MAPS

NUMMER 46 • NAJAAR 2015

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

Voorzitter
Hans Hagen
ntg-president@ntg.nl

Secretaris
Sietse Brouwer
ntg-secretary@ntg.nl

Penningmeester
Ferdy Hanssen
ntg-treasurer@ntg.nl

Bestuursleden
Frans Absil
fgj.absil@nlda.nl

Frans Goddijn
frans@goddijn.com

Postadres
Nederlandstalige TEX Gebruikersgroep
Korte Langestraat 2
2312 SK Leiden

ING bankrekening
1306238
t.n.v. NTG, Arnhem
BIC-code: INGBNL2A
IBAN-code: NL53INGB0001306238

E-mail bestuur
ntg@ntg.nl

E-mail MAPS redactie
maps@ntg.nl

WWW
www.ntg.nl

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

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

Lid worden kan door overmaking van de verschuldigde contributie naar de NTG-giro (zie links); vermeld IBAN- zowel als SWIFT/BIC-code en selecteer shared cost. Daarnaast dient via `www.ntg.nl` een informatieformulier te worden ingevuld. Zonodig kan ook een papieren formulier bij het secretariaat worden opgevraagd.
De contributie bedraagt € 40; voor studenten geldt een tarief van € 20. Dit geeft alle lidmaatschapsvoordelen maar *geen stemrecht.* Een bewijs van inschrijving is vereist. Een gecombineerd NTG/TUG-lidmaatschap levert een korting van 10% op beide contributies op. De prijs in euro's wordt bepaald door de dollarkoers aan het begin van het jaar. De ongekorte TUG-contributie is momenteel $85.

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

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

Productie. De Maps wordt gezet met behulp van een LATEX class file en een ConTEXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.40.15 en luatex 0.80.0 draaiend onder MacOS X 10.10. De gebruikte fonts zijn Linux Libertine, het niet-proportionele font Inconsolata, schreefloze fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

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

Contents

Redactioneel

Shortly before publication, we heard the sad news that Kees van de Laan had died. Kees had an effect on everyone he met. My children were very young when I brought them to BachoTEX, but they still remember Kees as, 'the nice man with the big dog.' This issue contains two articles by Kees, but begins with a testimonial written by Eric Frambach, Jerzy Ludwichowski and Philip Taylor.

An friend of mine recently ask me to help him typeset a series of books. My friend is a successful magazine publisher, but he is unschooled in the mechanics of typesetting. However, the prospect of typesetting beautiful books has induced him to learn TEX. As he adopts the edit-typeset-view work cycle the choice of tools, especially an editor, is fairly important. This edition of the MAPS could have been written especially for him, since it includes a review of several TEX specific and TEX aware editors. Here is a short summary.

Frans Absil describes TEXShop, a basic, reliable and open-source TEX editor running on Mac OS. TEXShop displays three windows: an editor, a console and a integrated PDF viewer corresponding to the edit-typeset-view cycle we all know and love.

Willi Egger describes TextMate, a commercial text editor running on Mac OS. Support for TEX and LaTEX include, e.g., syntax highlighting and keyboard shortcuts. ConTEXt support is provided by an add-on bundle written by Patrick Gundlach. I particularly like TextMate's ability to fold paragraphs; which helped me narrow my focus by hiding surrounding distractions.

Sytse Knypstra describes TEXworks, an open-source and cross-platform TEX specific editor. A particularly interesting feature is its integrated PDF viewer supports source/view synchronization; which allows one to switch between corresponding positions in the source and resulting PDF.

Siep Kroonenberg describes TEXstudio – the primary TEX editor in the University of Groningen's TEXLive installation, and particularly well suited for new and aspiring TEX users. Siep shows us what TEXstudio can do by leading us through progressively harder examples from our first session, through typesetting bibliographies to configuring custom templates.

And for those interested in beautiful mathematics made visible, there are two articles by Kees van de Laan. The first is about his PSlib.eps PostScript library –- a collection of PostScript snippets to facilitate drawing in PostScript. This article first appeared in the proceedings from BachoTEX 2014.

Kees's second article is about drawing spirals in PostScript; a task made easier, and the resulting PostScript code more elegant, by using polar coordinates and PostScript's user space rotation.

This issue contains two articles by Hans Hagen. For the connoisseur of fine editors, Hans Hagen has written a description of his favourite editor -– SciTE. Build on top of the scintilla editor framework, SciTE is an all-purpose editor; which Hans has customised to accommodate ConTEXt. With the custom lexers that ship with ConTEXt SciTE can understand TEX primitives, ConTEXt low level and user interface commands, Lua and MetaPost. For those, like myself, who prefer to use a general purpose editor with TEX specific customisations, SciTE has few competitors.

In the second article, Hans demonstrates the flexibility of ConTEXt MKiV by embedding Lua code inside MetaPost. This is a particularly handy approach when generating graphics from external data. Hans begins with a basic example, and then shows how predefined and custom 'helper' functions can help us organise and simplify our code.

I hope you will enjoy this issue of the MAPS. We are grateful to the authors for their contributions.

PS: The next MAPS will be a jubileum issue marking the twenty-fifth anniversary of the NTG. If you have any reminiscences or TEX experiences you would like to share, please consider writing an article.

Michael Guravage

Memories of Kees

Kees (really Cornelis Gerardus) van der Laan, 22-12-1943 – 24-08-2015.
TEX guru, Apple Macintosh and PostScript *aficionado*, but most of all a warm and generous man who made friends wherever he went, and who will be remembered with deep affection by all who had the pleasure and privilege of knowing him.

In 1988, Kees was one of the founders of the NTG (*Nederlandstalige TEX Gebruikersgroep*). He gathered together many other TEX enthusiasts and worked hard on building a network of people and organisations using and experimenting with TEX. Many activities were organised, such as meetings, courses and workshops. Kees persuaded several people to join him on his trips to BachoTEX and even to Russia, where lots of TEX activity was developing. Bringing together kindred spirits in order to generate even better ideas and develop them into new projects and new products was always on Kees's mind. He was a true TEX evangelist, and was made an Honorary Life Member of NTG in recognition of the quality and quantity of his contributions to the TEX world at large (his early work included TEX code for typesetting crosswords, Bridge and the Towers of Hanoi; the implementation of stacks and queues in TEX; TEX algorithms for sorting and searching; and many many other ideas as well).

In the Netherlands, he (and the NTG) actively supported a project to develop the very first plug & play CD-ROM with TEX software. This turned out to be a giant leap forward by providing a consistent TEX set-up for use both by beginners and experts alike that worked right out of the box.

In Poland, Kees was guest of honour at one of the very first GUST ('*(Polska) Grupa Użytkowników Systemu TEX*') conferences, and was formally made an Honorary Life Member of GUST at the AGM which took place during that meeting. Until then, national TEX user group meetings had tended to be just that – national – but BachoTEX set out to be a truly *inter*national user group meeting, and succeeded beyond its wildest dreams.

At that time, Kees was working on some extensions to Manmac, and his talk at the 1994 BachoTEX meeting was entitled 'Manmac BLUes'. By 1995, Kees's work on extending Manmac had grown into a complete format, 'BLUe's format', as Kees called it. Based on Manmac (amongst other sources), but primarily consisting of Kees's own unique work (Kees described it as 'build[ing] upon Manmac, upon functionalities provided in the TUGboat styles, and upon experience gained by the AMS in TEX formatting'), BLUe's format (or 'BLUeTEX', as it later came to be known, probably by analogy with Blue-tack !) was publicly released at the 1995 BachoTEX meeting, and Kees gave the first of many, many talks to GUST on that subject during the conference. BLUe's format, which takes its name from Knuth's apocryphal 'B. L. User' (in Britain he would be called 'the man on the top deck of a Clapham Common omnibus', i.e., the average person, as opposed to a TEX wizard), set out to make the full power of TEX accessible to the average user by encapsulating, in a single format, all of those elements that are essential for an advanced use of TEX but which would almost certainly be too difficult for the average TEX user to program for him/herself. Of particular interest to Kees was his wish that the user should adopt 'minimal markup', and he would return to this theme time and time again.

As well as pure computer science and programming, Kees had a keen interest in computer graphics (a theme that was to become ever more important to him as time passed), and his 1996 papers were on 'TEX and Graphics – a reappraisal of METAFONT/MetaPost' and on 'Turtle graphics in TEX (a child can do it)!'. During the years that followed Kees presented papers on 'TEX inside, or insights in TEX ?', 'A little bit of PostScript', 'Tiling in PostScript and METAFONT', 'Syntactic Sugar', and many many other topics. In recent years, he combined his all-time passion for mathematics and computer graphics by creating beautiful illustrations of mathematical

theorems; publishing his results in various journals and BachoTEX conference materials. He demonstrated how PostScript can be used to elegantly compute and display fractals of any kind. In fact, Kees showed that for any 2D, 2.5D and 3D problem that can be modelled as a mathematical equation, PostScript is a great tool to explore and solve that problem. But Kees was also an art lover. Whenever he noticed symmetries or mathematical inspiration in a piece of art, he would try to model and emulate it in PostScript. In particular, the works of Gabo, Mondriaan and Escher (all minimalists!) inspired him. And of course, this work was presented not only at BachoTEX and NTG meetings – Kees was a regular attender at EuroTEX meetings, TUG meetings, and anywhere else where he could exchange ideas on mathematics, computer science, computer graphics and computer programming with other like-minded individuals.

Kees was always very generous in sharing whatever he had developed. He always published his work for free, and wrote more than a hundred articles in MAPS (NTG's journal), explaining how it all works and how it could be adapted to anyone's own needs and preferences. He was a big advocate of keeping things simple ('minimal markup' was one of his hallmarks). The TEX or Postscript code he wrote was typically very neat, concise and elegant, much in line with the programming style that Knuth (whom Kees admired greatly) used in his Manmac macros. Sometimes his code and articles were hard to follow, but they always rewarded the careful reader. There was a lot to be learned from his fresh and clever approach to solving problems.

Kees's interests were not restricted to the technical domain – he was a great conversationalist, as well as great company, and the authors of this short tribute are just a few amongst many who have spent countless happy hours with Kees in and around his home in Garnwerd (in the early days, always accompanied by Beer-the-dog), and it was Kees who introduced many of us to the Dutch custom of eating fresh herring dipped in onion and washed down with some good Dutch lager. Also to *krupuk* (think 'giant prawn crackers'), and to many other Dutch delicacies as well.

Kees loved people, and made friends wherever he went. He was particularly interested in Russia, and made many very good Russian friends, some of whom would visit him in Garnwerd and stay at his home, and with one of whom, Sveta (Svetlana L Morozova) he fell in love and subsequently married. Sveta was Kees's constant companion in the later years of his life, and his greatest concern was how she would cope when he was no longer here. Sadly that time has now come, and this short series of recollections is dedicated to Sveta, in Kees's memory.

Farewell, Kees – it was an honour and a privilege to know you and to spend time with you; TEX conferences, and life in general, will never be the same again.

Erik Frambach
Jerzy Ludwichowski
Philip Taylor

TEXShop Review

Abstract
This paper is an introduction to and review of the TEXShop Mac OS X program for typesetting TEX document source files and previewing the output. Features of this tool and user experience will be presented. This information might be helpful to the novice user, looking for a TEX typesetting environment on the Mac.

Introduction

For some decades now I have been using plain TEX and LATEX computer typesetting for scientific documents, lecture notes and other stuff. After starting typesetting on DEC VAX/VMS computer terminals I was pleased to see the integrated development environments for both PC/Windows and Mac OS.

Currently, I use the TEXshop Version 1.42, installed with the teTEX distribution on an old Mac PowerBook G4. My office PC runs a MikTEX distribution with TEXnicCentre (discussed elsewhere in this Maps edition). My first impression was that TEXShop is fairly basic, but reliable.
The next section will describe a non-exhaustive number of features, included in TEXShop, and report user experience.

TEXShop features

This section list essential features of TEXShop, with some user comments added. In a typical typesetting session the program will open three windows, as shown in Figure 1: the source editor, the PDF preview and the console window that report the document compilation progress, warnings and errors.

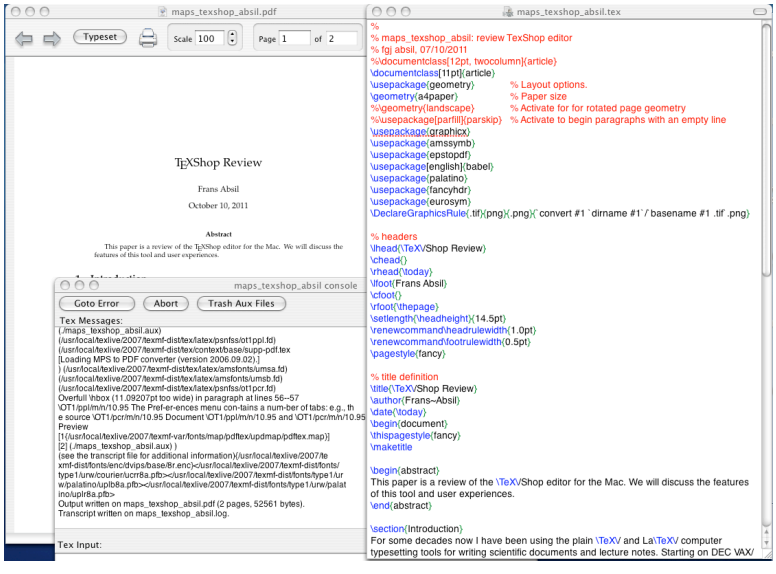


Figure 1. Overview of the three TEXshop windows: editor, previewer and console

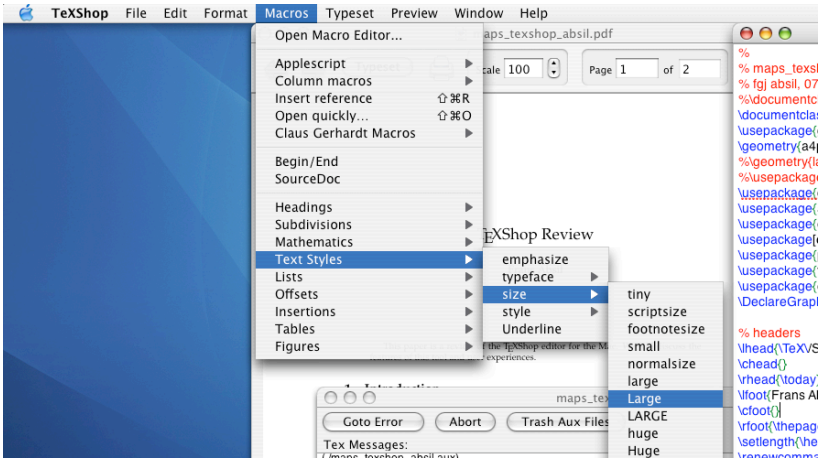


Figure 2. TEXshop **Format** menu with font selection items

Preferences and typesetting engine

The **Preferences** menu contains a number of tabs: e.g., the source `Document` editor font setting and window position, `Preview` window parameter settings, the typesetting `Engine` binary path and and options lists. The default typesetting engine is tick-marked in the `Typesetting` tab. The full set of engines, including TEX, LATEX, ConTEXt, XeLATEX is available in the pulldown menu `Typeset`, where also workflow enhancement scripts or tools such as Pdftex, TEX+Ghostscript, BibTEX and MakeIndex can be found.

As a LATEX user I select either the Pdftex or TEX+Ghostscript typesetting engine; the latter is required for including scientific diagrams created with the `PStricks` package. Obviously in that case all figures have to be available as separate PostScript `.eps` files, whereas for the former option all figures are included as PDF files. And, by the way, drop recognized figure filetypes on the editor window, and it will generate the appropriate `\includegraphics` command.

Editor features

The source file editor uses colour coding, with LATEX commands shown in blue, comments in red, and parenthesis grouping in green, for easy code consistency checks. Although line numbers are not displayed, the **Edit** menu contains a `Line Number` and `Go to Error` command. This menu also contains items for running a spell checker and showing document statistics, such as word, line and character count excluding the LATEX commands.

The **Find** panel, under the **Window** menu, has a number of nifty options: it accepts regular expressions for advanced search, it applies find and replace to selected subregions in the source document (local scope) and it will list all appearances of the search string, a convenient option for navigation and as check before a global replace. It will also remember all previous search and replace strings for the current session.

Panels for the novice user

LATEX commands are mnemonic. However, at the entry level TEXshop contains menu items and special panels for entering source code. Figure 2 shows the menu for font selection. A separate LATEX panel (see Figure 3) shows the most frequently used symbols, environments and other document elements. Clicking an item on the panel will put the LATEX source code in the current document in the editor. TEXshop allows multiple editor windows.

