TEX Education – a neglected approach

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

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

Contents

What TEX&Co education is available?

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the law of diminishing returns applies: our TEX tutorials are provided on the actual TEX Collection DVD and the outdated 4AllTEX CDs, and stored in the TEX archives of old, founded in the pre-WWW time, read before HTML appeared, if not in books.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

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

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

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

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

Conclusion: no TEX&Co classes are offered in general.

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

Education in NTG

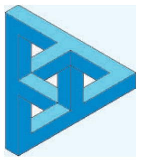
In the beginning of NTG we had working groups. The education Working Group organized a LATEX course with courseware `Publiceren met LATEX`, in Dutch. I was the SGML teacher at the Stanford `10 years of TEX and MF` TUG meeting. At Stanford I attended Dough Henderson's MF class. At the conference I met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

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

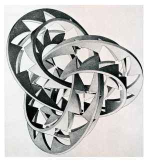
Amy, Bogus, and David were paid for their teaching and enjoyed hospitality at my place at my costs. During the Boston TUG meeting I visited AMS and invited Michael Downes and Ralph Youngen to tell us in Holland about how AMS supports their authors. I studied the clever AMS formats and styles, and criticized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

The 4AllTEX CD effort had a strong educational flavor.

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



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



M.C. Escher
← Knot
CGL's
→ Sort of

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

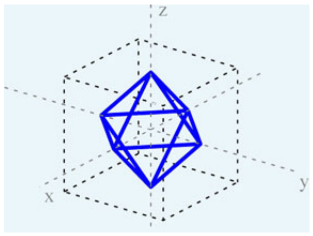
Is there a need for TEX&Co courses?

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

Apparently the public does not ask for courses.

Education material on TEX Live DVD

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



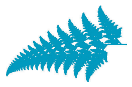
The public domain TEX program, distributed on the TEX Collection DVD, comes along with a wealth of documentation, software and books, which allow for self-study of the open and public domain programs AnyTEX&Co, while for advanced questions one may consult FAQs, or discussion lists with their archived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

But...
we might try.

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

Education issues

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



Language

A fundamental aspect in general education is language, it is the basis. Everybody speaks a language, can read and write, and may publish easily nowadays via the WWW. Language serves a lifetime! Language fluency is a prerequisite for participation in a culture, is fundamental in communication. Language lies at the heart of publications.
The TEXbook spends several chapters on just typesetting text, deals with the variety of

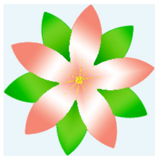
☐ type faces
☐ accented characters
☐ ligatures
☐ hyphenation
☐ line and page breaking,
☐ structuring commands . . .

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

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

Typesetting novels by TEX is a trifle. However, book production is culture biased, with variations in layout

and the used fonts. Is TEX the only tool available for this task?
Definitely not. Novels are produced by word processors, with excellent spelling checkers, I presume. I have heard of Adobe's Indesign, no hands-on experience as yet, however. MS Word I use quite often for contributions to our local gardening bulletin. These gardeners have not even heard of AnyTEX.



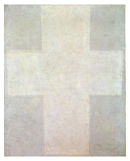
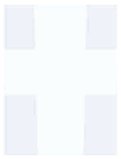
It amazes me that we have no BabelTEX as yet, where the commands and error messages can be instantiated for your language. For example in Russian you would then end with \пока instead of \bye, and just keyboard Cyrillics.

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

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

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

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


Malevich
Suprematism:
White cross on a
White background
Emulation ⟶


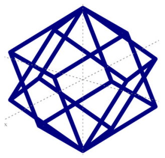
Maybe this fact should be brought to the attention of a casual user, and should not be burdened by all the other complex tasks TEX can do, which might frighten an innocent user.

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

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

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

I contacted budding LUGs in Germany, France, England and Scandinavia, and looked for cooperation. We cooperated in organizing EuroTEXs. In my time as president, NTG funded the LATEX2ε project. Much later NTG took part in funding the Latin and Gyre font projects.



