these new types only when the parser is running in a kind of 'mlbiblatex mode'. Another point is related to *dates*: in standard bibliography styles, they are specified by a YEAR field and optionally by a MONTH field. The biblatex package allows dates to be expressed this way, or by means of a DATE field allowing the specification of a *range* of dates [21, § 2.3.8]. The extension of our parser for biblatex has been revised to include these points. Let us mention that the specification of dates is crucial within bibliographies since they are used for the sort operation in most styles. A last point: the syntax of the PAGES field has been refined.

A framework similar to biblatex had been put into action by Taco Hoekwater's bib module of ConTEXt [8]: see Fig. 5 for a source text using a bibliographical reference. This reference, as it should be produced by a bibliography processor, is given in Fig. 6. The bib module can be used with ConTEXt MkII [2], it has been reimplemented in ConTEXt MkIV by Hans Hagen [3]. In this last case, the switch we installed considers a new @CONTEXTPREAMBLE directive when a .bib file is parsed. This directive aims to replace the 'traditional' @PREAMBLE directive, often used to put definitions of new LATEX commands [23, § 13.2.4]. This @CONTEXTPREAMBLE directive can be used to program some LATEX commands put throughout .bib files and non-existing in ConTEXt.

```
\usemodule[bib] % Needed for MkII, not for MkIV

\setupbibtex[database=mb]
\setuppublications[numbering=yes]

\starttext

Did you read \cite[holmstrom2011]?

\placepublications

\stoptext
```

Figure 5. Citations and bibliographies in ConTEXt.

```
\startpublication[k=holmstrom2011,
 t=book,a={{Holmstrom}},y=2011,n=2,s=Hol11]
\author[]{Darwin}[D.]{}{Holmstrom}
\pubyear{2011}
\title{Toxic Terrain}
\series{Don Pendleton's The Executioner}
\volume{390}
\pubname{Gold Eagle}
\month{5}
\stoppublication
```

Figure 6. Reference used by ConTEXt.

```
(<english? "ConTeXt" "ConTeXt")                                    ⟹  #f
(<english? "ConTeXt" "ConTeXt" (lambda () #f) < 'uppercase-1st)    ⟹  #f  ; Default values explicited.
(<english? "ConTeXt" "ConTeXt" (lambda () 'ok))                    ⟹  ok  ; Equal strings.
(<english? "ConTeSt" "ConTeXt")                                    ⟹  #t
(<english? "ConTeSt" "ConTeXt" (lambda () 'ok) >)                  ⟹  #f  ; Descending order.
(<english? "ConText" "ConTeXt" (lambda () 'ok))                    ⟹  #f
(<english? "ConText" "ConTeXt" (lambda () 'ok) < #f)               ⟹  ok  ; Case-insensitive equality.
(<english? "ConText" "ConTeXt" (lambda () 'ok) < 'lowercase-1st)   ⟹  #t  ; Lowercase letters take
                                                                           ; precedence.
(<english? "ConTeXt" "ConTeSt"
           (lambda ()
               (<english? "Mk" "Mk" (lambda () (<arithmetical? 2 4 (lambda () …))))) ⟹ #f
```

Figure 7. Order relations handled by MlbibTEX.

MlbibTEX's advantages

When the approach of biblatex and ConTEXt is used, a bibliography processor does not have to provide the text of successive references of a bibliography. Since it just produces structures whatever the bibliography style is—such a style is put into action by customising the command of LATEX or ConTEXt producing the final bibliography—the idea is to build two accurate bibliography processors out of MlbibTEX's kernel. These two programs —`mlbiblatex` (resp. `mlbibcontext`) for biblatex (resp. ConTEXt)—are written entirely in Scheme, in order to get more efficiency. Even if we are not interested in multilingual extensions of MlbibTEX during a first step, here are the features of interest for such bibliography processors.

Order relations

In [11], we showed how the lexicographic order relations handled by MlbibTEX were built. These order relations—implemented by means of Scheme functions—are language-dependent. A simple use of the `<english?` function—for English words—to compare two strings is given by the first example of Fig. 7—'`#t`' (resp. '`#f`') stands for the 'true' (resp. 'false') value in Scheme. In reality, these functions are more powerful since they use optional arguments—controlling the behaviour—in addition to the two strings to be compared:

- the third is a *thunk*[11] that is called if the two strings are equal;
- the fourth is < (resp. >) for an ascending (resp. a descending) order;
- the fifth is `#f` for a case-insensitive comparison, `uppercase-1st` (resp. `lowercase-1st`) if uppercase (resp. lowercase) letters take precedence when two strings are different only by the case.