In the **Preview** window there are the usual scrolling, paging and scaling functions. A nice feature is the magnifying glass; when selected clicking anywhere in

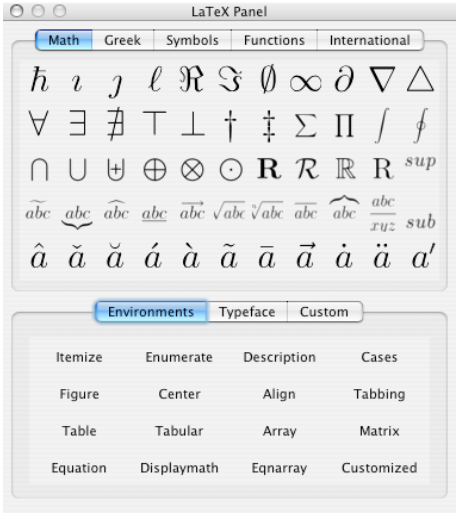


Figure 3. LATEX panel for selection of common symbols and document elements.

the previewer document will show that section in sufficient magnification to inspect typesetting details.

The novice user might also refer to the concise but adequate **Help** menu.

Miscellaneous

The **Format** menu has handy items for (un)commenting or indenting blocks of source code, a feature that is useful when debugging documents.

The **Macros** menu contains items for automating the workflow. My favourite from this set is the `Insert reference`: it opens a window with a list of `\label` commands in the current document for selecting the appropriate entry. Also convenient is the `Bibliography` Applescript, which does multiple runs for creating bibliographical references in a document.

For larger documents, that consist of a set of smaller files (e.g., book chapters), there is the option of setting the project root, i.e., the path to the project `main.tex` file; it is available under the **File** menu. What is missing however, is a window representing the structure of a large-scale project as a tree graph with directory paths to source files, figures etc. TEX/nicCentre has this window that is a great help during editing and debugging report or book class documents.

Conclusion

The old version of TEXShop, running on a Mac OS 10.3.8 operating system, is a reliable workhorse. Although there are differences I have no problem using this tool and TEXnicCentre on the Windows PC in parallel.

A much more recent version, i.e., TEXShop, vs. 3.11 for Mac OS 10.7 (Lion), is available at http://www.uoregon.edu/~koch/texshop/, and on the TEXLive DVD. A comparison between various LATEX editors can be found on http://en.wikipedia.org, containing a table with the features of each.

Frans Absil

TextMate

Power editor for Mac

Abstract
When editing text or code on a Mac, TextMate is an excellent choice. It offers an abundance of features which makes it a great editor for TEX as well as the macro-packages LATEX and ConTEXt. Editing HTML, XML and CSS is also supported, as are many other programming languages. Specially tagged texts are quick and easy to write, thanks to autocompletion and placing of start- and end-tags. Further the editor supports projects — a sidebar to the editing window that shows files belonging to the project. Another useful feature is the column selection method and the (near) end of line selection. TextMate offers a clipboard history, so multiple items can be retrieved. TextMate can be customized to a very large extent.

Introduction

Choosing an editor that suits every project is not easy. While personal taste and required features play important roles, TextMate is a high-end editor for the Mac platform with such a broad featureset that it should make almost anybody happy. It suits many environments from simple text editing to website programming. TextMate has built-in support for both TEX and LATEX. ConTEXt is supported by installing a bundle from Patrick Gundlach, that is also MetaFun aware. A bundle is a collection of shortcuts and automations that make a certain type of text or code editing easier. From TextMate one can run compilers and previews in viewers (e.g. web-browsers, PDF-viewers). To top things off, the editor allows for extreme personalization and customization.

How to get TextMate

The application can be downloaded from http://macromates.com/ for € 45.63 (single user license including VAT). A 30-day trial is also available and special arrangements can be made for multiple user licenses and site licenses.

Documentation

Manuals and other documentation are available online at http://manual.macromates.com/en/all_pages.html and at http://manual.macromates.com/en/.

There is also an exhaustive book written about TextMate: "TextMate Power Editing for the Mac" by James Edward Gray II. The book is available both as paperback and eBook. When buying the eBook, one gets three formats at once: epub (iPhone, iPad, Android and eReaders), mobi (Kindle) and PDF.

Language support

Language is set or extended by using bundles. When editing a file, one can select the bundle to be used from the *editing window*. The bundles menu lists all preinstalled bundles. Via this tab you can start the *bundle editor*, or you can browse through the list. For those who are looking even for more supported languages they can visit http://svn.textmate.org/trunk/bundles/ (official releases only).

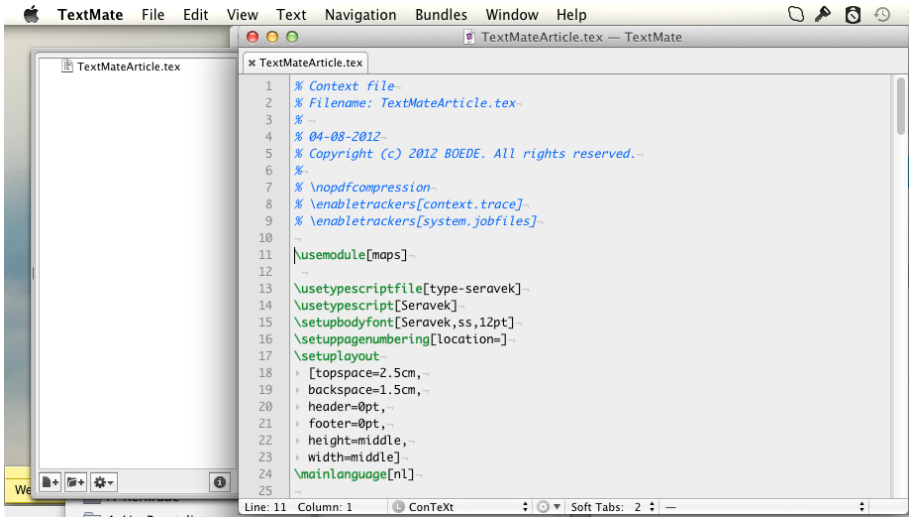


Figure 1. Editing window

The editing window and project drawer

In figure 1 you can see the *editing window* with the *menu-bar* of the application.

On the left side there is the *project drawer*, which is width adjustable to accommodate longer file names. The drawer can be expanded on either side of the *editing window*. The *project drawer* may contain any file or folder from your hard drive that you've associated with the open project. The files can be rearranged in sequence by simply dragging them to another place. The sequence of files and their location on the hard drive are not changed by doing this. Files that are not text files may be marked as binary, and will be opened in their designated application when double-clicked. Projects can be saved for later use, and after reopening the project all files in the project are automatically displayed.

Active files are shown in the top row of the *editing window* as tabs. Files that have been edited but not yet saved are marked by a dark grey dot at the left edge of the tab next to the filename.

At the bottom edge of the *editing window* there is a *status bar* with several fields. The first field indicates the line and column numbers of the cursor position. In the second field the type of file is shown. The type of document can be set by using a drop down menu. Behind the third field is a drop down list containing your bundles, presented in a menu structure for performing actions. The fourth field is used for the tab stop configuration. A nice feature are so called *soft tabs*; while editing they behave like tabs, but show up as spaces in the file.

For languages that support this, the rightmost pop-up in the status bar shows the current "symbol" e.g. a function prototype.

Gutter

Along the left edge of the *editing window* (see figure 1) there is a *gutter*, which displays various information about the file you are currently editing. On the far left there is the column for placing *bookmarks*, which TextMate represents by a star. When clicking in this column the mark is placed, clicking again removes it. Jumping forward or backward through these bookmarks works via shortcuts (F2 or SHIFT-F2) or through the navigation menu.

The second column in the *gutter* shows *line-numbers* and, if turned on, dots for indicating soft-wrapped lines.

The third column contains upward and downward arrows, which allow the collapsing/folding of sections between a downward and an upward arrow. Collapsed paragraphs are shown as the first line of the paragraph with an ellipsis at the end, and a right pointing arrow on a light red background in the *gutter*. Clicking on this arrow or the ellipsis will expand the paragraph. Collapsing paragraphs can also be done through the menu. There are in total 9 levels at which paragraphs can be (un)folded.

Features

Key-bindings are available for all actions, generally speaking. TextMate offers also key-bindings as they are known in emacs.

Selections

Beyond normal *block selections* where whole lines are selected, TextMate offers a *column selection* method. This method can be used e.g. for inserting information on multiple lines at the beginning of the line. This is very handy when e.g. coding a table in TEX using a preexisting table-text. For preparing a zero-width *column selection*, place the cursor in the first position of the last row of the selection. Make the selection you want with the mouse or keys. Then press ALT. Now you see a thin blue line in front of the text. You may now insert information on all selected lines, beginning at the blue line.

Another feature is the possibility to make *end of line selections*. This again is very useful while coding tables in TEX, or while adding the same text several times at once in a data set. Select the lines to be edited. Press ALT-CMD-A, edit the end of the lines.

Editing

- **Clipboard history**
  TextMate keeps a history of text fragments which were copied to the clipboard. CMD-V pastes the last copied fragment. If you want to paste an older fragment, pressing CTRL-ALT-CMD-V reveals the history in a pop-up window which then can be scrolled through. Pressing ENTER places the fragment.
- Freehand (insert) or overwrite mode is provided as in any other editor. ALT-CMD-E enables the freehand mode, ALT-CMD-O switches to overwrite mode.
- **Change case**
  The case of a selection may be changed by using the shortcuts for uppercase (CTRL-U), lowercase (SHIFT-CTRL-U), title case (CTRL-ALT-U) and opposite case (CTRL-G).
- **Goto line**
  Jumping to a specific line in the file is done by pressing CMD-L, which prompts a small window for entering the line number. This is especially advantageous when searching for a line number from the compiling window in a TEX run.
- **Goto file**
  Pressing CMD-T pops up a window, where the files of the project are listed. The most frequently used files appear at the top of the list. Looking for a specific file is made easy by typing the first letters of the filename, or by typing the first letters of words contained in a filename. Clicking the filename opens the editing window of the chosen file.
- **Search and replace**
  TextMate can search and replace within a file. Both a graphical interface and shortcuts are available for this action. Besides regular search and replace in a file, TextMate can also perform search and replace in an entire project. While doing this, TextMate presents a list of files, where the keywords are found. Double-clicking such filenames switches the editor window for that file.

Yet another way of looking for keywords in a file is by scanning. By pressing CTR-S, the status bar changes into a field for entering a search term. Pressing CTRL-S repeatedly moves from one found entry to the next.

□ **Power of regular expressions**
TextMate uses the Oniguruma library for regular expressions. This enables you to do all kinds of operations on the text. Regular expressions are especially beneficiary while coding TEX from a given text. Many things can be found and replaced this way, saving a lot of time in the process.

□ **Customization: bundles**
TextMate is already powerful out of the box, but it can be further improved by bundles. By studying the manual, you can easily build your own automations and place them in a personal bundle. This is good for things that repeatedly need to be inserted into a file. For example, even simple things such as complicated words which are therefore prone to typos can be placed into such a snippet, making it easy to have that word spelled correctly throughout the text.

□ **Customization: templates**
It is easy to add a customized template file to the *File menu → New from Template*. figure 1 shows part of the template file I use for ConTEXt. One advantage is that all your files have the same header information. In my particular template the standard font and the standard layout I use are placed together with `\starttext` and `\stoptext`. Because ConTEXt comes with an impressive list of trackers I inserted the most important ones as comments just beneath the standard header.

Starting TextMate from the terminal

TextMate can be easily started from the terminal. The only thing necessary is a symbolic link in /usr/local/bin. This link is generated during installation or can be installed later on through the help system.

Built-in bundles

It would be beyond the scope of this article to discuss all the included/available bundles.

Basically, TextMate bundles ensure the best for editing in a specific environment. This is evident especially while working with tagged text, where bundles enable the automatic insertion of the start- and end-tag. Syntax highlighting is supported and normally spellchecking does not interfere with commands and tags.

TextMate comes standard with bundles for TEX. There is a TEX — Plain-TEX — and a LATEX bundle. In the status bar of the editing window file types can be assigned to LATEX Beamer or Memoir packages. There is also a bundle covering BibTEX.

It is worthwhile to mention that there is also a bundle for working with Subversion. This makes editing a lot easier because the common commands can be issued within TextMate.

ConTEXt bundle

TeXtMate does not natively support ConTEXt. There is however a very good bundle available, written by Patrick Gundlach. The source of the bundle can be downloaded from http://wiki.contextgarden.net/TextMate. At the bottom of the Wiki page you will find links to the download section.

This bundle allows syntax highlighting for ConTEXt (see figure 1) and MetaFun. There are many automations (snippets) with command-completion. From this bundle you can run ConTEXt-MkII as well as ConTEXt-MkIV. The compilation can be followed in a window (see figure 2). Through CTRL-2 the result can be viewed in the preferred viewer, which can be changed in the bundle editor.

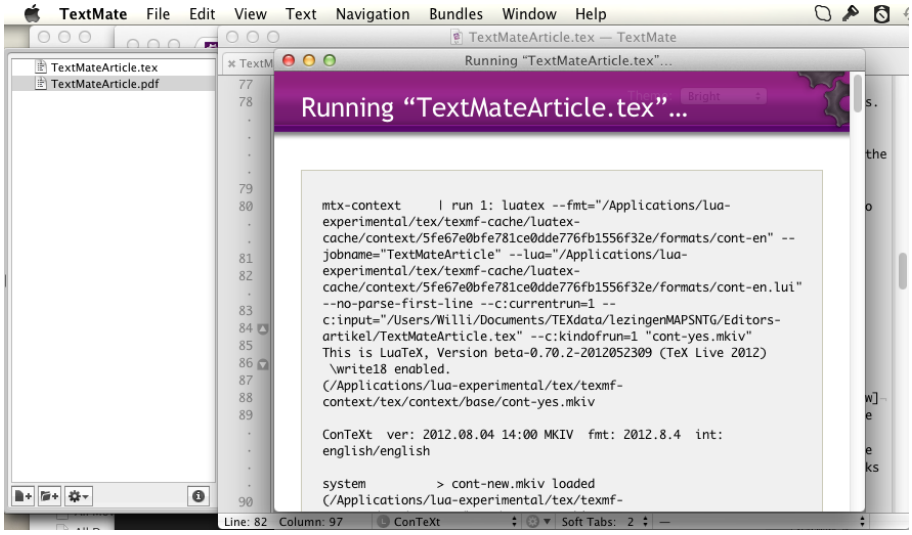


Figure 2. Compilation window

Summary

TextMate is one of the most powerful text editors available today. It offers a vast list of supported text-editing environments and many features that other editors are lacking. One of the most powerful tools is the *project drawer* which makes it easy to keep track of the files in a project. Another convenient feature is that the editor can be customized and adapted to whatever a workflow may require. Some might say that the fact that the editor requires a paid license is a drawback. However, considering what the application has to offer, the price is very reasonable. This article is in no way a comprehensive overview of TextMate's many features, and only highlights a small selection of gems in this editor. There is much, much more to be explored.

Willi Egger
w.egger@boede.nl

TEXworks

Abstract
Van de vele beschikbare TEX- (LATEX-, ConTEXt) editors is TeXworks een vrij nieuwe. Hij munt uit door zijn eenvoud in de gebruikersinterface, de koppeling tussen brontekst en pdf-resultaat en door een grote mate van flexibiliteit. Met behulp van scripts kan men in principe iedere gewenste optie zelf toevoegen.

Keywords
Editor, Synctex, scripts

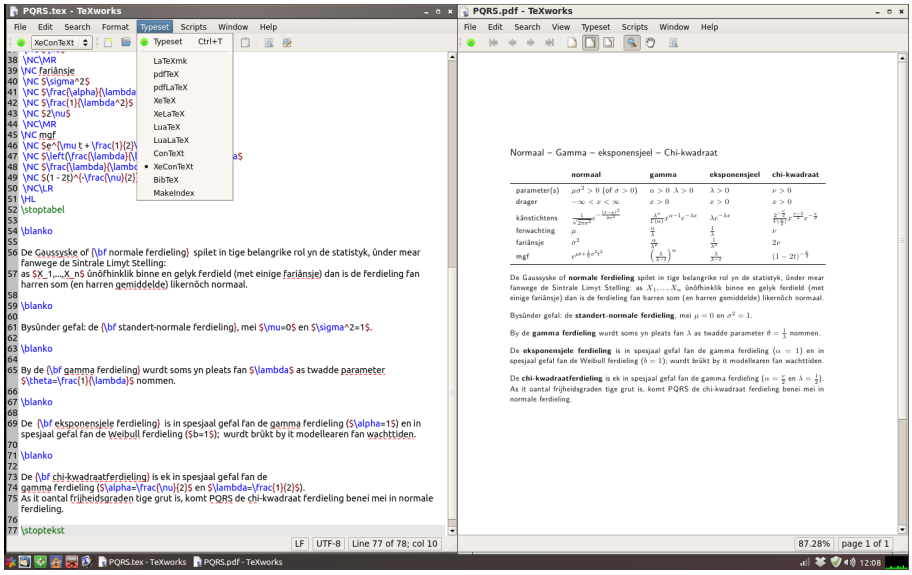
Inleiding

Als kleingebruiker van ConTEXt maakt het me niet zo veel uit welke editor ik gebruik, als hij maar simpel te installeren is en als ik maar brontekst kan intypen en er via een knop of toetsencombinatie een pdf-bestand van kan laten maken. Ook moet er een viewer aan gekoppeld zijn die niet gaat sputteren als bij een nieuwe run de oude versie moet worden overschreven. Eigenlijk geldt voor mij: hoe simpeler hoe beter.