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

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

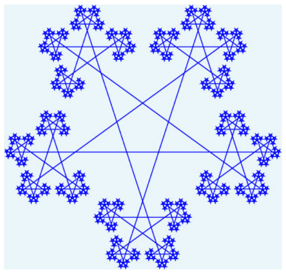
Other jargons can be typeset equally well, but you have to write the equivalent of the intelligent math mode yourself. I have not heard of such an effort.[6]

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

But...
That is not enough, at least with the mathematical training I enjoyed. No typesetting conventions of math was ever taught to me. No `Mathematical writing, 1989` by Knuth as main author (who else?) existed. Happily, TEX embodies it all: the subtle spacing around Math symbols, the awareness of operators and how to typeset them in context, the composition of symbols with limits for example, is all taken care of by TEX, though TEX is not perfect. In-line Math and displayed Math may look different. The choices made by TEX, implemented in the Math mode, are wired in, but... parameterized. Math constructs can be marked up by using macros. The Math fonts are not just ordinary fonts, they come along with a suite of `\fontdim` parameters for placement of the glyphs in context by the intelligent Math mode. Using

just another, commercial, Math font with the richness of the TEX math mode knowledge is not trivial. OpenType Math fonts come also with a suite of parameters, some the same as in TEX some different, and some beyond TEX's. Work on incorporating OpenType Math fonts for use in TEX, is underway by Bogusłav Jackovski et al., and about to be released.

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

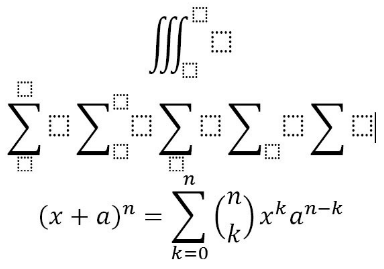


Awareness

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

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

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



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

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

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

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

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

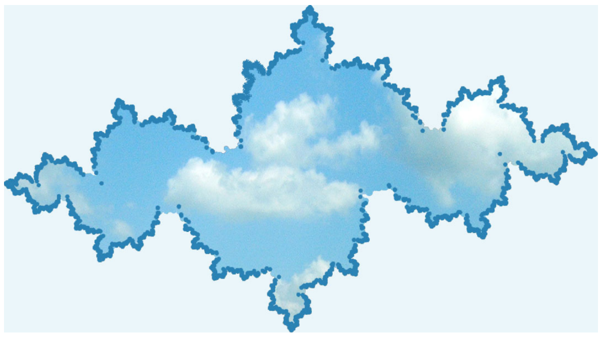
Do we have IDEs with really advanced editors with AnyTEX support, which not only prompt markup tags, but also prompt formula templates to be filled in?

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

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

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

> Ignoring the past is not a scientific tradition.



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

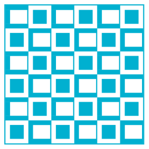
```
\def\dataMT{1\cs 2\cs 3\rs
            2\cs 4\cs 6}
```

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

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

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

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



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

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

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

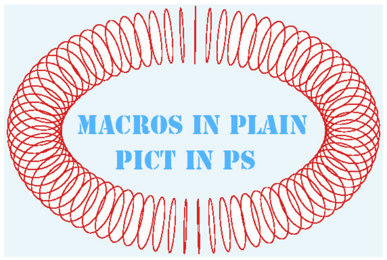
Awareness of other tools Phil Taylor in his recent `\parshape pre-processor`[9] starts with telling that he used HTML, because 'HTML can flow text around an embedded (rectangular) image in a totally straightforward manner, requiring nothing more than...'. He continues that he would like to use TEX and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

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

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

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

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



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

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

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

P. Mondriaan
Lozenge
Composition
with two lines

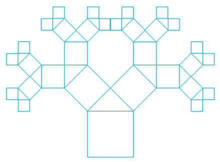
The bundling of the various macros, pictures, references, tools gave rise to my BLUe collection, nicknamed after Ben Lee User in the TEXbook.