Fig. 7's second example shows the default values of these three additional arguments. By default, these functions implement *strict* order relations, that is, *irreflexive*, asymmetric, and transitive; as < for numbers. The sixth example shows that our `<english?` function defaults to a case-sensitive relation in which uppercase letters take precedence over lowercase ones, the seventh example shows how to proceed if you would like lowercase letters to take precedence. Finally, the last example shows how the third argument can be used to *chain* order relations[12]: the idea is to sort persons regarding last names, first names, birth dates, and possibly other information. As you can see, this feature—sketched in [12, § 4]—makes a sort easier by means of several successive sort keys. More details about these order relations are given in [17].

Syntactical extensions

MlbibTEX's syntactical extensions about multilinguism are explained in detail in [9]. Presently, they are not used by the programs `mlbiblatex` and `mlbibcontext`. On the contrary, our extensions for authors' and editors' names can be directly usable by these two programs. In addition to bibTEX's conventions, *keywords* may be used to point to the four parts—*First*, *von*, *Last*, *Junior*—of a name, what may be very useful:

```
first => Jean, last => Le Clerc de la Herverie
```

(the four keywords '`first =>`', '`von =>`', '`last =>`', '`junior =>`' are available, the order of appearance being irrelevant). In addition, the '`abbr =>`' keyword may be used when a first name is not abbreviated according to the standard way, that is, retaining only the first letter. If an organisation's name is used as an author or editor, you can use the keywords '`org =>`' for the name as it must be typeset and '`sortingkey =>`' for the key used for sorting:

```
org => Euro\TeX~2012,
sortingkey => EuroTeX 2012
```

It is well-known that co-authors are connected by

means of the ‘ and ’ keyword. MlbibTEX also allows the specification of *collaborators*, by means of the ‘ with ’ keyword; an example is given in this article's bibliography: see the reference [23].

MlbibTEX's programs

MlbibTEX's distribution is located at:

http://lifc.univ-fcomte.fr/home/~jmhufflen/texts/superreport/smlbibtex-1.3.tar.gz

The easiest way to install it is to compile the source files by the bigloo [25] Scheme compiler; the installation procedure [17] uses the commands `configure` [28] and `make` [22], well-known within GNU[13] software; more details are given in [17, § 4.2]. The executable programs generated are described hereafter. The complete distribution's version number is given ‘classically’, that is, by means of sequence of numbers. Versions of particular variants are labelled by geographical names. Those demonstrated at the EuroTEX 2012 conference are ‘Breskens versions’.

`mlbibtex`
This programs aims to replace bibTEX and is described in [9]; you can use it analogously to ‘original’ bibTEX. This `mlbibtex` is the ‘historical’ origin of the present toolbox.

`mlbibtex2xml`
This program allows .bib files to be converted into XML files, according to the format internally used by MlbibTEX. You can run it as follows:

```
mlbibtex2xml ([-screen] | [-o output]) \
  f0.bib f1.bib ...
```

where `f0.bib`, `f1.bib`, ...—the .bib suffix can be omitted—are .bib files. If the `-screen` option is used, the result is displayed at the screen, otherwise it is written into a file. If the `-o` option is used, `output` gives the output file name, otherwise, this name defaults to `f0-mlbiblio.xml`, even if several .bib files are processed. Obviously, results look like Fig. 2.

`ar-style` **and** `hal`
These two programs are the first two extensions of MlbibTEX. The `ar-style` program can be used for activity reports' bibliographies, when they have to be conformant to the classification of the French agency AERES[14] [13]. See Section ‘MlbibTEX's extensibility’ and [14,15] about the `hal` program.

`mlbiblatex`
The `mlbiblatex` program builds .bbl files suitable for the biblatex package. You can run it as follows:

```
mlbiblatex filename.aux key-expr lg-code
```

where:

`filename.aux`
: —the `.aux` suffix can be omitted—is the auxiliary file where the information about bibliographical keys and database files has been stored;

`key-expr`
: gives successive sort keys, according to the pattern `(m | n | t | y)*`, where ‘`m`’, ‘`n`’, ‘`t`’, ‘`y`’ respectively stand for ‘**M**onth’[15], ‘**N**ame’ (person name as an author or editor), ‘**T**itle’, ‘**Y**ear’; all the other signs are ignored; there is no default order relation[16]: if no sign is recognised, the list of bibliographical items is left unsorted[17];

`lg-code`
: is the code for the language to be used for sorting strings—this information is relevant whenever person names and titles of works are compared—available values are `DE` for German, `EN` for English, `FR` for French, `PO` for Polish; there is no default value.

Results look like Fig. 4. More detais are given in [17].

`mlbibcontext`
The `mlbibcontext` program builds .bbl files suitable for ConTEXt. The corresponding command line looks like `mlbiblatex`'s:

```
mlbibcontext filename.aux key-expr lg-code
```

and `filename.aux`, `key-expr`, `lg-code` have the same meaning. Results look like Fig. 6.