Eenvoud

TeXworks voldoet hieraan. Het is zeer eenvoudig te installeren en te gebruiken, Er is een viewer aan gekoppeld die zonder problemen het pdf-resultaat laat zien, als je wilt met een vergrootglas. Die eenvoud in gebruik komt niet doordat het programma in zijn kinderschoenen staat, ook al is het pas enkele jaren oud. Het is beleid. De makers Jonathan Kew en Stefan Löffler hebben zich laten inspireren door het het succes van TeXshop dat alleen voor het Mac OS X besturingssysteem beschikbaar is. Dat programma heeft TEX (LATEX, ConTEXt) beter toegankelijk gemaakt voor een brede groep nieuwe gebruikers. TeXworks is er voor alle gangbare besturingssystemen, dus ook voor Linux en Windows. Het ziet er voor nieuwelingen op TEX-gebied niet intimiderend uit. Geen scherm vol pop-up-menu's, geen balken met wiskundige symbolen, maar slechts twee sobere vensters, die samen het hele scherm beslaan (zie figuur 1). Links een venster voor de brontekst, met daaronder eventueel een uitvoerpaneel dat weer verdwijnt als de TEX-motor geen fouten heeft gevonden. Rechts een venster met het resultaat in pdf (met opzet geen dvi om nieuwelingen niet te ontmoedigen). Tussen passages in beide vensters kun je gemakkelijk heen en weer springen: met 'Ctrl+muisklik' licht de overeenkomstige passage in het andere venster op. Deze koppeling wordt verzorgd door Synctex.

TeXworks richt zich ook niet zoals veel andere editors eenzijdig op één macropakket. Specifieke LATEX-opties, ontbreken in het menu en ook dit draagt bij aan de overzichtelijkheid. Maar de sobere gebruikersinterface heeft ook zijn keerzijde. Sommige meer geavanceerde of specifieke LATEX-opties die je bij andere editors wel direct in het menu kunt vinden, zijn hier afwezig, zitten verstopt in het menu of moeten met het bewerken van configuratiebestanden worden ingesteld. Dit geldt bijvoorbeeld voor de spellingscontrole en voor het automatisch aanvullen. Tot nu toe had ik aan de standaardmogelijkheden van TeXworks genoeg, maar ter voorbereiding van dit artikel heb ik een aantal extra mogelijkheden verkend.



Figuur 1. Schermafbeelding van TeXworks met uitgeklapt menu *Typeset*.

Spellingscontrole + aanvullen

Ten eerste de spellingscontrole. Normaal gesproken is het me te veel moeite om die te installeren (ik maak nooit spelfouten :)), maar dat kan wel (de spellingscontrole installeren, bedoel ik). Dat gaat bij TeXworks tamelijk eenvoudig: je moet van de gewenste taal een `.aff` en een `.dic` bestand hebben. Ik vond die ergens diep verstopt in mijn Linux-installatie onder `~/.mozilla/firefox`, maar ze schijnen ook te worden meegeleverd met LibreOffice. Vervolgens kopieer je ze naar `/usr/share/myspell/dicts` of je creëert koppelingen ernaar in die map. Als je daarna TeXworks opstart, kun je in het menu onder `Edit -> Spelling` de gewenste taal aanvinken. Het resultaat, met fy (Fries) als taal is te zien in figuur 1. De spellingscontrole negeert alle woorden die beginnen met een '\'; het maakt niet uit of het TEX-, LATEX- of ConTEXt- codes zijn.

En dan het automatisch aanvullen. In een configuratiebestand heb ik ingevoerd: `tw:=TeXworks`. Als ik nu 'tw' intik, gevolgd door 'TAB', verschijnt in plaats van `tw` één van de volgende twee woorden: '`\textwidth`' of '`TeXworks`'. Het eerst woord zit in de voorraad van ruim 600 afkortingen die voor LATEX reeds zijn ingevoerd; voor dit artikel heb ik het tweede woord nodig. Als het verkeerde woord eerst verschijnt, druk ik nog een keer op 'TAB'. Voor ConTEXt bestaat er een script 'autocomplete.js' van Henrik Skov Midtiby (zie onder Bronnen). Nadat dit bestand in de juiste map is gekopieerd, worden je ConTEXt commando's automatisch aangevuld als je één of meerdere keren op 'Ctrl+M' drukt.

Scripts

Dat brengt ons op het onderwerp scripts. De makers wilden hun product uiterlijk eenvoudig houden om beginners niet af te schrikken, maar tegelijkertijd ook de ervaren gebruiker die behoefte heeft aan geavanceerde opties, tevreden stellen. Daarom kan men sinds versie 0.4 scripts toevoegen, in principe in één van de scripttalen QtScript, Lua of Python. In de praktijk blijken vrijwel alle bestaande scripts te zijn geschreven in QtScript (dit lijkt sterk op JavaScript; de bestandsnamen eindigen ook op `.js`). Scripts zijn er in twee soorten: hook scripts en standalone scripts. Hook scripts worden uitgevoerd als er een bepaalde situatie optreedt, bijvoorbeeld direct nadat TeXworks is opgestart of direct na het compileren. Standalone scripts worden pas uitgevoerd als de gebruiker de betreffende keuzemogelijkheid in het menu heeft

aangeklikt. Een meegeleverd voorbeeld is het vet zetten van een stuk geselecteerde tekst. Het script plaatst dan '\textbf{' en ' }' om de geselecteerde tekst. Het gebruik van scripts biedt veel mogelijkheden, maar om ze zelf te maken, moet men de scripttaal goed beheersen. Gemakkelijker is het om reeds gepubliceerde scripts van het internet binnen te halen en in de juiste map te plaatsen. Eventueel kun je zo'n script nog een beetje aanpassen aan je eigen wensen. Zo kun je het laatste voorbeeld geschikt maken voor ConTEXt door '\textbf{' te vervangen door '{\bf '.

Handig voor LATEX-gebruikers is een beschikbaar hook script dat wordt uitgevoerd direct na het compileren (bij de gebeurtenis AfterTypeset). Als er bij het vertalen van de LATEX broncode een fout wordt ontdekt, vind je in het tweede tabblad in het uitvoerpaneel een apart lijstje van de gevonden fouten, zodat je die snel kunt traceren. In het eerste tabblad staan *alle* uitvoermeldingen.

Overzicht

(Nog) niet geïmplementeerd is de mogelijkheid om tekst in- en uit te klappen *(folding)*, zodat je beter overzicht houdt over lange lappen tekst. Maar er is wel een andere manier om dit overzicht te houden: je kunt een linkerpaneel (naast de brontekst) oproepen waarin de structuur van hoofdstukken en paragrafen wordt weergegeven en waarin bovendien eventuele bladwijzers *(bookmarks)* staan. De laatste breng je in je brontekst aan na %: aan het begin van een regel. Hij verschijnt dus niet in de pdf-tekst. Door op een paragraaftitel of bladwijzer te klikken spring je direct naar de betreffende plaats in de brontekst.

Al met al is TeXworks een erg geschikte editor om mee te beginnen als je nieuw bent in de TEX-wereld. Erg plezierig vind ik bijvoorbeeld de mogelijkheid om gemakkelijk heen en weer te springen tussen passages in de brontekst en in het pdf-bestand. Maar ook als je ervaren gebruiker bent zijn de mogelijkheden in principe onbeperkt. Dit komt door de mogelijkheid om scripts toe te voegen die meteen worden opgenomen in het menu en waar je bovendien een toetsencombinatie aan kunt koppelen.

- ☐ http://www.tug.org/texworks/
  Site van TeXworks.
- ☐ http://code.google.com/p/texworks/
  Hier wordt de ontwikkeling van TeXworks bijgehouden.
- ☐ http://www.youtube.com/watch?v=9-Z43CSPgM0
  Lezing van Jonathan Kew voor de TUG 2010 conferentie: *TeXworks for newcomers and what's new for old hands.*
- ☐ http://twscript.paulanorman.info/index.html
  De site voor scripting van Paul A. Norman.
- ☐ https://github.com/henrikmidtiby/TeXworks-scripts
  Bevat onder meer een script 'autocomplete.js' voor ConTEXt van Henrik Skov Midtiby.

Sytse Knypstra
Sytse.Knypstra@home.nl

TeXstudio: speciaal voor LaTeX starters

Abstract
TeXstudio is de primaire LaTeX editor bij de TeX Live installatie van onze universiteit. In dit stuk wil ik laten zien waarom ik voor deze editor heb gekozen en waarom TeXstudio interessant kan zijn speciaal voor de beginnende LaTeX-gebruiker en voor een LaTeX-cursus.

Inleiding

Bij een TeX installatie hoort een TeX editor. Voor onze universitaire TeX installatie wilde ik een editor die vooral geschikt was voor beginners. Ervaren gebruikers kunnen als ze dat nodig vinden altijd zelf overstappen op iets anders.

De keus viel op TeXstudio. Deze editor is open source, beschikbaar voor Windows, Linux en Mac OS X en wordt aktief ontwikkeld. Bovendien is er een USB-stick versie voor Windows. We zullen straks zien wat TeXstudio beginnende LaTeX-gebruikers te bieden heeft.

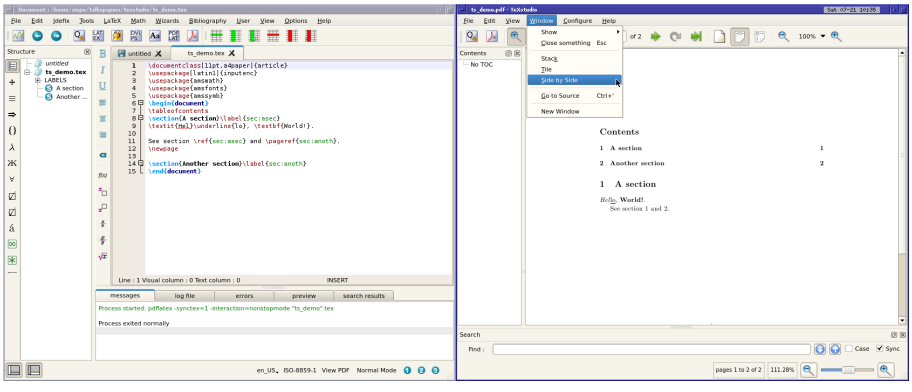
TeXstudio is gebaseerd op Texmaker en heette oorspronkelijk TexMakerX. Veel van wat volgt is evengoed op Texmaker van toepassing.

TeXstudio bevat tal van visuele hulpmiddelen om LaTeX-code in te geven en om documenten te compileren en te voorvertonen.

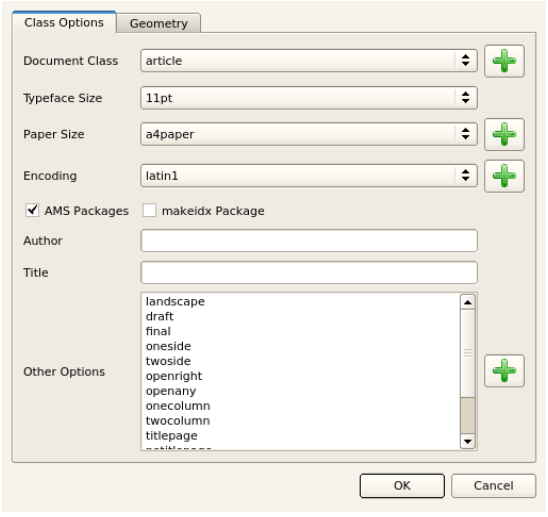
Help en documentatie

TeXstudio help bestaat uit twee html-documenten: een LaTeX naslag[1] die ook wordt gebruikt voor popup help bij het intypen van code (zie pagina 18) en een handleiding voor TeXstudio zelf, met hoofdstukken over onder andere configuratie, editen en compilatie.

Alle knoppen en ikonen hebben een tooltip beschrijving.



Figuur 1. TeXstudio edit- en preview-venster



Figuur 2. De Quick Start wizard

Voor een inleiding in LaTeX zelf moeten we elders zijn, bijvoorbeeld bij *The Not So Short Introduction to LaTeX 2ε*, dat beschikbaar is als TeX Live- en MiKTeX package. Onze TeX-installatie heeft hier een snelkoppeling voor.

Eerste sessie

Het TeXstudio edit-venster bevat diverse vakken en knoppenbalken, waarvan een aantal de revue zal passeren; zie het linker venster in Figuur 1. In deze schermfoto is de overweldigende hoeveelheid knoppen al enigszins gereduceerd middels configuratie.

De schermfoto's zijn deels van versie 2.3, deels van versie 2.4. Deze twee versies verschillen onder andere in ikonen en in menu-struktuur.

Natuurlijk moet er eerst een nieuw document worden aangemaakt. Hier heeft TeXstudio de Quick Start wizard voor, onder het Wizards menu; zie Figuur 2.

Het Quick Start venster laat ons kiezen tussen een aantal class files en opties, voor onder andere encoding, papier en het al dan niet laden van de AMS packages. We kunnen ook een author en title invullen, maar een `\maketitle` aanroep moeten we zelf doen. Als we alleen maar op OK klikken wordt de volgende tekst aangemaakt:

untitled

```
1  \documentclass[10pt,a4paper]{article}
2  \usepackage[latin1]{inputenc}
3  \usepackage{amsmath}
4  \usepackage{amsfonts}
5  \usepackage{amssymb}
6  \begin{document}
7
8  \end{document}
```

Helaas is TeXstudio zo slordig om niet eerst te kijken of er al een document met inhoud is, in welk geval de tekst domweg op de cursor-positie wordt ingevoegd.

Een andere manier om een nieuw document te starten is via File / New from template...; zie pagina 21.

Vervolgens kunnen we tekst invoeren tussen `\begin{document}` en `\end{document}`.

Voorvertonen

Nu er wat tekst staat kunnen we kennis maken met de edit – compile – preview cyclus. Voor 'compile' is er een compile knop (▶/PDF LaTeX), en voor 'preview' een knop met een

vergrootglas of Acrobat logo (/). De Quick Build knop (/) doet waarschijnlijk beide, maar dat hangt af van de configuratie. Er zijn meer specifieke commando's te vinden onder het Tools [/ Commands] menu.

De previewer is overgenomen van TeXworks, en ondersteunt synchronisatie tussen LaTeX bron-code en pdf. De previewer heeft een eigen venster, maar er is wel een handig menu-item Window / Side by Side, dat het hele scherm gelijk verdeelt tussen de edit- en preview venster; zie figuur 1.

LaTeX markup

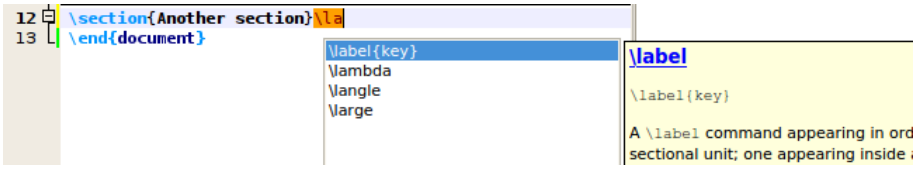
Met dit achter de kiezen kunnen we ons wijden aan het produceren van inhoud met markup.

Vet en cursief. Voor snelle toegang tot vet en cursief is er een vertikale knoppenbalk links van het edit-vak, en desgewenst een uitgebreidere horizontale versie voor de knoppenbalken bovenaan (uitgezet voor de schermfoto's). Als er op dat moment tekst is geselekteeerd dan wordt, zoals verwacht, `\textbf` toegepast op de selektie.

```
\textit{Hel}\underline{lo}, \textbf{World!}.
```

Het LaTeX-menu. Naast knoppen voor veelgebruikte LaTeX macro's is er een LaTeX-menu met een veel uitgebreidere selektie. Dit krijgen we bijvoorbeeld na het aanklikken van LaTeX / Tabular Environment / \begin{tabular}:

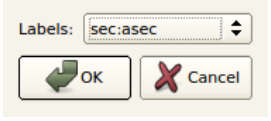
Autocompletion en popup help. Het LaTeX-menu bevat een Sectioning submenu, met sections met en zonder sterretje, maar we moeten er zelf een label bijdoen, desgewenst ook via het LaTeX-menu. Als we zelf beginnen `\label` in te typen dan probeert TeXstudio ons te helpen met een lijstje mogelijke completeringen, plus een toelichting voor de geselekteerde completering:



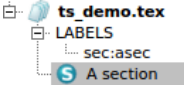
De toelichting bij het `\label`-commando is geplukt uit de reeds genoemde LaTeX naslag. Zolang de ingetypte tekens nog geen geldig LaTeX-commando vormen is de achtergrond oranje, ter waarschuwing.