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

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

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

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



Insight

Characteristics of insight are

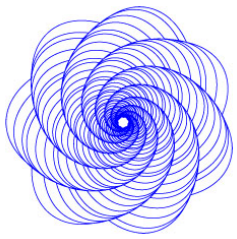
- ☐ Abstraction
- ☐ Separations of Concerns, SoC
- ☐ Parameterization
- ☐ To foresee the future
- ☐ To use TEX&Co
- ☐ To adhere Minimal Markup, Suprematism
- ☐ To use Paradigms
- ☐ To reuse parts

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

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

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

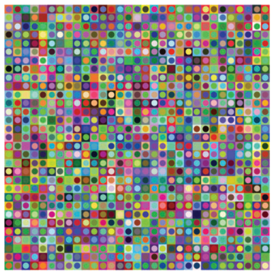
MetaFont is the big example of parameterization where each glyph is characterized by dozens of parameters. To handle gracefully related parameters Knuth invented the `suffix` concept, as far as I understand it is a unification of the common index and the PASCAL record, in the minimal style. In creating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



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

Using paradigms in macro writing will increase readability.

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



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

> TEX is a fixed point, only the environment changes

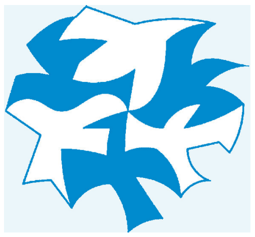
Adaptation to PS is for example taken care of by the driver `dvi(2)ps` and ilks, and pdf output can be obtained by `Distiller` or `Acrobat` (not tested though by me for a document), or just one of the various `pstopdfs`. TEX commands for handling colors and inclusion of graphics are dictated by the drivers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

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

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

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

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



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

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

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

```
\input adhocmacros

\author ...

\abstract ...

\keywords ...

\head Script

...
\subhead
```

```
...
```

```
\jpgD ...
```

```
...\ftn ...
```

```
\bye
```

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

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

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

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

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

Of course

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

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

But...

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

But...

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

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

Knuth's approach

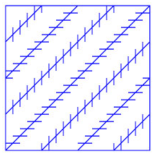
What astonishes me most is that Knuth's plain.tex is not embraced by the majority. His basic approach should be taught, because in TEX, well in automated digital typesetting, there are so many subtle things, where you will stumble upon sooner or later, which cannot be shielded away from you by AnyTEX,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

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

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

But...

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



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

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

John Plaice commented on my presentation

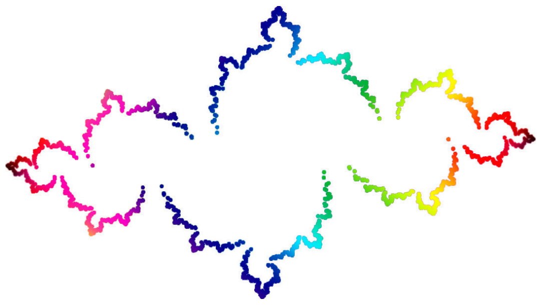
'... that it was not possible to praise Dijkstra and Knuth in the same sentence as the two held completely opposite points of view with respect to programming languages. When Dijkstra published his 'Go to considered harmful'(CACM 11(3):147-148, 1968), Knuth defended the goto statement with his 'Structured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

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

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

have fought much more strongly with Knuth for him to create a more usable language.'
Fair enough! Especially easier languages.
But...
I don't see what is against Knuth's attitude. Just a different approach.
Remember...
There Is More Than One Way To Do It.
I appreciate the features of high-level structured programming, with no, well... little, but well-defined side-effects, like for example exception handlers.
But...
when I learned Algol68 at the time, I was much surprised, because it seemed to me one big side effect. For me Knuth[12] knows what he is talking about, and he like nobody else, produced marvelous errorless tools, so SuPerBe documented

$\mathrm{S_{uper}uP_{leasanttoread}erB_{eyondthoroughness}e}$



TEX&MetaFont

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

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

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

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

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

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

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

or directly, the popular one-step

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

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

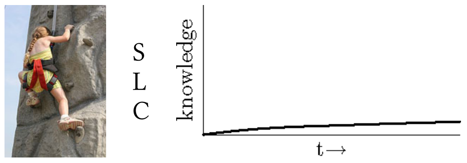
I don't believe that one person, or even a group, can write a monolithic successor of TEX, of the same quality as TEX and the time-proven cooperating little languages like dvi(2)ps, MetaPost, Acrobat and ilks.