Future directions

As we mention above, the interface between the functions of a word processor in charge of processing ‘References’ sections—the commands of the biblatex package or ConTEXt MkIV—could be improved. For example, the commands `mlbiblatex` and `mlbibcontext` only deal with ascending orders. This is just related to the rough interface we designed in order to propose first experimental versions of these programs: as shown in Section ‘MlbibTEX's advantages’, descending orders are provided by MlbibTEX's kernel. Concerning the biblatex package, we think that an option could be added[18]:

```
\usepackage
  [backend=mlbiblatex,...]%
  {biblatex}
```

other options allowing accurate information to be passed to MlbibTEX.

Likewise, ConTEXt MkIV users should be able to choose between bibTEX—or an ‘enriched’ bibTEX such that bibTEX8 or bibTEXu—or MlbibTEX. In this last case, we have to study how accurate information

could be passed to the `mlbibcontext` program.

Some present lack of MlbibTEX: only two encodings are available, for input .bib files as well as output .bbl ones. More precisely, .bib files are supposed to be encoded w.r.t. Latin 1. The characters that are not included in this encoding—e.g., some Polish letters, such that 'ł'—can be reached only by using TEX commands—like '\l'[19]. About generated .bbl files, either MlbibTEX detects that the Latin 1 encoding is used by looking into the document's preamble[20], in which case this encoding is used for the .bbl file produced; otherwise, this .bbl file is a pure ASCII[21] file, all the accented letters being specified by means of TEX commands[22]. Such behaviour is due to the Scheme programming language. MlbibTEX has been written using the fifth revision of this language [18], not Unicode-compliant. Most of Scheme interpreters can deal with Latin 1, some—not all—accept other encodings, but in a non-portable way. Besides, we want our functions to be able to work on as many Scheme interpreters as possible. A new revision of Scheme is in progress[23] and will be Unicode-compliant, so a future version of MlbibTEX should be able to deal with other encodings such that Latin 2, UTF-8, UTF-16, etc.

Last but not at least, we plan to update the programs `mlbiblatex` and `mlbibcontext`, in order for them to be able to deal with MlbibTEX's multilingual features. From our point of view, that should be quite easy for `mlbibcontext`, in the sense that all the languages are available *a priori* within ConTEXt MkIV—you do not have to put all the languages you use throughout a text as options of a module like the babel package [23, Ch. 9]—but might require more work for the texts to be processed by the commands of the biblatex package.

Conclusion

We are personally an adept of functional programming in general and Scheme in particular. But MlbibTEX has been able to be adapted to applications other than those initially planned, what is a good quality for a program[24]. In particular, the `mlbiblatex` program succeded in taking as much advantage as possible of biblatex's features[25] with just slight modifications of our kernel. We think that we have been able to reach such adaptability and flexibility because of the use of Scheme, even if these qualities could have been reached within other programming paradigms[26]. In addition, our programs can be used with a Scheme interpreter, but better efficiency is reached if programs are compiled. Even if we think that we are not in competition with a bibliography processor like biber, it is certain that a program written using Scheme is more efficient than a program written using Perl[27]. So we have spent much time when we began MlbibTEX's development, but we do not regret anything and were happy to be able to adapt this program to new requirements.

Notes

1. Personally, we always recommend users to adopt this convention, simpler, from our point of view.
2. e**X**tensible **M**arkup **L**angage.
3. **LIS**t **P**rocessor.
4. **M**ulti**L**ingual bibTEX.
5. Scheme implementation of **XML**.
6. This is just a warning message; the corresponding information is not lost. This *modus operandi* may be viewed as an advantage: for example, if you inadvertently type '`EDITORS = ...`' instead of '`EDITOR = ...`' inside an entry of type `@INPROCEEDINGS`, MlbibTEX will warn you whereas bibTEX will silently ignore that field. This feature may also be viewed as a drawback: if you specify a `MONTH` field, the associated value must be a symbol among `jan`, `feb`, ..., `dec`. Otherwise, MlbibTEX stops with an error message. This convention may appear as too restrictive, but MlbibTEX can sort w.r.t. month names, whereas bibTEX does not. To perform such an operation, month names must be recognised. Likewise, when years are to be sorted, MlbibTEX applies a numerical sort whereas bibTEX sorts years as strings, so the value associated with a `YEAR` field must be an integer.
7. For example, the styles '`apa...`', used by the American Psychology Association.
8. **I**nternational **S**tandardisation **O**rganisation.
9. We also accept declarations like: `ADDRESS = {Washington, District of Columbia, United States}` that is, a string of three comma-separated components. The first is supposed to be the town, the last the city.
10. Technically, it is not very difficult since we consider that Scheme—as a functional programming language—allows functions to be handled like any other value. MlbibTEX's parser uses *association lists* whose elements look like `(key . f)` where `f` is the function to be called to parse the value associated with `key`. To perform such a switch, just change the function associated with `key`.
11. A zero-argument function, w.r.t. Scheme's terminology.
12. The `arithmetical?` function, used within Fig. 7's last example is analogous to our order relations, in the sense that its third argument is called if the two numbers given as first two arguments are equal. Otherwise it behaves like `<`.
13. Recursive acronym: **G**NU is **N**ot **U**NIX.
14. ***A**gence d'**É**valuation de la **R**echerche et de l'**E**nseignement **S**upérieur*, that is, 'agency evaluating research and university courses'.
15. ... an item without month information being ranked after an item with such.
16. The default order relation used by both bibTEX and biber would be specified by `ynt`. Let us recall that by default, these two programs do not use any information about month during the sort step.
17. In this case, the bibliography is *unsorted*, that is, the order of items is the order of first citations of these items throughout the document.