We kunnen al die drukte uitzetten als het op onze zenuwen werkt. Daarvoor moeten de 'advanced options' echter wel aan staan; zie sectie 'Configuratie'.

Kruisverwijzingen. Verder is er een submenu voor kruisverwijzingen. Als we kiezen voor `\ref` dan krijgen we een lijstje met al bestaande labels:



Met een section en een label ziet het Structure vak links er als volgt uit:

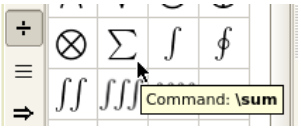


Dit vak kan worden gebruikt voor navigatie binnen het document. Merk op dat het preview-venster hyperref bookmarks ondersteunt voor soortgelijke navigatie binnen de pdf, maar daar is wel het hyperref pakket voor nodig.

Formules. TeXstudio heeft een uitgebreid Math menu, met onder andere inline math, al dan niet genummerde display math en de bekende math construkties zoals bijvoorbeeld `\frac`:

Het `\frac` commando is, samen met enkele andere veelgebruikte wiskundige construkties, ook beschikbaar in de vertikale knoppenbalk links naast het editor vak.

Daarnaast geeft het struktuur-vak toegang tot diverse pagina's met wiskundige symbolen. Deze pagina's kunnen worden opgeroepen met de vertikale knoppenbalk links ervan. Bijvoorbeeld operator symbolen:



Wizards. We hebben al kennis gemaakt met de Quick Start wizard. Maar TeXstudio heeft ook wizards voor het aanmaken van tabellen, inclusief 'arrays' en tabbing-tabellen. We kunnen maar één kolom-type specificeren. Verdere verfijningen moeten achteraf met de hand worden aangebracht.

Ik was blij verrast met de Insert Graphics wizard: we kunnen de figuur via een browser aanwijzen, en er wordt een relatief pad zonder onnodige extensie gegenereerd:

```
\includegraphics[width=\linewidth]{../figure}
```

Bibliografieën

TeXstudio biedt enige ondersteuning voor het maken van een bibtex database, hoewel het niet pretendeert een volwaardige bibliography manager te zijn.

Het Bibliography menu bevat items voor verschillende soorten publikaties. Als we bijvoorbeeld 'Article in Conference Proceedings' aanklikken wordt de volgende code aangemaakt:

```
@InProceedings{ID,
author = {author},
title = {title},
booktitle = {booktitle},
OPTcrossref = {crossref},
...
OPTannote = {annote},
}
```

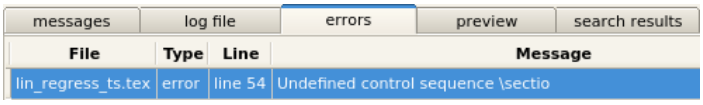
Het is de bedoeling dat we OPT weghalen voor de velden die we feitelijk gebruiken. Als we daarna Bibliography / Clean selekteren dan worden alle `OPT...` velden verwijderd.

Biblatex en biber. Ondersteuning voor biber en biblatex is nieuw in versie 2.4. Op het moment van schrijven is deze versie nog in beta.

Via Tools / Bibliography wordt, alnaargelang de configuratie, hetzij bibtex hetzij biber aangeroepen.

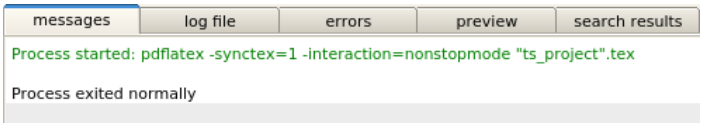
Foutafhandeling

TeXstudio gebruikt het berichtenvak onderaan voor foutmeldingen, bijvoorbeeld:

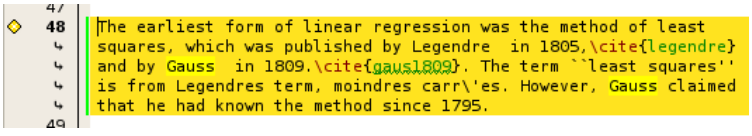


Na een klik op de blauwe balk van de foutmelding springt de cursor in het edit-veld naar de fout.

Na een succesvolle compilatie ziet het berichtenvak rechtsonder er ongeveer zo uit:



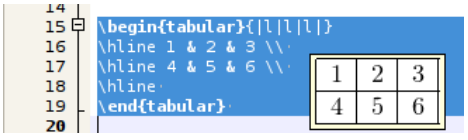
Merk op dat we nu de messages tab zien in plaats van de errors tab. Het is verstandig om toch even de errors tab te inspekteren voor problemen zoals bijvoorbeeld niet-opgeloste kruisverwijzingen. Problemen worden ook in het edit-vak nadrukkelijk gemarkeerd:



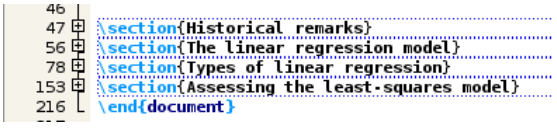
Grabbelton

Hieronder een lijstje met diverse handigheden:

Voorvertoning van een selektie. Een optie Preview Selection/Parantheses in het context menu produceert een voorvertoning van geselekteerde tekst. Deze voorvertoning kan verschijnen als tooltip (zie hieronder), tussen de code of in de Preview tab van het berichtenvak, afhankelijk van de configuratie.

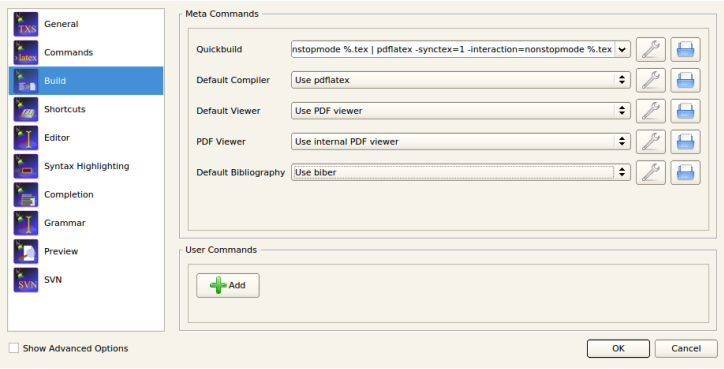


Outline mode. TeXstudio heeft een outline mode via de Collapse- en Expand ondermenu's van het View-menu:



Macros. Met het Macros menu kunnen eigen macro's worden aangemeld en desgewenst van een sneltoets voorzien. Deze macros worden automatisch meegenomen in autocompletion; zie pagina 18.

Tabel manupulatie. Het LaTeX-menu heeft een 'Manipulate Tables' ondermenu voor onder andere het invoegen, verwijderen en plakken van rijen en kolommen. Je kunt er ook de tabel broncode netjes mee uitlijnen.



Figuur 3. Het configuratie-scherm

Configuratie

Het configuratie-scherm

We kunnen allerlei aspekten van TeXstudio aan eigen voorkeuren aanpassen via Options / Configure TeXstudio... Het aantal opties is erg groot, en als we 'Show advanced options' linksonder in het configuratie-venster aanklikken dan worden het er nog veel meer.

De 'General' rubriek bevat een aantal nuttige opties zoals font grootte en default language.

Onder 'Build' kunnen we kiezen voor lualatex inplaats van pdflatex, en voor biber inplaats van bibtex. We kunnen ook eigen commando's toevoegen.

Onder 'Preview' kunnen we behalve de normale document-voorvertoning ook 'Segment preview' configureren; zie pagina 20.

Onder 'Completion' kunnen we aankruisen welke pakketten moeten worden meegenomen bij autocompletion; zie pagina 18. Dit gaat via .cwl-bestanden, die lijsten met macro's bevatten. Ik denk dat ze in de executable zijn opgenomen, want ik kon ze niet in het filesysteem terugvinden, maar ze werken wel. De handleiding legt uit hoe je eigen .cwl-bestanden kunt toevoegen.

Configuratie van templates en Quick Start

In sectie 'Eerste sessie' maakten we melding van de Quick Start wizard en van templates als manieren om een nieuw document aan te maken. Beide methoden kunnen door de gebruiker worden uitgebreid.

Terugblikkend op figuur 2 zien we plus-knoppen () om keuzen toe te voegen. Er is echter geen corresponderende min-knop.

We kunnen ook eigen templates toevoegen door een document op te slaan via File / Make Template... Maar ook hier is er geen voorziening om templates vanuit de GUI te verwijderen.

Het ini bestand

TeXstudio slaat configuratie-gegevens op in een bestand `texstudio.ini` in een map onder de home- of profile map van de gebruiker. Het heeft de struktuur van een ouderwets Windows ini-bestand en kan zonodig handmatig worden gewijzigd, bijvoorbeeld voor beheer van templates; zie hierboven.

De usb-stick versie bewaart de ini-file bij de rest van het programma.

De Windows-versie maakt geen gebruik van het register voor configuratie, behalve voor aanmelding van de uninstaller. Er worden ook geen bestands-typen gedefinieerd.

TeXstudio, Texmaker en Kile

TeXstudio is een afsplitsing van Texmaker en heette oorspronkelijk TexMakerX. Pascal Brachet, de auteur van Texmaker, is tevens de oorspronkelijke auteur van Kile, een LaTeX editor voor KDE/Linux. De drie editors hebben dan ook een sterke familie-gelijkenis.

Kile viel af vanwege KDE. Installatie van Kile en KDE onder Windows is wel mogelijk maar niet simpel.

Een sterk punt van TeXstudio is het open ontwikkel-proces, dat op SourceForge kan worden gevolgd, inclusief diskussies en bug reports.

URLs

Oetiker, Tobias e.a. (2011). *The Not So Short Introduction to LaTeX 2ε*. Included in most free TeX distributions. URL: http://mirror.ctan.org/info/lshort/.

Texmaker Home. URL: http://www.xm1math.net/texmaker/.

TeXstudio Home. URL: http://texstudio.sourceforge.net/.

TeXstudio Project Page. URL: http://sourceforge.net/projects/texstudio/.

Notes

1. Deze help file is zo te zien gebaseerd op hoofdstuk 2 van de latex2e-reference op CTAN, auteur onbekend.

Siep Kroonenberg
n dot s dot kroonenberg at rug dot nl

PSlib.eps Catalogue, preliminary and abridged version

— *use of PostScript libraries* —

Abstract
A selection of PostScript definitions collected in my PSlib.eps library and documented as e-book catalogue is presented. Now and then variant pictures have been included from pic.dat which comes with Blue.tex. Old Metafont codes have been included which may be useful for MetaPost programmers. Variants of pictures enriched by postprocessing in Photoshop show other possibilities. Escher's doughnut is a teaser which has to be done in MetaPost. Next to PSlib.eps comes the file PDFsfromPSlib, which contains the pictures in .pdf format. The complete PSlib.eps, PDFsfromPSlib as well as the catalogue as e-book, will be released on occasion of NTG's 25th lustrum which will be celebrated in the fall of 2014, on www.ntg.nl. A prerelease will be offered to the GUST's file server. The (static) library for TEX alone pictures, pic.dat, packaged with Blue.tex, will be redistributed as well.

Keywords
Apollonius, Bluebook.eps, Cantor, CD-DVD label, Deubert, Escher, flowcharts, fractals, Gabo, Julia, Koch, Lancaster, Lauwerier, length Bézier curve, Lévy, Lindenmayer, Mandelbrot, Mondriaan, Op Art, orthogonal circles, Photoshop, pie chart, pi decimals, PostScript, projection, PSlib.eps, Pythagoras Tree, smiley, Soto, stars, (plain) TEX, text along path, tiling, toroid, SoC (Separation of Concerns), Schrofer, Vasarely. Yin Yang

Contents

Introduction

Gutenberg organized in Toulouse in the 90-ies a meeting themed Graphics and TEX. Later, especially after MetaPost had been released in the public domain, GUST paid attention to graphics. Piotr&Piotr developped PSView. LATEX came with its picture environment and Knuth c.s. developed the gkp-macros with the picture environment functionality for use in plain TEX. Blue.tex appeared with its library pic.dat of TEX alone pictures, biased by the gkp-macros. On the EuroTEX92 in Prague Karel Horak showed his math pictures created by Metafont.[1] Later Timothy van Zandt released

1. Karel works nowadays in MetaPost, with .eps and SVG output.

PSTricks, which uses PostScript as the graphics engine and LaTeX's picture environment as user interface. Herbert Vo maintains and extends PSTricks. PostScript was there all the time, since 1985 to be precise, but ...was apparently overlooked for graphics, except by Don Lancaster, who came with his PSSecrets and Gonzo collection.[2] PostScript was in use as (abstract) page description language for the upcoming laser printers.

In relation to TeX, we could in the mid-90-ies include .eps pictures in documents via the psfig command, to be processed by DVIPS.[3] Jackowski developed MFtoEPS, to transform Metafont pictures into PostScript for inclusion in TeX documents. Alas, pdfTeX does not allow direct inclusion of .eps files, a retrograde. Happily, ConTeXt not only allows inclusion of .eps files, but also works interactively with MetaPost.[4]

Of my ≈25 years of involvement with, and using, TeX, I spent ≈5 years on developing Blue.tex and ≈20 years on creating and collecting pictures. Lauwerier's BASIC pictures, especially his fractals, I have been translated into PostScript. As an aside I exercised HTML for my WWW-site.

PSlib.eps was introduced at the EuroTeX2009. The library facilitates programming in PostScript. Hundreds of codes for the pictures are stored in a ordered way, facilitating its use. The catalogue is the accompanying documentation,[5] where next to the code the result of the code has been shown. It contains Adobe's Blue book defs as starting point. Much more, what I needed, has been added. The collection has been tested by example. I don't claim that it has been thoroughly tested nor is the collection robust, although the numerical concept robustness does not apply to graphics, in general, it occurs. Since EuroTeX2012 it has grown substantially. Older PostScript codes with local dictionaries have been added. For the 25th lustrum meeting of NTG, in the fall of 2014, I decided to make the library more easily available, and come up with a catalogue, such that the desired code for the wanted graphics can be more easily spotted and used, hopefully. Numerici have their program libraries, we should have our catalogues for pictures, to start with Adobe's Blue book collection and my PSlib.eps extensions. I consider catalogues for graphics useful, because we lack a taxonomy, the world of pictures is too diverse. May the catalogue contribute to the "literature" of PostScript graphics programming. May PostScript start a second glorious graphics life. Hang on and enjoy!

Why PSlib.eps?

The idea of a library of pictures I already realized in Blue.tex in the 90-ies. The collection is called pic.dat and will also be redistributed along with PSlib.eps. It contains graphics by TeX alone via the macros of Graham-Knuth-Pasternak, the gkp-macros.

PostScript is a powerful language for graphics. But ... it is not enough. Extended with defs as given in Adobe's Blue book makes the use of PostScript feasible. Reality has it that Adobe's Bluebook is apparently not enough for programming in PostScript. Why?

My answer is

- ☐ there are not enough Blue book defs, there are hardly no contributions from the community
- ☐ the viewer and convertor Acrobat Pro is not in the public domain; GhostView and PSView appeared later

2. Gonzo utilities facilitate using PostScript as formatter for texts. Don does not use local dictionaries for library robustness.
3. My papers of that time can't be processed by pdfTeX. -;(. They have to be adapted and the pictures must be converted into .pdf, .jpg, ...
4. An example where TeX-Graphics interaction occurs naturally is in typesetting crosswords. But, ...a crossword is a simple graphic, so we can do with TeX alone.
5. Physical thick, logical thin!

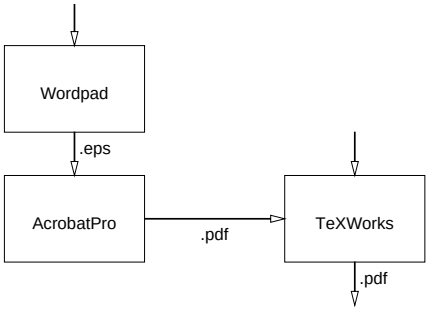
- the defs have not been made easily available, there was no PD PostScript library file on the WWW
- MetaPost is of higher level, with its inheritance of the beautiful Metafont language, which is more in use than PostScript, within the TEX community, Knuth included
- Adobe has abandoned Type 1 fonts in favour for OTF, and seems no longer to support the run command, which I use for library inclusion[6]
- and, indeed ... PostScript is unusual, low-level, difficult and error-prone to program in, alas. But ... if you do, the rewards are more than enough. We should finish up our programming of pictures by collecting the pictures and their codes into a library and document it, in for example the accompanying catalogue.