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

Drawbacks of TEX

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

- ☐ TEX SLC 120+% Energy[14]

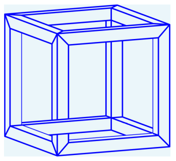


- ☐ After so many years AnyTEX&Co lack some sort of stability. Me, for example, I don't have MetaPost running in an IDE. The TEXnicCenter is not perfect, does not provide for menus suited for plain TEX, too smart editor...
- ☐ - No GUI[15]
  But...
- ☐ TEX&Co 99+%Quality
- ☐ TEX&Co gives you full control

Drawbacks of MetaFont

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

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



Literate programming

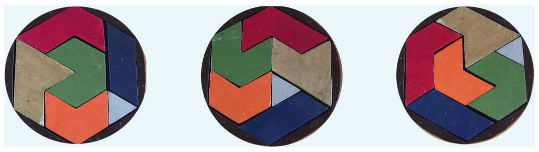
I like to characterize literate programming by

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

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

But...

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



Even at Stanford I could not find offerings for TEX classes nor classes for literate programming.

No need? Still ahead of time?

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

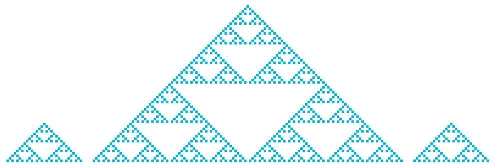
TEX Collection DVD

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

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

The Center does not provide buttons for processing plain TEX with a format file of your own, at least that is not clear to me. I had to fool the system: opened a template, threw all that I did not need away and processed my minimal plain TEX job as LATEX!

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



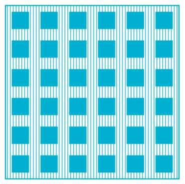
This paper asks among others to include in TEXnicCenter also buttons for processing by Knuth's `plain.tex`. It would not harm to include as example the minimal TEX job

```
Hello world!
\bye
```

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

TEX→ `dvi`, `pdf` (or `ps`).

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



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

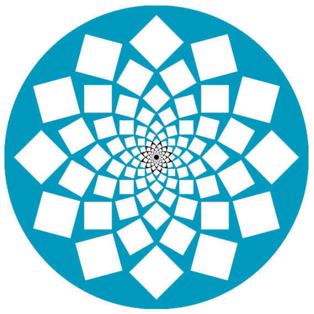
TEXing Paradigms Master Class

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

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

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

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



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

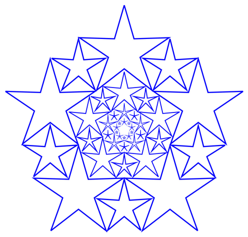
This Master Class I would organize via the internet, although a TEX course via the internet is not new. Times have changed. The teacher, or conductor, does not have to be the most knowledgeable TEXie, just the coordinator, like a chairman in a meeting. Attendees and coordinator commit themselves for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

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

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

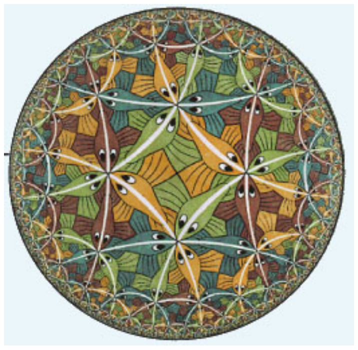
1. Just as in plain TEX
2. Provide for running heads in the headline and provide for a ToC creation, like in `manmac`
3. Provide for hypertext cross-referencing links and PDF bookmarks.

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



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

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



M.C. Escher
Limit Circle III

TEXing Paradigms beneficial?

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

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

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

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

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

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

ConTEXt and LATEX provide for automatically numbered lists.

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

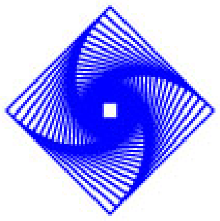
If you have to mark up a document with long lists of numbered paragraphs a minimal `\nitem` macro, similar in use as `\item`, can be handy. I would propose the following `\nitem` nowadays, where I took care of my perceived impossibility at the time by redefining `\par` at an appropriate local place.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

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

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

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

needed.