18. Presently, the possible values for the backend option of biblatex are 'bibtex', 'bibtex8', 'bibtexu', 'biber'.
19. For example, the name of the Polish city 'Łódź' should be written down '{\L}\'{o}d\'{z}' or '{\L}ód\'{z}' within a .bib file, its internal form handled by MlbibTEX is '{\L}ód\'{z}', since 'ó' belongs to Latin 1, whereas 'Ł' and 'ź' do not.
20. bibTEX just read .aux files and never reads a .tex file [23, § 12.1.3], whereas the mlbibtex program may look into a document's preamble.
21. **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
22. Let us recall that ConTEXt MkIV texts are supposed to be encoded w.r.t. UTF-8. Since MlbibTEX cannot deal with this encoding, the output files of the mlbibcontext program are presently encoded w.r.t. pure ASCII.
23. See the Web page http://scheme-reports.org. In fact, MlbibTEX has been implemented using the conventions of R5RS, what stands for '**R**evised[5] **R**eport on the algorithmic language **S**cheme' [18]. Later, a new revision (R6RS) was designed and ratified [26][27], including functions dealing with the whole range of Unicode and different encodings [27, §§ 1 & 2.9]—but for some reasons that we do not give here, most Scheme implementors did not update their programs. So MlbibTEX is still R5RS-compliant. It seems that Scheme's next version (R7RS)—see some drafts at the Web page abovementioned—will be adopted by most Scheme implementors. So we hope that we will be able to get a Unicode-compliant version of MlbibTEX very soon.
24. More generally, some people already announced the end of Lisp dialects, or the end of TEX & Co... and these programs are still in action.
25. Especially the notion of field *type*: for example, @AUTHOR is a *list of names*, @TITLE is a *literal*, according to the biblatex package's terminology. Analogous notions exist within MlbibTEX.
26. But we think that more effort would have been needed.
27. **P**ractial **E**xtraction and **R**eport **L**anguage. A good introduction to this language is [30].

References

[1] François Charette and Philip Kime: *biber. A Backend Bibliography Processor for biblatex. Version biber 0.9 (biblatex 1.6).* August 2011. http://freefr.dl.sourceforge.net/project/biblatex-biber/biblatex-biber/development/documentation/biber.pdf.

[2] CONTEXTGARDEN, http://wiki.contextgarden.net/Bibliography: *Bibliographies in MkII.* April 2012.

[3] CONTEXTGARDEN, http://wiki.contextgarden.net/Bibliography_mkiv: *Bibliographies in MkIV.* July 2012.

[4] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis B. Roegel and Herbert Voß: *The LATEX Graphics Companion.* 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. January 2009.

[5] Michel Goossens, Sebastian Rahtz and Frank Mittelbach: *The LATEX Graphics Companion. Illustrating Documents with TEX and PostScript.* Addison-Wesley Publishing Company, Reading, Massachusetts. March 1997.

[6] Hans Hagen: *ConTEXt, the Manual.* November 2001. http://www.pragma-ade.com/general/manuals/cont-enp.pdf.

[7] Hans Hagen: 'The Luafication of TEX and ConTEXt'. In: *Proc. BachoTEX 2008 Conference*, p. 114–123. April 2008.

[8] Taco Hoekwater: 'The Bibliographic Module for ConTEXt'. In: *EuroTEX 2001*, p. 61–73. Kerkrade (the Netherlands). September 2001.

[9] Jean-Michel Hufflen: 'MlbibTEX's Version 1.3'. TUG*boat*, Vol. 24, no. 2, p. 249–262. July 2003.

[10] Jean-Michel Hufflen: 'Names in bibTEX and MlbibTEX'. TUG*boat*, Vol. 27, no. 2, p. 243–253. TUG 2006 proceedings, Marrakesh, Morocco. November 2006.

[11] Jean-Michel Hufflen: 'Managing Order Relations in MlbibTEX'. TUG*boat*, Vol. 29, no. 1, p. 101–108. EuroBachoTEX 2007 proceedings. 2007.

[12] Jean-Michel Hufflen: 'Revisiting Lexicographic Order Relations on Person Names'. In: *Proc. BachoTEX 2008 Conference*, p. 82–90. April 2008.