MetaPost has an explicit datastructure for paths which comes with nice operators such as cutbefore, cutafter, intersectionpoint, interpath, and intersectiontimes, which I used in the MetaPost code for Escher's doughnut in order to cope elegantly with the hidden lines. PostScript has the arc operator, for circular arcs. PostScript has no explicit paths. Nor has it a picture datastructure. PostScript has the powerful flattenpath which delivers an arbitrary path as a broken line, of which the points can be used by pathforall. By these tools the length of a path can be (approximately) calculated by Blue book's pathlength, for example. The absence of the path and picture datastructures in PostScript is a big lack. We will concentrate on graphical shapes and leave the glorious laserwriter history for what it is. Having said that, let us move on and see what kind of graphics can be written in PostScript.

However, ... 20 years after the birth of MetaPost, I still don't have a pleasing MetaPost working environment on my PC.[7] The TEXlive DVD does not provide me such an IDE, except as integrated part of ConTEXt. Besides, much graphics can be programmed in PostScript, as is demonstrated in my PSlib.eps. A notational exception is the Escher doughnut, and ilks, with its hidden lines.

In my PSlib.eps catalogue the Blue book defs have been made digitally available and extended by local dictionaries. This collection has been extended by defs of my own and some collected from the WWW.[8] A couple of hundreds of defs emerged over the past 20 years. It has been strived after that defs can be read, understood, reused and adapted to your circumstances. Simplicity of use is paramount. The accompanying PDFs have also been collected and can be reused as such. Some graphics programs are no longer relevant, for example Adobe's program for a pie chart, because of special and easy to use packages, not in the least the wealth of LATEX packages.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Flowchart TeX processing and PS inclusion
%%BoundingBox: -101 -31 102 221
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
flowchartTeXandPS   showpage
%%EOF
```



6. Your old Acrobat Pro 7 is a treasure for converting .eps into .pdf and viewing it. Cherish Acrobat Pro 7!
7. Troy Henderson has his MetaPost viewer, which can be accessed and used remotely. It is just a pity that he has not released a local, strand-alone version.
8. Operators given in Lancaster's PSSecrets are hard to read and yielded non-working results. For example his Archimedian spiral, where he did not use polar coordinates, is unnecessarily complicated and did not work. On the other hand his Fonts for Free are a gem.

The first 6 lines are standard overhead for the kind of file, `!PS` means PostScript, providing values for the `BoundingBox`, specifying the library and its loading via `run`, cropping of the graphics to the `BoundingBox`, via the `BeginSetup` and `EndSetup` duo, `BeginProlog` and `EndProlog` enclose the needed `defs` (for example the library) and finally the invoke of the `def`: `flowchartTeXandPS`.

For the invoke one just has to known the name of the library `def` and its parameters. The parameters are documented in the code. The code is as extra supplied in the library contributions in this catalogue. Adding the library definitions to the example of use makes the example look more complex, but ... has been done on purpose to facilitate adaptability, sacrificing simple look-and-feel. However, this approach supports stand-alone use.

Processing the graphics separately and include the resulting `.pdf`, or converted `.jpg` which is more efficient in TEXworks, into your AnyTEX(, or Word, or ...) document, adheres the Separation of Concerns principle. A wider audience than the TEXcommunity, such as PostScript users, users of Word, ..., is served.

`PSTricks` based packages look nice and easy, but they suffer intrinsically from the deficiency of LATEX's picture environment, as 'graphical language,' such as the limited orientation of lines. But, ... the advantage is that the user is hidden from PostScript, does not have to know PostScript.

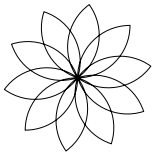
Why PostScript?

The use of a programming language has two aspects: the richness and power of the language proper and the ease of working in an IDE.

PostScript language The language was designed by Adobe, and introduced in 1985, as a device-independent page description language with powerful graphics capabilities, with initially ≈ 400 operators — i.e. built-in procedures of the system dictionary — in PostScript level 1, with substantial more in PostScript level 3. The extensive graphics capabilities are embedded in the framework of a general-purpose programming language. The language includes a conventional set of data types, such as numbers, booleans, arrays and strings; control primitives, such as conditionals, loops and procedures; and some unusual features, such as dictionaries, font transformation matrix, next to higher-level structures, such as patterns and forms. These features enable application programmers to define higher-level operations that closely match the needs of the application and then to generate commands that invoke those higher-level operations. The LRM 3 is the defining document for the syntax and semantics of the language, the imaging model, and the effects of the graphics operators.

Powerful concepts

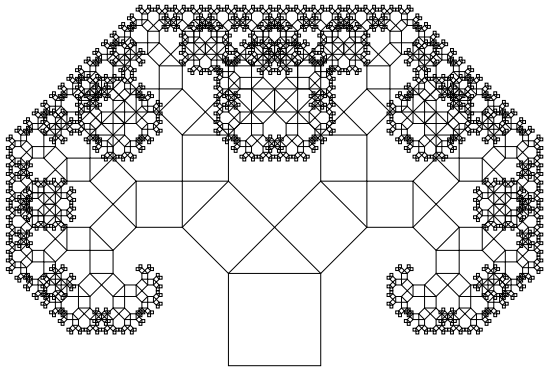
- A user space which can be altered: "the coordinate system's origin may be *translated*, moved to any point in user space; the axes may be *rotated* to any orientation; the axes may be *scaled* to any degree desired; the scaling may be different in the x and y directions." Reflection and skewing are also supported. My favourite illustrations of the US concept are a stylistic flower and the recursive programming of the Pythagorean tree, which is all about drawing a square in repeatedly transformed US.



← Stylistic Flower

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

Pythagorean Tree →
BachoTEX 2011 pearl



Another nice example of the usefulness of transforming US is to create a path of an ellipse by the use of the `arc` operator, for example `1 2 scale 0 0 25 0 360 arc` (Courtesy the Blue Book, but ... be aware of the fact that the pen width has been scaled too).
To translate the centre of the coordinate system, default in the left lower corner of the current page, was the first thing I used to do. No longer necessary for my stand-alone illustrations in an EPSF program, which begin with the 4 lines as given in the stylistic flower example.

□ The colour spaces, which notion was introduced in PostScript 2 of 1991, and elaborated upon in 1997 in PostScript 3, with among others much more efficient shading functionality. In PostScript 1 there were already the concepts colour mode and half-tones, with operators `setrgbcolor` and `setgray`, which were generalized in level 2 into `setcolorspace` with `setcolor`. The chapter headings of the level 1 Red and Blue Books reflect the gradients, or smooth shading, functionality.

Inspired by The Green Book p139
Much can be learned from this example
which was state of the art in 1985
Level 3 features rich colour shading

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 200 36
/DataString 256 string def
/oshow {true charpath stroke} def
/H20 {/Helvetica-Bold 20 selectfont} def
/H8 {/Helvetica 8 selectfont} def
%%EndProlog
0 0 moveto
gsave 175 36 scale
0 1 127 {DataString exch dup 128 add put}bind for
128 1 8 [128 0 0 1 0 0] {DataString} image
grestore
0 0 200 36 rectstroke 1 setgray
H20 95 14 moveto  (PostScript) show 0 setgray
    95 14 moveto     (PostScript) oshow
H8  95  4 moveto 2 0 (Language)   ashow
showpage
%%EOF
```

The number of pages of the PostScript level 3 LRM has increased to 912p, while the number of pages of the level 1 LRM is 321p. See Appendix A of the LRM 3 for the ways the PostScript language has been extended with new operators and other features over time. The language versions are upward compatible.

- ☐ Handy and useful optimizations, such as `rectstroke rectfill userpath selectfont` ...
  but ... don't use `store` in library procedures instead of `def`, with variables which **are intended** to have a local scope. Use a dictionary local to the library procedure for variables.
- ☐ The graphics state, with commonly used operators `gsave grestore` ... .
- ☐ The current page, the abstract page, the raison d'être concept behind PostScript, to be rendered by the interpreter in the rendering device (printer, screen, ...), commanded by the `showpage` operator.
- ☐ Variability of fonts by the font transformation matrix as argument of `makefont`, with scalability, mirrored fonts (as in XƎTEX) smallcaps, and outlines as obvious examples.
- ☐ Stacks `operand dictionary graphics state execution`
- ☐ Immediately evaluated names, i.e. `//name` is immediately replaced by its value; useful for named constants.
- ☐ `bind` operator which looks up values of the **operator** names in the procedure operand and replaces these by their values, the so-called early binding, and returns the modified procedure. It enhances efficiency and robustness against (unintended) change of used operators, especially in library procedures. Optimize loop bodies too by `bind`.
- ☐ Idiom recognition feature of Level 3. A functionality added to the bind operator, which can be switched on/off by setting the user parameter `IdiomRecognition`. Bind can find and replace commonly occurring operators, called *idioms* by operators of higher quality and/or better performance. For example PostScript level 2 shading operators are replaced by PostScript level 3 improvements, silently behind the scenes, with new snazzy codes.
- ☐ `execform` for repeatedly placing a graphic, e.g. a logo, a fill-in form, ... efficiently (since level 2).

Operator groups, with a few operators named from each group:

- ☐ operand stack `pop exch dup ...`
- ☐ arithmetic and math `add div ... srand`
- ☐ array `array ... getinterval`
- ☐ dictionary `dict ...dictstack`
- ☐ string `string ... run ... token`
- ☐ relation, boolean, and bitwise `eq ... bitshift`
- ☐ type, attribute, and conversion `type ... cvs`
- ☐ file `file = == ... echo`
- ☐ file inclusion `run`
- ☐ virtual memory `save restore ...`
- ☐ miscellaneous `bind null usertime version`
- ☐ graphics state `gsave grestore ... currenttransfer`
- ☐ coordinate system and matrix `matrix ... invertmatrix`
- ☐ path constructing `newpath`, `moveto lineto curveto arcto ... clip eoclip ... charpath ...`
  Only one path is possible, although by juggling with several graphics states one may maintain several collateral paths
- ☐ painting to the current page of
  – paths `stroke` paints lines, `fill eofill ...` fills an area
  – strings `show ...`
  – a sampled image `image shshow ...`
- ☐ device setup and output `showpage ...`
- ☐ character and fonts `findfont scalefont setfont` (or level 2 onward optimized concatenation of the 3 into `selectfont`) `makefont` (transforms more general than `scalefont`) `... kshow ... cshow ...`

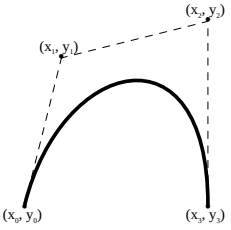
□ font cache `setcharwidth ...`
□ errors `dictfull ...VMerror`.

Overwhelming, isn't it?

Let us pick out a few, which I use most of the time, apart from the arithmetic, math and relation operators, whose use we are already familiar with from our favourite programming language.

`def`	associates names with procedures or values available on the operand stack, and stores the associated pair in the current dictionary
`run`	includes the file given in parenthesis
`moveto`	creates the starting point of an (internal) path
`stroke`	and ilks, to paint the path to the current page
`lineto`	adds a line to the current path
`curveto`	adds a curve to the current path
`arc, arcn`	adds a circular arc to the current path
`image`	to render the (bitmap) image onto the current page
`gsave`	pushes the current graphics state on the gs-stack and creates a new current one
`grestore`	pops the (top) graphics state off the gs-stack and makes it the current graphics state, en passant obseleting the graphics state in use
`makefont`	is used by Don Lancaster (and undoubtedly by others) for creating a variety of fonts from the available ones. He calls his collection "Fonts for Free". I love his embossed variant
`kshow`	I used kerning show in my BachoTEX 2010 pearl for the typesetting of π-decimals along a spiral
`forall`	handy for creating concise code, used for example in my BachoTEX 2010 pearl.

curveto



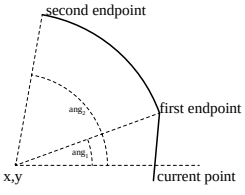
The language (version 3) is stable and flexible, also called extensible, meaning one can add procedures. It is the industry page description standard language. Programs are interpreted, line by line, not compiled, which at the time was important because of small memories. It is maintained by Adobe — the stewards of PostScript — and already with us for more than 25 years! Interpreters are generally provided by 3^{rd} parties, especially the interpreters which come with printers.

For graphics `lineto`, `curveto` and `arc`, and ilks, is what makes the drawing. Let us assume a_0 is the current point.

a_1 `lineto` adds a straight line to the path.
A point $z(t)$, t∈[0,1] on the line is given by $z(t) = (1-t)a_0 + ta_1$.

a_1 a_2 a_3 `curveto` adds a B-cubic to the path with end point a_3 and control points a_1 and a_2.
A point $z(t)$, t∈[0,1] on the B-cubic is given by $z(t) = \sum_{k=0}^{3}(1-t)^{3-k}t^k a_k$.

x y r ang_1 ang_2 `arc` appends a counterclockwise arc of a circle to the current path, possibly preceded by a straight line segment. The arc has (x,y) as center, r as radius, ang_1 the angle of a vector from (x,y) of length r to the first endpoint of the arc, and ang_2 the angle of a vector from (x,y) of length r to the second endpoint of the arc.



Knuth, in the Metafont book, paid much attention to a path through a set of points. A smooth curve through a set of points is not available in PostScript, as such.

The Red, Green and Blue books — Reference Manual, PostScript Language Program Design, respectively Tutorial and Cookbook — are published by Addison-Wesley and also available for free on the WWW, even the level 1 and 2 (774p) LRM's. The Blue Book, which was aimed at to set a standard for effective PostScript programming, contains examples of procedures, such as `oshow` `outsidecircletext` `insidecircletext` `pathtext` `pathlength` `printposter` `DrawPieChart` `centerdash` `smallcaps`

and arrow, to name but a few.[9] The Red Book is indispensable. The Green Book is about software engineering in PostScript, not only to get the programs to work, but to create correct, readable, efficient, maintainable and robust PostScript programs. The underlying goal is to develop a printer driver. There is also an Adobe White Book about Type 1 fonts, also for free on the WWW. Mnemonics: the RGB-collection of Adobe :-). PostScript programs, in ASCII, are generally generated by programs, hardly self-written. They facilitate exchange of (stand alone EPSF) picture descriptions, and of course (the pages of) a complete publication. The structuring conventions of Appendix C of the Red book level 1 have grown out into an Adobe Technical note #5001, 109p. Nowadays, illustrations are easily exchanged in .pdf, and everybody can view them everywhere, because of the ubiquitous, free Acrobat reader. The Mathematica reader is also freely available, and facilitates the exchange of Mathematica notebooks. The exchange of ASCII PostScript goes without problems and is useful.

Although a powerful graphical language, PostScript is considered low-levelish by the TEX community at large. They favour John Hobby's preprocessor MetaPost, Knuth included, with exceptions: Taco Hoekwater, me ... . Taco includes PostScript on-the-fly in escrito, his PostScript compatible interpreter in Lua. He is also on the MetaPost maintenance team and works on extensions of MetaPost. At BachoTEX 2010 he announced the release of MetaPost2.000.

"PostScript is underestimated and underused," to quote Taco Hoekwater.

"I agree with him ... I'm not saying he is right ;-)," well ... he is.

IMHO, a little bit of PostScript does not harm. On the contrary, you will benefit from the general-purpose programming language framework, the imaging model, or you may extend your knowledge about the interactive system for controlling raster output devices, but ... self-study is dangerous. What we need is a teacher à la John Deubert who thoroughly understand the PostScript concepts. John in his Acumen Journal pays attention to among others the relation of PostScript to PDF and XPS, and gives many nice, good and useful examples, clearly explained line by line.

From the LRM: "PDF lacks the general-purpose programming language framework of the PostScript language. A PDF document is a static data structure that is designed for efficient random access and includes navigational information suitable for interactive viewing."

Finally, and in contrast with TEX, a mnemonic

All what you type in PostScript are

- ☐ comments after %
- ☐ numbers
- ☐ operators
- ☐ names to be looked up in the dictionaries, and at last
- ☐ strings which contain text.

From the LRM

"The interpreter manipulates entities called PostScript objects. Some objects are data, such as numbers, boolean values, strings, and arrays. Other objects are elements of programs to be executed, such as names, operators, and procedures."

In TEX the source is the text of the publication, interspersed with as few markup commands as possible, at least that is what Knuth aims at in his minimal markup style, which I love and practice too.

So ...

`add def ...` are names to be looked up in the dictionaries

`( < / [ ...` are operators:

9. The procedure texts and the examples are available on the WWW as a UNIX shell archive. I have assembled the procedures into a BlueBook.eps library, which is system independent and also incorporated in PSlib.eps.

- `(` starts a string, where all is interpreted as text, with `\` as escape character; a TEXies niche
- `<` starts a hexadecimal string, consisting of the "digits" in the hexadecimal system `0,1, ... 9, a, b, c, d, e, f`, commonly used in the executable array, ie, procedure, as argument for the `image` operator
- `/` starts a literal name
- `[` starts an array, which may contain heterogeneous elements, in contrast with other languages.