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

Examples of macro writing

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

Tough exercise

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

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

My current solution reads

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

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

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

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

Wind macros

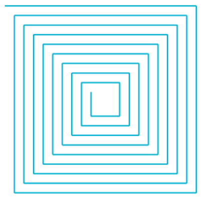
Despite the powerful MetaPost, I will talk for the first time about my plain TEX turtle line drawing graphics macros of ≈13 years ago, which I have used for a variant solution of the TEXbook exercise 22.14 (see later), but also for drawing some fractals, e.g. my favorite the Binary tree and the H-fractal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

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

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

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

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



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

$\backslash x = -\backslash k * \backslash\mathrm{unitlength}$

$\backslash y = \backslash k * \backslash\mathrm{unitlength}$

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

$.5\backslash k * \backslash\mathrm{unitlength}$.

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

Contest

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

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

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

by either the minimal markup

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

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

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

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

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

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

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

But...
I must confess, it took me some time to understand it. A Master Class would have been beneficial for me, to learn to understand these kinds of macros more quickly ☻.
The near winner was Bernd Raichle with a thorough, straight-forward solution, not avoiding the pitfall, and which is a bit too long to include in this note.

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

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

2D Graphics

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

- ☐ straight line drawings by the wind macros and by gkppic in plain TEX, and PS, such as fractals and flowcharts
- ☐ graphics with curved lines by PS and MP, such as graphics composed of oblique lines (and spurious envelopes), circles, and general curves, to be drawn by splines.

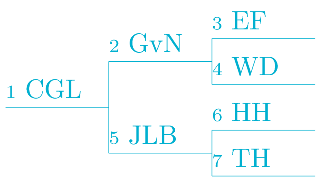
Binary tree

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

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

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

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



The previous drawing is obtained via

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

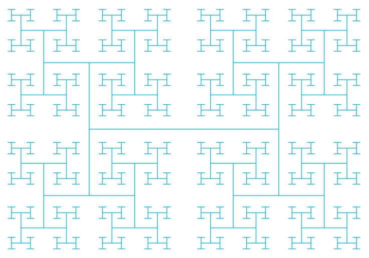
Remarks. The (educational) indices at the nodes are inserted to identify the nodes, to make the (order of) traversal explicit. The replacement text of \1 will be placed at node 1, etcetera. Adding the leaves, the initials, is done by adapting the \E macro and invoking \xytxt, which places text at the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

The variant PS code of the binary tree macro reads

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

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

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

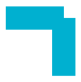


My PS variant is even more concise and reads

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

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

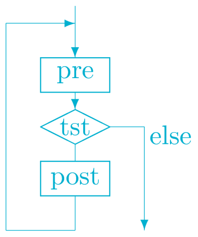
But...
when linethickness becomes noticeable, they suffer from a severe disadvantage of ugly line joinings like the notch



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

Flowchart

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



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

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

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

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

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

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

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

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

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

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

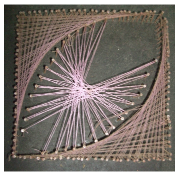
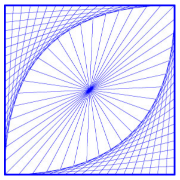
```
drawarrow boxb.e--z1--z2;
z3=boxb.w-(20,0);
z4=boxc.w-(20,0);
drawarrow boxc.w--z4--z3--boxb.w;
endfig
end
```

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

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

Oblique lines

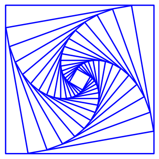
When my youngest daughter was at school, she produced the following object created by 'oblique lines' with spurious envelopes (left). My emulated result in MP is at the right, and created just for this note.

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

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

If we rotate and shrink appropriately we get the interesting figure



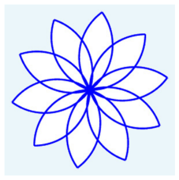
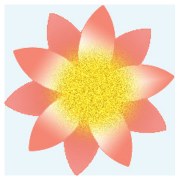
In Goossens' et al. The LATEX Graphics Companion this figure is shown as an example of a recursive object.[20]

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

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

PostProcessing by Photoshop

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

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

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

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

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

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

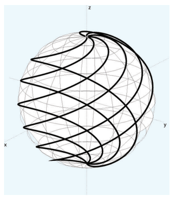
PostScript spaces

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

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

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

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



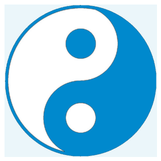
Yin Yang

A neat timeless MetaPost code I borrowed from Hobby

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

without the 'eyes'.[22]

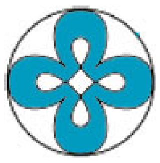
Below my enriched PostScript variant of Hobby's MetaPost code, for the real Yin Yang picture.