[13] Jean-Michel Hufflen: *Classe superreport — Manuel d'utilisation.* March 2010. http://lifc.univ-fcomte.fr/home/~jmhufflen/superreport/superreport-readme.pdf.

[14] Jean-Michel Hufflen: 'Using MlbibTEX to Populate Open Archives'. In: Tomasz Przechlewski, Karl Berry, Gaby Gic-Grusza, Ewa Kolsar and Jerzy B. Ludwichowski, eds., *Typographers and Programmers: Mutual Inspirations. Proc. BachoTEX 2010 Conference*, p. 45–48. April 2010.

[15] Jean-Michel Hufflen: 'From Bibliography Files to Open Archives: the Sequel'. In: Karl Berry, Jerzy B. Ludwichowski and Tomasz Przechlewski, eds., *Proc. EuroBachoTEX 2011 Conference*, p. 61–66. Bachotek, Poland. April 2011.

[16] Jean-Michel Hufflen: 'A Comparative Study of Methods for Bibliographies'. TUG*boat*, Vol. 32, no. 3, p. 289–301. Proc. TUG 2011 conference. October 2011.

[17] Jean-Michel Hufflen: 'MlbibTEX and the biblatex package'. In: Tomasz Przechlewski, Karl Berry and Jerzy B. Ludwichowski, eds., *Twenty Years After. Proc. BachoTEX 2012 Conference*, p. 91–99. Bachotek, Poland. April 2012.

[18] Richard Kelsey, William D. Clinger, and Jonathan A. Rees, with 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, p. 7–105. August 1998.

[19] Jonathan Kew: 'XETEX in TEX Live and beyond'. TUG*boat*, Vol. 29, no. 1, p. 146–150. EuroBachoTEX 2007 proceedings. 2007.

[20] Oleg E. Kiselyov: XML *and Scheme.* September 2005. http://okmij.org/ftp/Scheme/xml.html.

[21] Philipp Lehman: *The biblatex Package. Programmable Bibliographies and Citations. Version 1.6.* 29 July 2011. ftp://ftp.tex.ac.uk/archive/Archive\%20directory/macros/latex/exptl/biblatex/doc/biblatex.pdf.

[22] Miki Loukides and Andy Oram: *Programming with* GNU *Software.* O'Reilly & Associates, Inc. December 1996.

[23] 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.

[24] Oren Patashnik: *bibTEXing.* February 1988. Part of the bibTEX distribution.

[25] Manuel Serrano: *Bigloo. A Practical Scheme Compiler. User Manual for Version 3.3b.* March 2010.

[26] Michael Sperber, R. Kent Dybvig, Matthew Flatt, and Anton van Straaten, with Richard Kelsey, William Clinger, Jonathan Rees, Robert Bruce Findler and Jacob Matthews: *Revised^6 Report on the Algorithmic Language Scheme*. September 2007. hhtp://www.r6rs.org.

[27] Michael Sperber, R. Kent Dybvig, Matthew Flatt, and Anton van Straaten, with Richard Kelsey, William Clinger and Jonathan Rees: *Revised^6 Report on the Algorithmic Language Scheme—Standard Libraries*. September 2007. hhtp://www.r6rs.org.

[28] Gary V. Vaughn, Ben Ellison, Tom Tromey and Ian Lance Taylor: GNU *Autoconf, Automake, and Libtool*. Sams. October 2000.

[29] Herbert Voß: *Bibliografien mit LaTeX*. Lehmans Media, Berlin. 2011.

[30] Larry Wall, Tom Christiansen and Jon Orwant: *Programming Perl*. 3rd edition. O'Reilly & Associates, Inc. July 2000.

Jean-Michel Hufflen
FEMTO-ST (UMR CNRS 6174) & University of Franche-Comté, 16, route de Gray, 25030 Besançon Cedex, France

Demonstration of the ‘mlbibcontext’ Program

Abstract
This short statement aims to sketch the broad outlines of the presentation performed at the 6th ConTEXt meeting.

Introduction

When the bibTEX bibliography processor [17] builds a ‘Reference’ section for a source text typeset by the LATEX word processor [16], it only uses information stored in auxiliary (.aux) files [16, § 12.1.3]. In particular, such an .aux file gives the *bibliography style* to be used, as a .bst file[1]. Such a style is monolithic, in the sense that nothing can be customised when bibTEX is called: for example, the order relation used to sort bibliographical items is hard-wired in any .bst file. The biber program [1]—often used in conjunction with the biblatex package [14]—is more flexible: when it runs, it uses a configuration file (.bcf[2]) file—using XML[3]-like syntax—as explained in [8, § 2.5]: in particular, such a .bcf file allows the sort of bibliographical items to be customised. However, let us recall that biber has a drawback from a point of view related to ConTEXt: it only builds ‘References’ sections suitable for the biblatex package. As explained in [10], the `mlbibcontext` program aims to build ‘References’ sections suitable for the bibliography support for ConTEXt [2,3]. The main point of the demonstration is to show which information is needed by mlbibcontext, in order for this program to be as powerful as possible. In other words, we aim to help design a nice interface between ConTEXt and `mlbibcontext`[4].