Be aware of the difference between `name` and `/name`. The first is a `name` to be looked up in the dictionaries, while the slash in the second starts a *literal* name, which is only pushed on the operand stack, without execution. Unlike other programming languages such as PASCAL there are no reserved words.

Comments start with % Comments are used for structuring. Special comments are

`%!` the start of a PostScript program with `%!PS-Adobe-3.0 EPSF-3.0` the complete line for illustrations to be encapsulated

`%%` at the beginning of a line starts structural information about the PostScript program, as explained in Appendix C of the LRM version 1, or see ATN #5002; syntax `%%<keyword>: parameter values`.

EPSF Encapsulated PostScript File format Looking more closely at the `.eps`, which resulted from `.mp`, I found that header comments of a PostScript program starting with the error-prone mumbo jumbo

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: llx lly urx ury
%%BeginSetup
%%Endsetup
```

yields the image cropped to the supplied `BoundingBox: llx lly urx ury`, centred on the page, when processed by Acrobat Pro 7.1.[10] Explicit translation of the origin via `... translate` is no longer necessary in an EPSF with a BB around the origin. Very Handy! Not only very handy ... also with better results than my old way via selecting, copying and the reuse of the copy from the clipboard. The dimensions of the BoundingBox can be requested for in the program, to create a perfect cropped illustration.

From Adobe Technical Note #5001 PostScript Language Documents Structuring Convention Specification, the following about the keyword `EPSF-3.0`

... `EPSF-3.0` states that the file is an Encapsulated PostScript Format, which is primarily a PostScript language file that produces an illustration. The EPS format is designed to facilitate including these illustrations in other documents. ...

Appendix H of the LRM version 2 details on EPSF, which by the way is not included in the LRM 3.

Contents of the library

`PSlib.eps` contains constants, `defs` for operations, `defs` for pictures, and dictionaries which are coupled to the `defs`.

Constants are for example

```
/sqrt2 1.414213562 def
/sqrt3 1.732050808 def
/sqrt5 2.236067977 def
```

10. An undocumented feature? Non-universal?

```
  /pi 3.14159265358979 def
  /reus 2 30 exp  dup 1 sub add def
```
i.e. 2.14748e+09.

□ font abbreviations, such as `/H14pt /Helvetica 14 selectfont def`

□ colour presettings, such as `/lightred{1 0.3 0 setrgbcolor}`.

Definitions for operations, the utilities

Definitions for 2D pictures For example the flowcharts, Escher, Soto, Schrofer, ...

Definitions for 2D pictures as projection of 3D emulated objects For example a sphere, pictures of the emulations of Gabo's constructions.

```
/linearconstructionno1{linearconstructionno1dict begin
 /phi 30 def /theta 5 def /scalingfactor 50 def
reversevideo 1 setgray%white
3  setlinewidth frame        stroke
2  setlinewidth approxellipsestroke %sampleellipsestroke
.5 setlinewidth dostringing  stroke
annotation
end}def
```

Definitions for dictionaries The dictionaries contain auxiliaries which are declared, initialized and kept local. For example the dictionary for `linearconstructionno1` has the components[11]

```
/linearconstructionno1dict 50 dict def

linearconstructionno1dict begin
/reversevideo {-130 -135 260 270 rectfill} def %Mimics BoundingBox
%Data
/origin { 0 0 } def
/a0f  { .1 s  -2 s  -2 s ptp  }def%y constant
/a1f  { .6 s  -2 s  -1 s ptp  }def
/a2f  { .6 s  -2 s   1 s ptp  }def
/a3f  { .2 s  -2 s   2 s ptp  }def
...

/sampleellipsestroke{gsave -45 rotate .6 1.2 scale%kind of ellipse
...
stroke grestore}def
%
/approxellipsestroke{gsave -45 rotate .6 1.2 scale%kind of ellips
...
grestore} def

/frame{a0f moveto  a1f  a2f  a3f curveto
...} def

/dostringing{0.02 .02 .5001{/t exch def
...
closepath }for } bind def

.1 setlinewidth stroke

/annotation{-32 -125  moveto ...}
 ..} def
end%linearconstructionno1dict
```

11. Much of the contents had been omitted here because it is about structural elements.

Structure of each listed code contribution
The approach is similar as adopted in Adobe's Blue book. Of each code included in the library, `PSlib.eps` cq `Bluebook.eps` the example of use is listed. The code is also included. The library codes used are mentioned in the code now and then. At the right of the code the resulting picture is shown. Variants are included.

Utilities
Just a few utilities have been selected and discussed.

Length. A too simple operator? Not so.

One must circumvent intermediate (numerical) overflow, and... use the stack only. Moreover, the name `length` is already in use as a so-called polymorphic operator — takes different kinds of arguments — also called an overloaded operator in Ada, for example.

$$\begin{aligned} |(x,y)| &= \sqrt{x^2+y^2} \\ &= |y|\sqrt{1+(x/y)^2} \qquad \text{numerically better if} \quad |y| \geq |x| \\ &= |x|\sqrt{1+(y/x)^2} \qquad \text{numerically better if} \quad |x| \geq |y| \end{aligned}$$

```
/size %x y ==> sqrt(x^2+y^2)
      %not robust against 0 0 on the stack
{abs dup  3 -1 roll abs dup 3 1 roll % y x y x
le {size}                           % y <= x
   {exch dup 3 1 roll               % y x y
     div                            % y x/y
     dup  mul 1 add sqrt mul
    }ifelse
}def
```

The operator is related to the polar coordinates (r, ϕ) of a point (x, y). In PostScript the angle ϕ can be obtained via the `atan` operator; for the size r one has to provide an operator, length, oneself.

Overloading length[12]

```
%!PS Overloading length. Taco Hoekwater April 2010
/PSlength {length} bind def % save old meaning
/lengthdict 5 dict def
lengthdict /arraytype   {PSlength} put
lengthdict /dicttype    {PSlength} put
lengthdict /stringtype  {PSlength} put
lengthdict /integertype {size} put
lengthdict /realtype    {size} put
/length { lengthdict begin dup type exec end} def
```

```
%Test
(whatever) length pstack pop
[1 2 3] length pstack pop
3 dict length pstack pop
3 4 length pstack pop
```

pathlength is part of Adobe's Blue book P5 p143, Centered dash patterns. The procedure `pathlength` computes the length of any given path. It does so by first flattening the path with the `flattenpath` operator. `flattenpath` converts any `curveto` and `arc` segments in a path to a series of `lineto` segments. Then the procedure `pathforall` is used to access each segment in the path, find its length and add the length to a total. A PostScript programming paradigm.

12. Courtesy Taco Hoekwater

Mean of two points The problem is to calculate .5[p_1, p_2]. Too trivial?

I found it worthwhile to communicate, because the operator makes use of the stack only. Stack-oriented PostScript, different from the PostScript I used in my `Just a little bit of PostScript`, of old.

```
/mean%p0 p1 on stack ==> .5[p0, p1] i.e. the coordinates of the mean
 {exch  4 -1 roll add .5 mul 3  1 roll add .5 mul}def
```

More general is the calculation of a point on the line between the two points. The `mediation` operator: t[p_1, p_2], t$\in [0,1]$.

```
/mediation {% a b t ==> c. c is weighted average of a and b;
            % c = a * (1-t) + b * t
dup 1 exch sub 4 -1 roll mul
3 1 roll mul add
} bind def
```

Rotation of a point Despite PostScript's functionality to rotate user space, I needed an operator to rotate just points. I chose conform the PostScript tradition that a positive angle rotates counterclockwise.

$x\ y$ angle `rot` $x'\ y'$, rotates the point $x\ y$ over the angle `angle` in degrees, counterclockwise in the PostScript tradition

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

```
/rotdict 6 dict def
/rot %x y phi: point and angle of rotation (counterclockwise)
     %         ==> x y coordinates of point after rotation
{rotdict begin
 /phi exch def /y exch def /x exch def
 /xaux x phi cos mul y phi sin mul sub def
 /y     x phi sin mul  y phi cos mul add def
 /x xaux def
 x y
 end
 } def
```

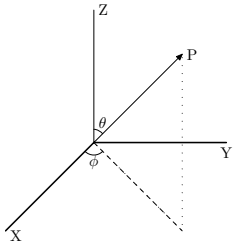
Projection of a point, operator ptp. For projections I specify the graphics in 3D and project the data onto the plane by the operator `ptp` with viewing angles as parameters.[13]

□ num_1 num_2 num_3 ptp num_1 num_2, projects the 3D point on the plane with viewing angles ϕ and θ

The projection formula, with ϕ and θ viewing angles, reads

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

```
/ptp{% point x y z ==> x' y'
      % use: /pair { x y z ptp } def
      % parameters: a, b viewing angles
   0 begin
   /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
```



13. The projection I also coded in MetaPost.

```
   z b cos mul add
   end}def
/ptp load 0 3 dict put
```

Indispensible. A practical variant with **f**ixed viewing angles, `ptpf`, is part of the `PSlib.eps` library too.

dotsandnames I consider it convenient to mark points in a picture with their names. I decided to supply the names of the points as a string in an array. Each name, a1 for example, is a definition which contains the coordinates of the point a1.

```
/dotsandnames{%[ str,..., str]==>
%str is a name, which stands for a pair, such as in pair moveto
{dup cvn load exec moveto (.) H15pt setfont centershow
                              H10pt setfont show}forall
}def
```

A nice use of `forall` and of using the contents of the name and the name itself. `dotandnames` is used in the emulations of Gabo's constructions. A PostScript programming paradigm.

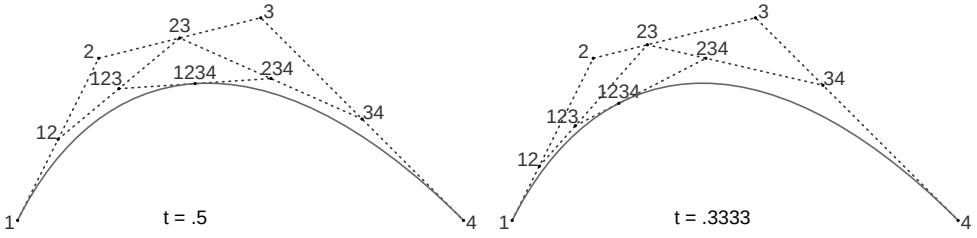
tonSpline yields the coordinates of a point on a spline given by the points (a0,a1,a2,a3)[14] as function of the time parameter $t \in [0,1]$: $z(t) = \sum_{k=0}^{3}(1-t)^{3-k}t^k a_k$. Algorithmically,[15]

$$\begin{aligned} z(t) =&(1-t)\Big((1-t)\Big((1-t)a_0 + t\,a_1\Big) + t\,\Big((1-t)a_1 + t\,a_2\Big)\Big) \\ &+ t\Big((1-t)\Big((1-t)a_1 + t\,a_2\Big) + t\,\Big((1-t)a_2 + t\,a_3\Big)\Big) \end{aligned}$$

The (vector) value $z(t) = a_{0123}$, of the B-cubic characterized by a_0, a_1, a_2, a_3 for the value of the (time) variable t can algorithmically also be described as

$$\begin{matrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{matrix} \rightarrow \begin{matrix} a_{01} = a_0\,(1-t) + a_1\,t \\ a_{12} = a_1\,(1-t) + a_2\,t \\ a_{23} = a_2\,(1-t) + a_3\,t \end{matrix} \rightarrow \begin{matrix} a_{012} = a_{01}\,(1-t) + a_{12}\,t \\ a_{123} = a_{12}\,(1-t) + a_{23}\,t \end{matrix} \rightarrow a_{0123} = a_{012}\,(1-t) + a_{123}\,t$$

Graphically, Knuth displayed in the Metafont book p13 the determination of a point on a cubic spline.



The PostScript operator in `PSlib.eps` has been written by Piotr Strzelczyk.

Intersection of 2 straight lines. This functionality was needed when I imitated GUST's batteship logo in PostScript. Under the hood 2 linear equations in 2 unknowns have to be solved. The operator is called `intersect`.

14. It is interesting to see the control points back as coefficients in the Bernsteĭn polynomial representation.
15. Named after de Casteljau.

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

The library contains also an operator for solving 3 linear equations in 3 unknowns.

Fonts for free Don Lancaster in his PSsecrets gives many font variations.

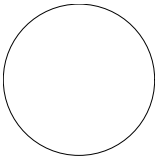
shadow

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

BoundingBox calculation by PostScript.
PostScript provides pathbbox for calculation of the size of the surrounding rectangle of a path. Together with setpagedevice this may be used to crop a picture on the fly.


```
%!PS-Adobe-3.0
%%Title: One-pass cropping, LRM 2
/one-passcroppingdict 6 dict def
/one-passcropping{one-passcroppingdict begin %example
/Times-Roman 30 selectfont
0 0 moveto (StarLines) false charpath
                flattenpath pathbbox
/ury exch def /urx exch def /lly exch def /llx exch def
/w urx llx sub cvi def /h ury lly sub cvi def
<</PageSize [w h]>>setpagedevice
newpath
/rays{120{0 0 moveto 108 0 lineto 1.5 rotate
     }repeat stroke}def
0 1  moveto (StarLines) true charpath clip
newpath 50 -15 translate rays showpage
end}def
```

For a picture PostScript prompted me values, which I inserted in the BoundingBox structure command.



```
%!PS-Adobe-3.0
%%Title: One-pass cropping, LRM 2.3
%%BoundingBox: -50 -50 150 150
%%BeginSetup
%%EndSetup
(c:\\PSlib\\PSlib.eps) run
/one-passcroppingdict 6 dict def
/one-passcropping{one-passcroppingdict begin %example
50 50 100 0 360 arc flattenpath pathbbox
```

```
/ury exch def /urx exch def /lly exch def /llx exch def
ury showobject urx showobject lly showobject llx showobject stroke
showpage end}def
one-passcropping
%%EOF
```

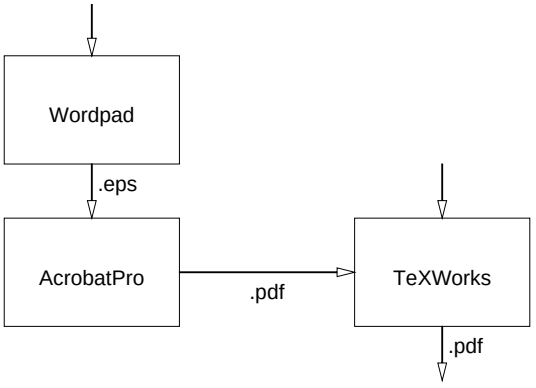
In practice I do not use this facility. I estimate the BoundingBox of a picture, reasonably. In the example it is a trifle.

2D pictures

Merging .eps pictures in .tex documents: flowchartTeXandPS

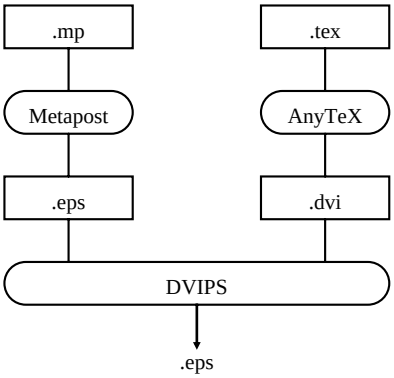
.eps pictures as such can't be included when processing with the TEX-engine pdfTEX, which I use in TEXworks. They first have to be converted into .pdf. They are more efficiently processed by TEXWorks when a priori converted into .eps.

MetaPost users prefer Hobby's boxes package for structure diagrams or flowcharts, I presume. I consider the use of straight PostScript equally simple or equally difficult, depending on your expertise.