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

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

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



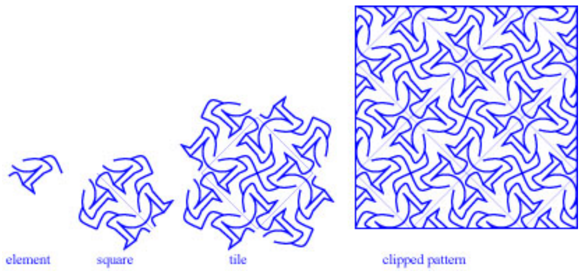
Syntactic sugar? Yes, but the PS code can directly be used in documents, to be inserted by the `dvi(2)ps` driver, alas not directly by pdf(Any)TEX. Strange.

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

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

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

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



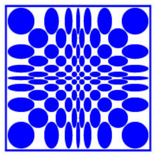
Smiley

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

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

Schröfers opart

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



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

EuroTEX94 battleship logo

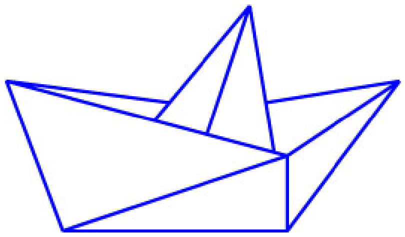
In order to illustrate the calculation of intersection points in PS I have borrowed the EuroTEX94 battleship logo.

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

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

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

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



The data in PS code reads

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

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

```
/p1{0 0}def /p2{10 0}def ...
%
/p {p1 p2 p3 p4 intersect}def
%
/intersect {%p1 p2 p3 p4 -> x y
makecoef 7 3 roll
makecoef
solveit}def %end intersect
```

```
%
/makecoef{%z1 z2 -> e a b
4 copy       %x1 y1 x2 y2 x1 y1 x2 y2
4 -1 roll mul
3 1 roll mul sub
5 1 roll 3 -1 roll sub
            %(y2x1-y1x2) x1 x2 y2-y1
3 1 roll sub%(y2x1-y1x2) y2-y1 x1-x2
}def %end makecoef
```