Plan

Let us recall that the `mlbibcontext` program—written entirely using the Scheme programming language [13]—builds ‘References’ sections suitable for the commands of Taco Hoekwater’s `bib` module [5], reimplemented within ConTEXt MkIV by Hans Hagen [3]. The demonstration will focus on the following points:

- its installation: the easiest way is to compile the source files by the bigloo [18] Scheme compiler[5]; the installation procedure [9] uses the commands `configure` [19] and `make` [15], well-known within GNU[6] software; the source files are available at the Web page http://lifc.univ-fcomte.fr/home/~jmhufflen/texts/superreport/smlbibtex-1.3.tar.gz;
- the `mlbibcontext` program allows order relations used to sort bibliographies to be customised w.r.t. successive keys given by bibTEX’s fields [10,11]; only ascending orders can be used presently, but this point could be improved by a nicer interface: the kernel of MlbibTEX[7] also provides descending order relations;
- the `mlbibcontext` program allows you to put many basic commands of LATEX inside values of bibTEX’s fields, even if the result is processed by ConTEXt; moreover, some commands specific to ConTEXt may be grouped into a special preamble within .bib files: the `@CONTEXTPREAMBLE` directive instead of the traditional `@PREAMBLE` directive [6].

To end up, let us mention the `mlbibtex2xml` program [10], part of MlbibTEX. This program allows bibliographical items to be given using XML-like syntax. This kind of text can be processed by ConTEXt MkIV (cf. [7, Fig. 8]). However, we think that `mlbibtex2xml`’s outputs could be processed by programs written using Lua [12]—as allowed by ConTEXt MkIV [4]—rather than ConTEXt features related to TEX. When .bib files are processed by `mlbibtex2xml`, no sort operation is performed.

Notes

1. Except if the biblatex package is used [14], in which case the bibliography style applied by bibTEX is implicitly the biblatex bibliography style.
2. **B**iber **C**onfiguration **F**ile.
3. e**X**tensible **M**arkup **L**anguage.
4. Let us mention that `mlbibcontext` could deal with configurations described by XML files—in particular, it could process

bibliographical entries given using XML-like syntax—; it can also process additional definitions written using the Scheme programming language [13].

5. Of course, it is preferable for `mlbibcontext` to be compiled, in order to get more efficiency. The use of other Scheme compilers or interpreters is possible.
6. Recursive acronym: GNU is Not UNIX.
7. MultiLingual bibTEX.

References

[1] François Charette and Philip Kime: *biber. A Backend Bibliography Processor for biblatex. Version biber 0.9 (biblatex 1.6)*. August 2011. http://freefr.dl.sourceforge.net/project/biblatex-biber/biblatex-biber/development/documentation/biber.pdf.

[2] CONTEXTGARDEN, http://wiki.contextgarden.net/Bibliography: *Bibliographies in MkII*. April 2012.

[3] CONTEXTGARDEN, http://wiki.contextgarden.net/Bibliography_mkiv: *Bibliographies in MkIV*. July 2012.

[4] Hans Hagen: 'The Luafication of TEX and ConTEXt'. In: *Proc. BachoTEX 2008 Conference*, p. 114–123. April 2008.

[5] Taco Hoekwater: 'The Bibliographic Module for ConTEXt'. In: *EuroTEX 2001*, p. 61–73. Kerkrade (the Netherlands). September 2001.

[6] Jean-Michel Hufflen: 'MlbibTEX Meets ConTEXt'. TUG*boat*, Vol. 27, no. 1, p. 76–82. EuroTEX 2006 proceedings, Debrecen, Hungary. July 2006.

[7] Jean-Michel Hufflen: 'Processing 'Computed' Texts'. MAPS, Vol. 41, p. 68–78. 2010.

[8] Jean-Michel Hufflen: 'A Comparative Study of Methods for Bibliographies'. TUG*boat*, Vol. 32, no. 3, p. 289–301. Proc. TUG 2011 conference. October 2011.

[9] Jean-Michel Hufflen: 'MlbibTEX and the biblatex package'. In: Tomasz Przechlewski, Karl Berry and Jerzy B. Ludwichowski, eds., *Twenty Years After. Proc. BachoTEX 2012 Conference*, p. 91–99. Bachotek, Poland. April 2012.

[10] Jean-Michel Hufflen: 'MlbibTEX and Its New Extensions'. In: *Proc. EuroTEX 2012*. Breskens, The Netherlands. October 2012.

[11] Jean-Michel Hufflen: *Gestion d'ordres lexicographiques multilingues avec xindy et MlbibTEX*. À paraître dans les *Cahiers GUTenberg*. 2012.