```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -51 -63 252 153
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/flowchartTeXandPSdict 15 dict def
flowchartTeXandPSdict
begin%local auxiliaries
/s 50 def /t s .61803 mul def
/2s s s add def /2t t t add def
/3s 2s s add def /3t 2t t add def
/4s 2s 2s add def
/-s s neg def /-t t neg def
/-2s 2s neg def /-2t 2t neg def
/-3s -2s -s add def /-3t -2t -t add def
/Courier 12 selectfont
end
/flowchartTeXandPS%uses: arrow, centershow from PSlib.eps
{flowchartTeXandPSdict begin
gsave
-s -t 2s 2t rectstroke 0 -7 moveto (AcrobatPro) centershow %Acrobat Pro 7 box
0 2t 0 t  .5 5 10 arrow stroke                             %.eps downarrow
0 1.5 t mul moveto ( .eps) show                            %text right of downarrow
0 3t translate
0 2t 0 t .5 5 10 arrow stroke
-s -t 2s 2t rectstroke 0 -7 moveto (Wordpad) centershow    %WordPad box
grestore 4s 0 translate
-s -t 2s 2t rectstroke 0 -7 moveto (TeXWorks) centershow   %TeXWorks box
-3s 0 -s 0 .5 5 10 arrow stroke                            %graphics inclusion arrow
-2s -15 moveto ( .pdf) centershow                          %subcript below arrow
-3s 0 -s 0 .5 5 10 arrow stroke                            %graphics inclusion arrow
0 -t 0 -2t .5 5 10 arrow stroke 0 -1.5 t mul moveto ( .pdf) show %result down arrow
end} def
%%End Prolog
flowchartTeXandPS        showpage                           %invoke
%%EOF
```

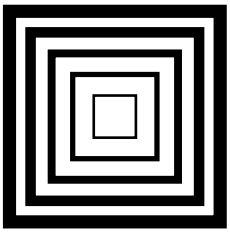
Compared with the flowchart as given in *Just a lttle bit of PostScript*[16] the inclusion of pictures produced by PostScript in TEX documents has become simpler, thanks to pdfTEX and TEXworks. For the flowchart `diamondstroke` and `ovalstroke` have been written in analogy with PostScript's `rectstroke`, and of course they have been added to `PSlib.eps`. One can easily imagine packages where the elements can be dragged and dropped on a grid to build flowcharts. For simple flowcharts the PostScript operators will do.

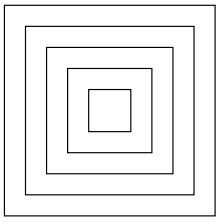


16. GKP picture lost alas after 20 years, so I programmed it anew in PostScript.

Nest of squares. Blue book, P2 p131

A triviality? Not so in PostScript, because the scaling of user space also scales the `linewidth`.[17] The example shows how to program à la PostScript with invariant `linewidth`. Below, I also included the old procedural way of programming a nest of squares, which circumvents the problem.

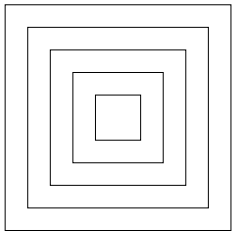




```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nest of squares Adobe
%%BoundingBox: 80 335 525 529
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nestofsquares% used from library: centersquare
{gsave 2.5 inch 6 inch translate 1 16 div setlinewidth
  1 1 5{ gsave  .5 mul inch dup scale
           centersquare stroke
         grestore} for
grestore
gsave
  6 inch 6 inch translate 1 setlinewidth
  /cmtx matrix currentmatrix def%store current matrix
  1 1 5{ gsave .5 mul inch dup scale
           centersquare
           cmtx setmatrix %reuse stored matrix
           stroke%invariant linewidth
         grestore} for
grestore}def
%%EndProlog
nestofsquares
showpage
%%EOF
```

Interesting use has been made of `currentmatrix` and `setmatrix`. A PostScript programming paradigm.

Variant nest of squares



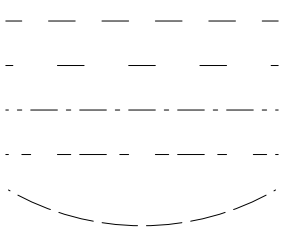
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nest of squares, variant
%%BoundingBox: -181 -181 181 181
%%BeginSetup
%%EndSetup
1 1 5{ 36 mul /s exch def /-s s neg def /2s s s add def
          -s -s 2s 2s rectstroke
       } for
showpage
%%EOF
```

Straight forward procedural approach, with `linewidth` invariant.

17. Sometimes very useful, for example in Fractal Arrow, Pythagoras trees, or the Lévy fractal... .

Centered Dash Patterns. Blue book P5 p143

In illustrations dashed lines occur frequently. Adobe provides nice variations.



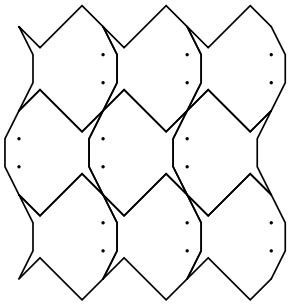
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Centered Dash Patterns. Blue book P5 p143
%%BoundingBox: 71 255 379 510
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/centerdashdict 9 dict def
/centerdash%uses pathlength from the library
  {centerdashdict begin /pattern exch def
    /pathlen pathlength def
    /patternlength 0 def
    pattern { patternlength add /patternlength exch def } forall
    pattern length 2 mod 0 ne
      { /patternlength patternlength 2 mul def } if
    /first pattern 0 get def
    /last patternlength first sub def
    /n pathlen last sub cvi patternlength idiv def
    /endpart pathlen patternlength n mul sub
       last sub 2 div def
    /offset first endpart sub def
    pattern offset setdash
  end } def
%%EndProlog
newpath 72 500 moveto 378 500 lineto   [30] centerdash stroke
newpath 72 450 moveto 378 450 lineto   [30 50] centerdash stroke
newpath 72 400 moveto 378 400 lineto   [30 10 5 10] centerdash stroke
newpath 72 350 moveto 378 350 lineto   [30 15 10] centerdash stroke
newpath 225 570 300 240 300 arc        [40 10] centerdash stroke
showpage
%%EOF
```

Interesting is the use of `flattenpath` in `pathlength` for calculating the length of a path, be it a Bézier cubic (or a circulare arc), or a line, respectively an arbitrary combination.

```
/pathlengthdict 7 dict def
/pathlength{pathlengthdict begin%Bluebook P5p143
   flattenpath /dist 0 def
      { /yfirst exch def /xfirst exch def
        /ymoveto yfirst def /xmoveto xfirst def }
      { /ynext exch def /xnext exch def
        /dist dist ynext yfirst sub dup mul
          xnext xfirst sub dup mul add sqrt add def
        /yfirst ynext def /xfirst xnext def }
      {}%path has been flattened, so no curveto or arc anymore
      { /ynext ymoveto def /xnext xmoveto def
        /dist dist ynext yfirst sub dup mul
          xnext xfirst sub dup mul add sqrt add def
        /yfirst ynext def /xfirst xnext def }
      pathforall
      dist
end} def
```

Escher fishes

The example has been included because a little deformed 3x3 tile can yield creative results.



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escherfishes, cgl 1997
%%BoundingBox: -85 -90 85 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/escherfishesdict 12 dict def
/escherfishes{escherfishesdict begin %tilLxxii
0 setlinejoin 1 setlinecap
/a 50 def          /ma a neg def
/ha .5  a mul def  /mha ha neg def
/qa .5 ha mul def  /mqa qa neg def
/d .3333 a mul def /dd .5 d mul def
/Courier findfont 25 scalefont setfont
/tile{gsave
  2{mha ha moveto
  mha dd add  ha d sub lineto
  mha dd add mha d add lineto
  mha mha lineto
  a 0 translate}repeat stroke
  grestore
  gsave
  2{ha ha moveto
    qa ha qa add lineto
    mqa qa lineto
    mha ha lineto stroke
    gsave ha  ha d sub translate
    0 0 moveto 0 0 1 0 360 arc fill
    grestore
    1 -1 scale}repeat
  grestore
}def
ma a a{/i exch def
ma a a{/j exch def
   gsave j i translate
         tile grestore
}for -1 1 scale}for
end}def
%%EndProlog
escherfishes showpage
%%EOF
```

Escher Alhambra tile

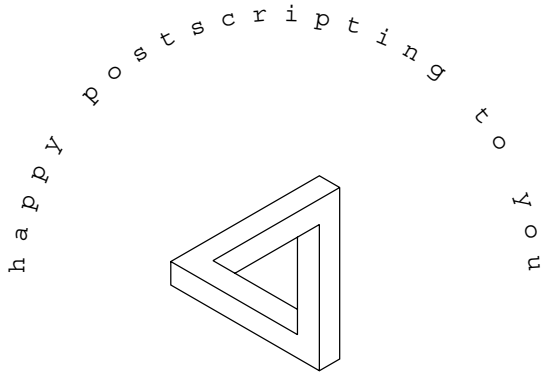
My old Metafont example has been included because it is a nice tile, and it has been processed by Troy Henderson's remote MP Previewer.

```
beginfig(0) size:=3; path p[]; picture pic[];
draw (-10size,5size)--(0,5size)--(0,-5size)--(10size,-5size);
draw (5size,10size)--(5size,0)--(-5size,0)--(-5size,-10size);
pic1:=currentpicture;
clearit;
%flower
p1= (origin--(3size,0)) shifted (size,0) rotated 22.5;
p2= point 1of p1{point 1 of p1 -origin}..{down}(5size,0);
p3= p2 reflectedabout(point 1 of p1, origin);
draw p1..p2;draw p3;
p4= (p1..p2)rotated 45;
p5= p3 rotated 45;
draw p4; draw p5;
draw (-5size,10size)--(-2.5size,10size)--(10size,-2.5size)--(10size,-10size);
addto currentpicture also currentpicture rotated90;
addto currentpicture also currentpicture rotated180;
draw fullcircle scaled 2size;
draw unitsquare scaled 10.5size shifted (-5.25size, -5.25size);
pic2:=currentpicture;
%
addto currentpicture also pic1 shifted (20size,0);
addto currentpicture also pic1 shifted (0,20size);
addto currentpicture also pic2 shifted (20size,20size);
addto currentpicture also currentpicture shifted (40size,0);
addto currentpicture also currentpicture shifted (0,40size);
pickup pencircle scaled 5;
draw unitsquare scaled 80size shifted (-10size,-10size);
endfig
end
```

Seal

A nice use of kshow.[18] Interesting is the Escher triangle. Once the rotation symmetry has been discovered the programming becomes a trifle. The text along an implicit path, via kshow, is also nice and a PostScript programming paradigm.

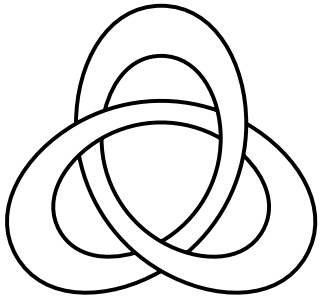
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Seal
%%BoundingBox: -110 -60 110 110
%%Beginsetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/sealdict 3 dict def
/seal{sealdict begin
/Courier 10 selectfont /r 100 def
/text (happy postscripting to you) def
gsave 89.9 rotate
0 r moveto
{-7.04 rotate 0 r moveto} text kshow
grestore
.75 dup scale eschertriangle
end}def
seal showpage
%%EOF
```



18. What puzzles me is that rotation over 89.9 and 90 differ so much in appearance???

Escher doughnut

The doughnut was programmed in declarative Metafont. On occasion of EuroTEX2012 the code was processed by MetaPost and a PostScript version emerged by editing the PostScript code. How to program this doughnut from scratch in PostScript is not clear to me. What makes it hard are the hidden lines.

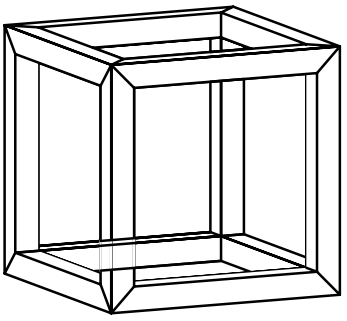


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escher doughnut
%%BoundingBox: -40 -32 40 45
%%Creator: MetaPost and JJW, CGL. Doughnut
%%BeginSetup
%%EndSetUp
3{-21.6507 12.5 moveto
-21.6507 27.74551 -13.78212 42.5003
    0 42.5003 curveto
13.78212 42.5003 21.6507 27.74551
   21.6507 12.5 curveto
21.6507 -0.24506 16.04897 -12.19249
   6.58395 -20.3313 curveto
%
-14.3152 15.86746 moveto
-12.43301 23.58928 -7.45113 29.75021
   0 29.75021 curveto
9.64748 29.75021 15.15549 19.42186
   15.15549 8.75 curveto
15.15549 -2.07904 9.37823 -12.08548
   0 -17.5 curveto
120 rotate
}repeat
stroke showpage
%%EOF
```

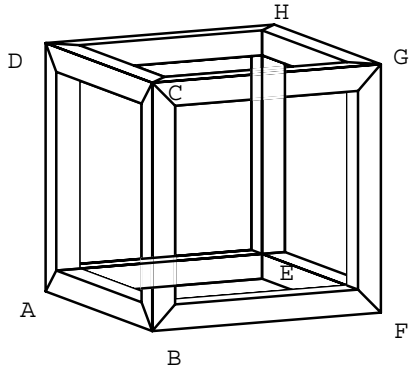
Postprocessed by Photoshop.

Escher's impossible cubes

The example has been included because I have not seen Escher impossible cubes programmed.



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Imp Cube in projection. cgl 2009
%%BoundingBox: -20 -30 150 120
%%Beginsetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
/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
/a 25 def /b 10 def%projection parameters
/r 100 def /mr r neg def /hr r 2 div def%size
/d 10 def  /md d neg def  /rmd r d sub def  /dmr rmd neg def  %d<<r
1 setlinewidth 1 setlinejoin /Courier 10 selectfont
/a1 {0 0 0 ptp} def
/a2 {0 d 0 ptp} def
/a3 {0 d d ptp} def
/a4 {0 0 d ptp} def
/a5 {md 0 0 ptp} def
/a6 {md d 0 ptp} def
/a7 {md d d ptp} def
/a8 {md 0 d ptp} def
%a1 moveto -10 -10 rmoveto (A) show
/b1 {0  rmd  0 ptp} def
/b2 {0  r    0 ptp} def
/b3 {0  r    d ptp} def
/b4 {0  rmd  d ptp} def
/b5 {md rmd  0 ptp} def
/b6 {md r    0 ptp} def
/b7 {md r    d ptp} def
/b8 {md rmd  d ptp} def
%b1 moveto 10 -15 rmoveto (B) show
/c1 {0  rmd  rmd ptp} def
/c2 {0  r    rmd ptp} def
/c3 {0  r    r   ptp} def
/c4 {0  rmd  r   ptp} def
/c5 {md rmd  rmd ptp} def
/c6 {md r    rmd ptp} def
/c7 {md r    r   ptp} def
/c8 {md rmd  r   ptp} def
%c2 moveto 6 3 rmoveto (C) show
/d1 {0  0 rmd ptp} def
/d2 {0  d rmd ptp} def
/d3 {0  d r   ptp} def
/d4 {0  0 r   ptp} def
/d5 {md 0 rmd ptp} def
/d6 {md d rmd ptp} def
/d7 {md d r   ptp} def
/d8 {md 0 r   ptp} def
```