As last piece the definition of solveit

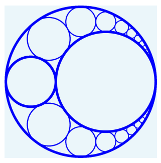
```
/solveit{%e a b f c d  -> x y
%Equations: ax + by = e   p=pivot
%           cx + dy = f
%pivot handling  %e a b f c d
1 index abs      %e a b f c d |c|
5 index abs      %e a b f c d |c| |a|
gt {6 3 roll} if %exchange 'equations'
%stack: e a b f c d or f c d e a b,
%first is in comments below
exch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

Finally, the drawing of the battleship

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

Circle covered by circles

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



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

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

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

2.5D Graphics

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

- ☐ 2D PostScript Device Space
- ☐ 2D PostScript User Space

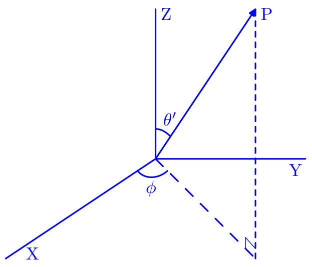
I added

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

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

The projection formula reads

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



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

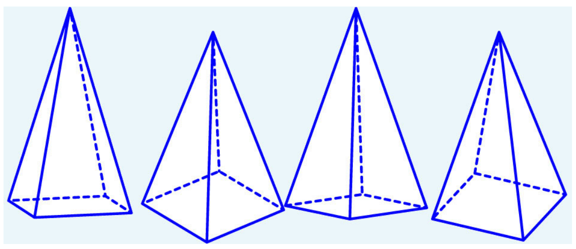
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sin\theta & \cos\theta \\ 0 & \cos\theta & -\sin\theta \end{pmatrix} \begin{pmatrix} -\cos\phi & \sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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

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

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

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



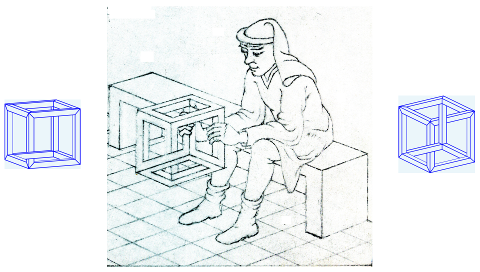
The PS code reads

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

Escher's impossible cube

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

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

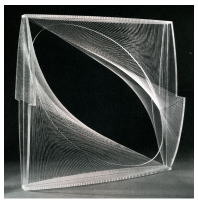


I'm happy that after so many years, I had the guts to emulate the cubes.

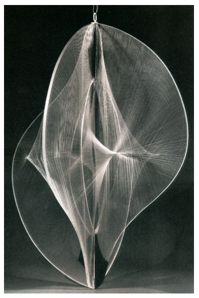
I consider each corner of the (impossible, non-mathematical) cube as a cube itself, with its 8 corners as data points, which yields in total 64 data points. After projection I could connect the appropriate points and make the (impossible) cube. First, I did the erasing and adjusting in Photoshop, but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

Gabo's constructive art

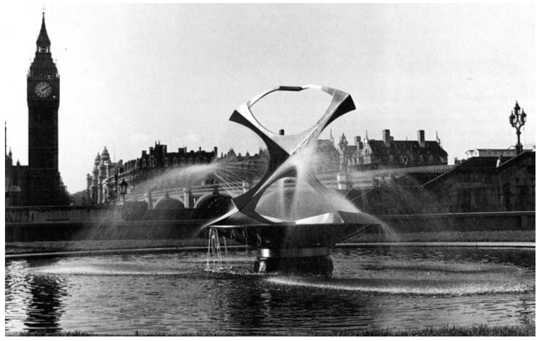
Long ago, I was still a student, I visited the Tate Gallery in London, and was captivated by the constructive art of Gabo,[27] especially his `linearii` of the early 1940-ies.



Naum Gabo
← Lineari
→ Linearii



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



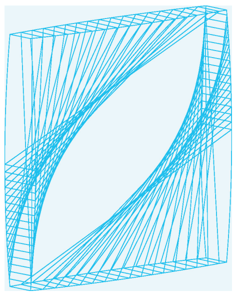
In the fountain, the regular surface is formed by jets of water, and changes dynamically, because it rotates, due to the 'action is reaction' principle.

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

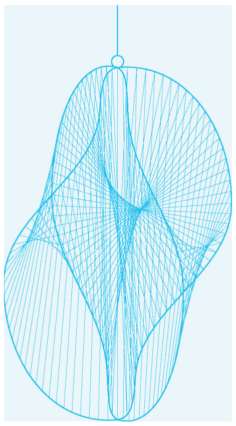
With my youngest daughter I imitated Gabo's `linearii` in plastic.

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

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



Naum Gabo
Lineari


Naum Gabo
Linearii

Of linearii I have included the MP code below

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

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

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

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

Reuse

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

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

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

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

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

Sometimes reuse implies some extra work.

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

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

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

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

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

If only `\pdfTEX` could handle PostScript...

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

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

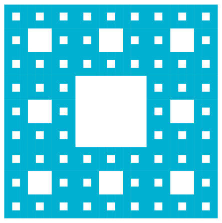
but, alas TEX has no garbage collector, it would not save on memory, it only reduces the possibility of conflicts.

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

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

But...

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



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

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

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

3D metaPost?

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

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

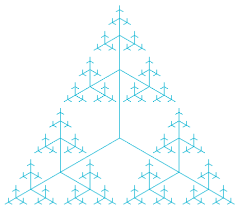
Conclusions

Whatever your tool in computer-assisted typesetting, a Master Class on macro writing in plain TEX and MetaPost is worthwhile, along with discussing tools for similar and complementary tasks to increase awareness, insight and productivity.
A course in literate programming would be great too.

Wishes