[12] Roberto Ierusalimschy: *Programming in Lua*. 2nd edition. Lua.org. March 2006.

[13] 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, p. 7–105. August 1998.

[14] Philipp Lehman: *The biblatex Package. Programmable Bibliographies and Citations. Version 1.6*. 29 July 2011. ftp://ftp.tex.ac.uk/archive/Archive%20directory/macros/latex/exptl/biblatex/doc/biblatex.pdf.

[15] Miki Loukides and Andy Oram: *Programming with* GNU *Software*. O'Reilly & Associates, Inc. December 1996.

[16] Frank Mittelbach, Michel Goossens, 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.

[17] Oren Patashnik: *bibTEXing*. February 1988. Part of the bibTEX distribution.

[18] Manuel Serrano: *Bigloo. A Practical Scheme Compiler. User Manual for Version 3.3b*. March 2010.

[19] Gary V. Vaughn, Ben Ellison, Tom Tromey and Ian Lance Taylor: GNU *Autoconf, Automake, and Libtool*. Sams. October 2000.

Jean-Michel Hufflen
FEMTO-ST (UMR CNRS 6174) & University of Franche-Comté, 16, route de Gray, 25030 Besançon Cedex, France

Abstracts without papers

Run for Fun

Jano Kula

Sports and especially long distance runs are known for the good doses of endorphin. Instead, we will show some adrenaline challenges while preparing such a sport event: 10 km run through the historic center of Prague. Plotters, tables, layers, composition.

ConTEXt: the script

Hans Hagen

The ConTEXt runner `context` has inherited a few features from its predecessor `texexec`. Instead of hardcoding functionality for typesetting listings and manipulating PDF files in the script itself they are now isolated in TEX files. In this presentation I will show some of these lesser known features of the script.

ConTEXt: after the cleanup

Hans Hagen

After the transition from MkII to MkIV a cleanup stage has been started. What is involved in this cleanup and what will happen afterwards. This is more a discussion than a presentation and users are invited to express their wishes and priorities.

Metapost workshop

Mari Voipio

'A pragmatic approach to MetaPost', or, 'How to get useful results out of MetaPost if you are not a programmer, are not a mathematician, and are a complete beginner besides.'

A couple of styles

Hans Hagen

When you keep an eye on what modules get added to ConTEXt, you will notice that quite some of them are a mixture of TEX, METAPOST and Lua. I will show a few that might have gone unnoticed. They can often serve as an example for your own local usage.

Lexing

Hans Hagen

As I use SciTE most of the time, there is some mutual influence between coding and visualization in this editor. Especially the possibility to write more advanced lexers has lead to some changes (for the good) in the code base. Here I will show some of that (as it might help those who browse the source).

(visual) debugging

Hans Hagen

Compared to MkII the MkIV code has more tracing on board. At the Lua end we have trackers and recently a start has been made to extend that to the TEX end, where it will replace the `\trace*true` like macros. As part of the cleanup the original visual debugger module has been replaced by an even less intrusive variant. It provides the usual visual clues about what goes on the page. The new mechanism is more advanced than the old one but still assumes some knowledge of what happens inside TEX. In this presentation we will explain some of this.

Japanese Typesetting with LuaTEX

KITAGAWA Hironori

There are some issues for typeset Japanese documents by LuaTEX. Some of them, such as end-of-line rule and the value of a grouped variable 'at the end of a `\hbox`', are (partially) resolved by writing Lua codes. Also we can discuss the specification of LuaTEX on vertical typesetting, referring to that of Japanese pTEX.

Mixed columns

Hans Hagen

One of the last things to redo in MkIV is the page builder. Although bits and pieces have been redone, some major effort is needed to upgrade multi columns mechanisms. We (currently) have three mechanisms: regular columns that can be mixed with single column mode, simple columns that can be used in a boxed way, and columnsets. The first two have been replaced by a new mechanism tagged as mixed columns. This mechanism permits instances of multicolumns, either or not in the page flow or in boxes, and the old mechanisms will go. Of course we try to remain compatible as much as possible. In this talk we can discuss some of the issues involved and identify future needs.

Differences in typesetting rules between Czech and Slovak languages (in the context of ConTEXt)

Tomás Hála

During the existence of Czechoslovakia, Czech and Slovak typesetting rules were defined by one common norm. At present, Slovak rules have mostly been fixed by official documents whereas Czech rules are rather custom based.

This contribution deals with the comparison of the rules in both languages, especially with the use of hyphen, dashes, lists etc. In addition to that, some of these items in Czech and Slovak differ considerably from those in other languages. All important items have been compared with facilities in the typesetting system ConTEXt and it seems that some situations have not been covered in configuration files. Therefore several suggestions for language settings have been made in order to make ConTEXt more general and comfortable for ordinary users.

Participant list

Leo Arnold, Technische Universität München,
Garching bei München, Germany
latex@arney.de

Doris Behrendt, Gymnasium Marktbreit,
Biebelried, Germany
doris.behrendt@me.com

Sietse Brouwer,
The Netherlands
sbbrouwer@gmail.com

Gyöngyi Bujdosó, Faculty of Computer Science, University of Debrecen,
Debrecen, Hungary
bujdoso.gyongyi@inf.unideb.hu

Andreas Dafferner, Heidelberger Akademie der Wissenschaften,
Heidelberg, Germany
andreas.dafferner@adw.uni-heidelberg.de

Karin Dornacher, DANTE e.V,
Heidelberg, Germany
office@dante.de

Willi Egger, BOEDE,
Sambeek, The Netherlands
w.egger@boede.nl

Kai Eigner, Tat Zetwerk,
Utrecht, The Netherlands
eigner@tatzetwerk.nl

Ivo Geradts, Tat Zetwerk,
Utrecht, The Netherlands
geradts@tatzetwerk.nl

Frans Goddijn,
Amsterdam, The Netherlands
frans@goddijn.com

Patrick Gundlach, Dante e.V,
Berlin, Germany
patrick@gundla.ch

Hans Hagen, Pragma ADE,
Hasselt, The Netherlands
pragma@wxs.nl

Tomás Hála, Mendel University Brno,
Brno, Czech Republic
thala@mendelu.cz

Taco Hoekwater, Bittext,
Breskens, The Netherlands
taco@bittext.nl

Karel Horak,
Lety, Czech Republic
horakk@math.cas.cz

Jean-Michel Hufflen, University of Franche-Comté,
Besançon Cedex, France
jmhuffle@femto-st.fr

Bogusław Jackowski, GUST,
Gdañsk, Poland
jacko@bop.com.pl

Hironori KITAGAWA,
Tokyo, Japan
h_kitagawa2001@yahoo.co.jp

Harald König,
Balingen, Germany
koenig@linux.de

Reinhard Kotucha, Capical GmbH,
Hannover, Germany
reinhard.kotucha@web.de

Siep Kroonenberg, RUG,
Groningen, The Netherlands
siepo@cybercomm.nl

Silke Krumrey, Fachhochschule Stralsund,
Stralsund, Germany
silke.krumrey@fh-stralsund.de

Jan Kula,
Prague, Czech Republic
jano.kula@gmail.com

Yusuke KUROKI,
Yokohama, Japan
kuroky@users.sourceforge.jp

Johannes Küster, typoma GmbH,
Holzkirchen , Germany
info@typoma.com

Dag Langmyhr, University of Oslo,
Oslo, Norway
dag@ifi.uio.no

Lucien Lemmens,
Laakdal, Belgium
lcnlmmns@me.com

Manfred Lotz,
Frankfurt, Germany
manfred@dante.de

Jerzy Ludwichowski, GUST,
Toruñ, Poland
jerzy.ludwichowski@umk.pl

Bernd Militzer,
Kempen, Germany
bernd@militzer.net

Christina Möller, Fachhochschule Stralsund,
Stralsund, Germany
christina.moeller@fh-stralsund.de

Thomas Ratajczak, German Army,
Langenfeld, Germany
ratajczak@gmail.com

Heiner Richter, Fachhochschule Stralsund,
Stralsund, Germany
heiner.richter@fh-stralsund.de

Edgaras Šakuras, VTEX UAB,
Vilnius, Lithuania
edgaras.sakuras@vtex.lt

Luigi Scarso,
Padova, Italy
luigi.scarso@gmail.com

Volker Schaa, Dante e.V,
Darmstadt, Germany
v.r.w.schaa@gsi.de

Martin Schröder,
Duisburg, Germany
martin@oneiros.de

Robbert Schwippert, Docwolves B.V.,
Dordrecht, The Netherlands
ras@elvenkind.com

Martin Sievers, Dante e.V.,
Trier, Germany
martin@dante.de

Linas Stonys, VTEX UAB,
Vilnius, Lithuania
lstonys@vtex.lt

Piotr Strzelczyk, GUST,
Gdynia, Poland
piotr@eps.gda.pl

Sigitas Tolušis, VTEX UAB,
Vilnius, Lithuania
sigitas@vtex.lt

Kees Van der Laan,
Garnwerd, The Netherlands
kisa1@xs4all.nl

Ulrik Vieth,
Stuttgart, Germany
ulrik.vieth@arcor.de

Mari Voipio, Lucet.fi,
Vantaa, Finland
mari.voipio@lucet.fi

Herbert Voß, Freie Universität Berlin,
Berlin, Germany
herbert@dante.de

Munehiro YAMAMOTO,
Japan
munepixyz@gmail.com

Uwe Ziegenhagen, DB Private Equity,
Cologne, Germany
ziegenhagen@gmail.com