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

May the TEX Collection maintainers, the TEXnicCenter authors, and Jonathan Kew in his TEXworks, support the plain TEX and MetaPost users as well as the LATEX and ConTEXt users.



Hopes...

TEX&Co world will

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

I learned a lot at and after the conference.

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

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

Interesting for me with respect to Cyrillics.

Acknowledgements

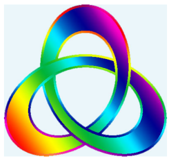
Thank you NTG for inviting me for this conference. Because of this invitation I contributed this paper, refreshed my TEX knowledge, polished my pictures undusted `blue.tex`, and tried to get a modern communicating TEX⇔ MetaPost IDE running on my PC.[33]

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

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

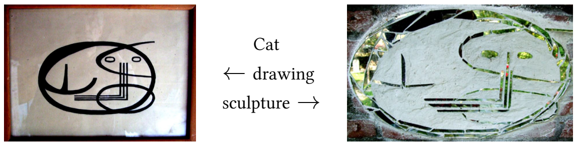
Thank you Taco Hoekwater for procrusting this note into the proceedings, and for your advice in getting me TEXing again.

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

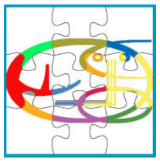


I needed the equation solver of MetaFont, and thanks to my familiarity with splines I could reuse the MetaFont splines in PostScript. Recently, I made in MP a variant, see Appendix II.

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



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



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

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

Notes

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

19. My MetaPost1.005 did not yet provide it.
20. This rotating shrinking squares and a few other pictures, which I borrowed from H.A. Lauwerier's 'Meetkunde met de microcomputer', such as the Koch fractal, the Hilbert curve, the Jura fractal, Escher's knot,. . . and published in MAPS in the mid-90-ies, in my 'Just a little bit of PostScript', 'Graphics & TEX—a reappraisal of Metafont', or 'Tiling in PostScript and Metafont—Escher's wink', I found back, without reference and translated in MetaPost.
21. As yet not! Be aware that the PostScript code is just handed through, not taken notice of.
22. The numeric equation, `u=1cm`, looks a bit strange, looks more like a dimension à la TEX. It becomes clear when you realize that `cm` is just a scaling factor.
23. The problem is that generally I got a picture per page, and I did not know how to trim the picture to its bounding box. After the conference I found out how to trim these pictures in Acrobat 7 professional: select the picture, copy it to the clipboard, and then click create PDF and select From Clipboard Image.
24. The conversion was generally done by opening the `.pdf` pictures in `Photoshop`, trim them and save them as `.jpg`. Later I became aware of the `prologues:=3;` statement, which yields a.o a picture trimmed to the Bounding Box.
25. M.C. Escher, 1898–1972, Dutch artist.
26. I must have the negative somewhere, but can't find it, alas. I'll give it another try.
27. Naum Gabo, 1890–1977. Born Naum Borisovich Pevsner. Bryansk. Russian Constructivist.
28. I colored the picture by post processing in Photoshop. A work flow approach is simpler via the use of the earlier defined `\blue1` and switching back via `\black`.
29. The term hypertext was coined by TeD Nelson during the 1960s, the concept can be traced to Vanneger Bush in 1945. See the Hypertext issue of the CACM july, 1988.
30. I read in the PDF manual the excuse that PDF is a successor of PS, but does not have a programming language built in???
31. This wish will be fulfilled, as reported earlier.
32. Courtesy Bogusłav Jackovski et al. and Ulrik Vieth.
33. Still a wish. But... Wybo installed Ubuntu Linux for me on my laptop, with TEXworks, so I can explore that. Will give me enough work to do.
34. After the conference the NTG discussion list told me how to run the MetaPost utility supplied on TEX Live. Open a new directory, copy `cmd.exe` into it as well as your `filenamme.mp`. Double click the `cmd.exe` icon and type after the prompt `mpost filename.mp`. Another suggestion was to use MetaPost from within the `SciTE` editor. It would have been handy if the `readme`, which comes with the TEX Live, would have contained information on how to use the various utilities. Don't assume a casual user knows it.

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

Appendix I: Contest Solutions

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

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

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

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

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

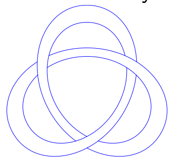
I came up with the solution below, which I will add to my FIFO paradigms list in my TEXing Paradigms collection. Of course, I used this one for my purpose.

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

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

Appendix II: Escher's knot

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



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

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