```
%d1 moveto -15 0 rmoveto (D) show
/e1 {dmr 0 0 ptp} def
/e2 {dmr d 0 ptp} def
/e3 {dmr d d ptp} def
/e4 {dmr 0 d ptp} def
/e5 {mr 0 0 ptp} def
/e6 {mr d 0 ptp} def
/e7 {mr d d ptp} def
/e8 {mr 0 d ptp} def
%e5 moveto 2 -3 rmoveto (E) show
/f1 {dmr rmd  0 ptp} def
/f2 {dmr r    0 ptp} def
/f3 {dmr r    d ptp} def
/f4 {dmr rmd  d ptp} def
/f5 {mr  rmd  0 ptp} def
/f6 {mr  r    0 ptp} def
/f7 {mr  r    d ptp} def
/f8 {mr  rmd  d ptp} def
%f6 moveto 5 -10 rmoveto (F) show
/g1 {dmr rmd  rmd ptp} def
/g2 {dmr r    rmd ptp} def
/g3 {dmr r    r   ptp} def
/g4 {dmr rmd  r   ptp} def
/g5 {mr  rmd  rmd ptp} def
/g6 {mr  r    rmd ptp} def
/g7 {mr  r    r   ptp} def
/g8 {mr  rmd  r   ptp} def
%g7 moveto 5  0 rmoveto (G) show
/h1 {dmr 0 rmd ptp} def
/h2 {dmr d rmd ptp} def
/h3 {dmr d r   ptp} def
/h4 {dmr 0 r   ptp} def
/h5 {mr 0 rmd ptp} def
/h6 {mr d rmd ptp} def
/h7 {mr d r   ptp} def
/h8 {mr 0 r   ptp} def
%h8 moveto 0 2 rmoveto (H) show
%AEHD
/AEHDo {a1 moveto e5 lineto h8 lineto d4 lineto closepath}def
/AEHDoi{a8 moveto e4 lineto h1 lineto d5 lineto closepath}def
/AEHDii{a7 moveto e3 lineto h2 lineto d6 lineto closepath}def
%EFGH
/EFGHo {e5 moveto f6 lineto g7 lineto h8 lineto closepath}def
/EFGHoi{e7 moveto f8 lineto g1 lineto h6 lineto closepath}def
/EFGHii{e3 moveto f4 lineto g1 lineto h2 lineto closepath}def
%ABCD
/ABCDo{a1 moveto b2 lineto c3 lineto d4 lineto closepath}def
/ABCDoi{a3 moveto b4 lineto c1 lineto d2 lineto closepath}def
%BFGC
/BFGCo{b2 moveto f6 lineto g7 lineto c3 lineto closepath}def
/BFGCoi{b7 moveto f3 lineto g2 lineto c6 lineto closepath}def
%CGHD
/CGHDii{c3 moveto g7 lineto h8 lineto d4 lineto closepath}def
/CGHDoi{c8 moveto g4 lineto h3 lineto d7 lineto closepath}def
/CGHDo {c3 moveto g7 lineto h8 lineto d4 lineto closepath}def
```

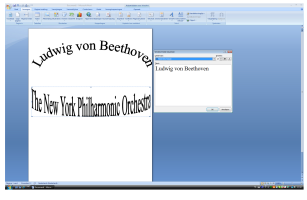
```
%ABFE
/ABFE0i{a6 moveto b5 lineto f1 lineto e2 lineto closepath}def
/ABFEii{a7 moveto b8 lineto f4 lineto e3 lineto closepath}def
%Snijpunten ivm clipping window
/uc1{h2 d6 d7 c8 intersect}def
/uc2{h2 g1 c8 g4 intersect}def
/uc3{c6 g2 g1 f4 intersect}def
/uc4{f4 b8 b7 c6 intersect}def
/uc5{d6 a7 d2 c1 intersect}def
/uc6{a7 b8 b4 c1 intersect}def
%3 windows as clipping path
/windows{newpath %
a7 moveto uc6 lineto c1 lineto uc5 lineto closepath %ABCDoi
c6 moveto uc3 lineto f4 lineto uc4 lineto closepath %BFGCoi
c8 moveto uc1 lineto h2 lineto uc2 lineto closepath %CGHDoi
}def
%Draw Front and top
ABCDo ABCDoi BFGCo BFGCoi CGHDo CGHDoi stroke
%upper what is in sight
h8 moveto h3 lineto h2 lineto
d2 moveto d4 lineto d7 lineto
g4 moveto g7 lineto g2 lineto
c1 moveto c3 lineto c8 lineto
c3 moveto c6 lineto
h2 moveto uc1 lineto
h2 moveto uc2 lineto stroke
% einde achterbovenkant in zicht, lower corners
a1 moveto a3 lineto a7 lineto
b4 moveto b2 lineto b7 lineto
e4 moveto e3 lineto e2 lineto
e3 moveto e7 lineto
f4 moveto f3 lineto f6 lineto stroke
```

Adobe's Blue book CD label, part1

The PostScript def has been published in Adobe's Blue book. The `insidecircletext def` demonstrates how to process a string character by character, a PostScript programming paradigm.

Insidecircletext prints the text in a counter-clockwise fashion with its baseline along the circumference, on the inside of a circle.

Circular Text by Photoshop



Circular Text by Word

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Blue book CD Label Program 10 p167
%%BoundingBox: -166 -166 166 166
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\Bluebook.eps) run
/circtextdict 19 dict def
circtextdict begin /pi 3.1415923 def
/findhalfangle{ stringwidth pop 2 div
         2 xradius mul pi mul div 360 mul} def
/outsideplacechar{ /char exch def
       /halfangle char findhalfangle def
       gsave halfangle neg rotate
         radius 0 translate
         -90 rotate
         char stringwidth pop 2 div neg 0 moveto
         char show
       grestore
       halfangle 2 mul neg rotate} def
/insideplacechar{ /char exch def
       /halfangle char findhalfangle def
       gsave  halfangle rotate
         radius 0 translate 90 rotate
         char stringwidth pop 2 div neg 0 moveto
         char show
       grestore
       halfangle 2 mul rotate} def
end%circletextdict

/insidecircletext{%parameters: text pointsize centerangle radius
circtextdict begin  /radius exch def/centerangle exch def
          /ptsize exch def/str exch def
         /xradius radius ptsize 3 div sub def
         gsave centerangle str findhalfangle sub rotate
           str{ /charcode exch def
                  ( ) dup 0 charcode put insideplacechar
              } forall
         grestore
end} def
%%EndProlog
/Times-Roman 15 selectfont (The New York Philharmonic Orchestra) 15 270 118
insidecircletext showpage
%%EOF
```

Adobe's Blue book CD label, part2

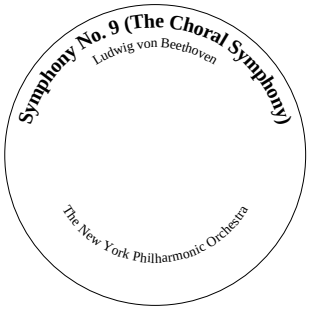
The PostScript def has been published in Adobe's Blue book. The `outsidecircletext` def demonstrates how to process a string character by character, a PostScript programming paradigm.

Outsidecircletext prints the text in a clockwise fashion with its baseline along the circumference, on the outside of a circle.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Blue  book, Program 10 p167
%%BoundingBox: -166 -166 166 166
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\Bluebook.eps) run
/circtextdict 19 dict def
circtextdict begin/pi 3.1415923 def
/findhalfangle{ stringwidth pop 2 div
         2 xradius mul pi mul div 360 mul} def
/outsideplacechar{ /char exch def
       /halfangle char findhalfangle def
       gsave halfangle neg rotate
         radius 0 translate -90 rotate
         char stringwidth pop 2 div neg 0 moveto
         char show
       grestore
       halfangle 2 mul neg rotate} def
/insideplacechar
{ /char exch def
       /halfangle char findhalfangle def
       gsave halfangle rotate
         radius 0 translate
         90 rotate
         char stringwidth pop 2 div neg 0 moveto
         char show
       grestore
       halfangle 2 mul rotate} def
end%circtextdict

/outsidecircletext%parameters: text pointsize centerangle radius
 {circtextdict begin
   /radius exch def /centerangle exch def /ptsize exch def /str exch def
         /xradius radius ptsize 4 div add def
         gsave centerangle str findhalfangle add rotate
           str{ /charcode exch def
                  ( ) dup 0 charcode put outsideplacechar
              } forall
         grestore
       end} def
%%EndProlog
0 0 165 0 360 arc stroke
/Times-Bold 22 selectfont (Symphony No. 9 (The Choral Symphony)) 22 90 140
outsidecircletext showpage
%%EOF
```

Adobe's Blue book CD label, complete

The PostScript `def` has been published in Adobe's Blue book.




```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Blue Book CD Label Program 10 p167
%%BoundingBox: -166 -166 166 166
%%BeginProlog
(C:\\PSlib\\Bluebook.eps) run
/circtextdict 19 dict def
circtextdict begin /pi 3.1415923 def
/findhalfangle{ stringwidth pop 2 div
         2 xradius mul pi mul div 360 mul} def
/outsideplacechar{ /char exch def
       /halfangle char findhalfangle def
       gsave halfangle neg rotate
         radius 0 translate -90 rotate
         char stringwidth pop 2 div neg 0 moveto
         char show
       grestore halfangle 2 mul neg rotate} def
/insideplacechar{ /char exch def
       /halfangle char findhalfangle def
       gsave  halfangle rotate
         radius 0 translate 90 rotate
         char stringwidth pop 2 div neg 0 moveto
         char show
       grestore halfangle 2 mul rotate} def
end%circtextdict
/outsidecircletext{ circtextdict begin
        /radius exch def /centerangle exch def /ptsize exch def /str exch def
         /xradius radius ptsize 4 div add def
         gsave centerangle str findhalfangle add rotate
           str{ /charcode exch def
                  ( ) dup 0 charcode put outsideplacechar
                } forall
         grestore end} def
/insidecircletext{ circtextdict begin
        /radius exch def /centerangle exch def /ptsize exch def /str exch def
         /xradius radius ptsize 3 div sub def
         gsave centerangle str findhalfangle sub rotate
           str{ /charcode exch def
                  ( ) dup 0 charcode put insideplacechar
                } forall
         grestore end} def
 %%EndProlog
/cddvdlabel{0 0 165 0 360 arc stroke
/Times-Bold 22 selectfont (Symphony No. 9 (The Choral Symphony)) 22 90 140
outsidecircletext
/Times-Roman 15 selectfont
   (Ludwig von Beethoven) 15 90 118 outsidecircletext
   (The New York Philharmonic Orchestra) 15 270 118 insidecircletext
}def
 /EndProlog
cddvdlabel
%%EOF
```

Adobe's Blue book CD label, enriched by background notes

The PostScript def has been published in *CD and DVD labels.* A simplified version has been incorporated. `partituur9e.eps` had to be placed in the c-directory!?!


```
%!PS-Adobe-3.0
%%Title: Blue Book label with score illustration added
%%Creator: Kees van der Laan, kisa1@xs4all.nl, 2011
%%BoundingBox: -175 -175 175 175
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/toplabel{/mm {72 25.4 div mul} def
0 0 60 mm 0 360 arc
20 mm 0 moveto
0 0 20 mm 0 360 arc
0 -20 mm moveto 0 20 mm lineto
-20 mm 0 moveto 20 mm 0 lineto
}def%toplabel
%
%---Program--- the script
%
3 setlinewidth toplabel gsave stroke grestore eoclip
gsave
-170 -170 translate
(c:\\partituur9e.eps) run %put the partituur clipped to the label
grestore
%annotations as prompted by Program 10 of Adobe's Blue Book
1 setgray     %annotations in white
/size 27 def
/Times-Roman size selectfont
(Symphony No. 9 (The Choral Symphony)) size  90 135 outsidecircletext
/size 20 def
/Times-Roman size selectfont
(Ludwig von Beethoven)                 size  90 118 outsidecircletext
(The New York Philharmonic Orchestra)  size 270 118 insidecircletext
showpage
%%EOF
```

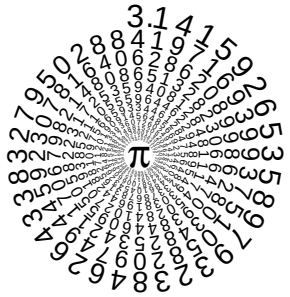
In MS Word and Nero (Version 10 with LightScribe) one can create labels interactively. LightScribe, a HP technique to burn labels on special discs as well, allows only for black and white labels.

My wife, not at all a TEXie, designs labels in Photoshop and prints them on prefab glued paper by the tool CDFACE1.6 (Media labeling software templates for: CDs, DVDs, jewel cases, envelopes, floppy discs, audio cassettes, dat tapes, zip discs). The special tool CDFACE can handle all, no PhotoShop is needed.

Willi Egger has shown, MAPS 2009, how to use ConTEXt for the purpose.

Pi decimals

Released as BachoTEX2012 Programming Pearl. The spiral path is implicit!

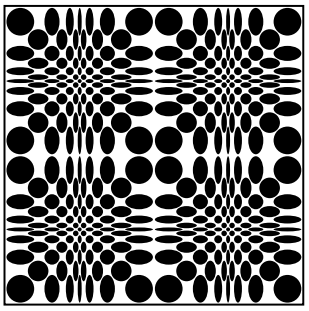



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pi-decimals along a  Spiral cgl 2010, 2012
%%BoundingBox: -80 -100 100 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%Endprolog
/Symbol 26 selectfont
1 -18 moveto (p) show%p denotes pi in the symbol font
/Helvetica 20 selectfont
0 70 moveto (3) show 1 0 rmoveto (.) show -2 0 rmoveto
-10 rotate .995 dup scale
%
{pop pop -10 rotate 3 0 rmoveto .995 dup scale}
(3.14159265358979323846264338327950288419716939937 5\
10582097494459230781640628620899862803482534211 70\
67982148086513282306647093844609550582231725359 40\
81284811174502841027019385211055596446229489549 30\
38196442881097566593344612847564823378678316527 12\
01909145648566923460348610454326648213393607260 24\
91412737245870066063155881748815209209628292540 91\
71536436789259036001133053054882046652138414695 19\
41511609433057270365759591953092186117381932611 79...) kshow
showpage
%%EOF
```

Vo in his *PSTricks — Graphics and PostScript for TEX and LATEX, p294* shows a variant π-decimals in `PSTricks` in LATEX.

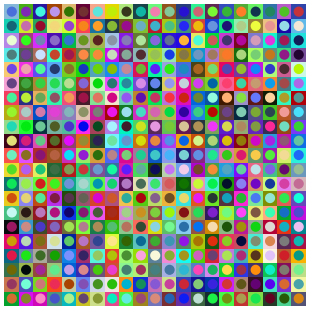
The original picture from the CWI calendar of 1972

Schrofer's Op Art



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Schrofer's OpArt
%%BoundingBox: -195 -195 570 570
%%BeginSetup
%%endSetup
%%BeginProlog
/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 1 moveto
   0 0 1 0 360 arc
   fill
grestore}def
/schrofer{/flipflop true def
/indices [30 21 14 9 5 2 0 -2 -5 -9 -14 -21 -30] def
indices{/r exch s mul def
   gsave
   indices{/c exch s mul def
      flipflop{drawgc}if
      /flipflop flipflop not def}forall
   grestore
   }forall
closepath 5 setlinewidth stroke
}def
%%EndProlog
gsave
2{gsave
  2{schrofer
    375 0 translate}repeat
  grestore
  0 375 translate
 }repeat
grestore
5 setlinewidth -190 dup 755 dup rectstroke%border
%%EOF
```

Vasarely's Op Art: random coloured squares with circles



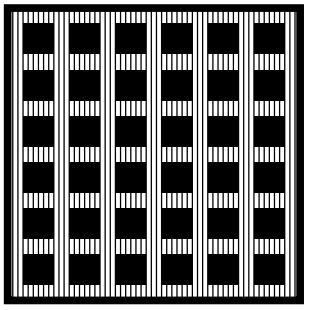
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Vasarely random coloured Squares. CGL 2007
%%BoundingBox: -211 -211 211 211
%%BeginProlog
/vasarelyrandomdict 6 dict def
vasarelyrandomdict begin
/r 20 def             %side square
/R r 10 mul def       %bound loops
/r2 r 2 div def       %r/2
/ri r2 2 sqrt div def %radius inner circle
/s 2 31 exp 1 sub def %scaling random  numbers, reus
/square
{rand s div rand s div rand s div setrgbcolor
 r2 neg dup r dup rectfill
 rand s div rand s div rand s div setrgbcolor
 0 0 ri 0 360 arc fill} def
end%dict
/vasarelyrandom{vasarelyrandomdict begin 1951 srand
R neg r R{/i exch def
R neg r R{/j exch def
          gsave i j translate square grestore
         }for
         }for
end}def
%%EndProlog
vasarelyrandom showpage
%%EOF
```

In Metafont the following Vasarely's I programmed 20 years ago. Interesting is the use of `interpath`, similar as used in the heart figure in the Metafont book p134. `interpath` is not available in PostScript, because we don't have explicit paths. Below I have imitated `interpath` functionality in PostScript for this simple example. No path data structure nor picture datastructure. Just transformation of User Space. Compare the code with the MetaPost code given in Recreational use of TEX&Co. Simpler to read isn't it? The code of the right picture resembles much the code of Schrofer and is supplied in `PSlib.eps` under the name `vasarelybloks`.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Vasarely lines. CGL 2014
%%BoundingBox: -221 -211 221 211
/pattern{newpath
/h 210 def /k h 3 div def /d 30 def
0 10 200{/x exch def
  x d add 0 moveto x d add k  x h k sub %control points
  x h curveto /d d 1 sub def
  }for
stroke}def
2{2{pattern 1 -1 scale pattern -1 1 scale
   }repeat 90 rotate}repeat
2 setlinewidth -210 -210 420 420 rectstroke showpage
%%EOF
```

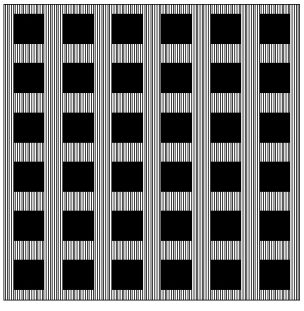
Soto's Op Art

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Soto, cgl 2012
%%BoundingBox: -1 -1 58 58
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/soto{%cgl 2012
gsave .25 setlinewidth
57{1 0 translate 0 0 moveto 0 57 lineto}repeat stroke
grestore
gsave 1.5 setlinewidth 0 0 57 57 rectstroke grestore
3 3 translate
6{gsave
  6{0 0 6 6 rectfill 9 0 translate}repeat
  grestore
  0 9 translate
 }repeat
}def
%%EndProlog
soto showpage
%%EOF
```



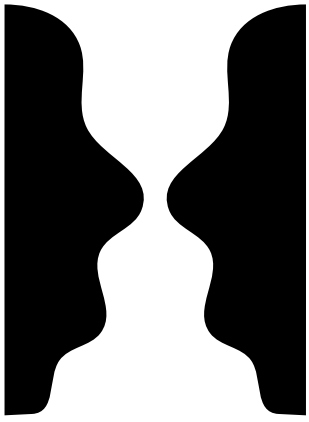
Earlier in TEX alone

```
%cgl feb 2010 tst Soto
\def\boxit#1{\vbox{\hrule\hbox{\vrule#1\vrule}\hrule}}
\newbox\cb  \newdimen\ul \ul=6ex \newdimen\size \size12\ul
\setbox\cb\vbox to2\ul{\vss%colored box with transparent boundary
   \hbox to2\ul{\hss\vrule height1.2\ul width1.2\ul \hss}%
        \vss}%

$$\boxit{\vbox{\offinterlineskip
\hbox{\xleaders\hbox to.5ex{\hss\vrule height\size\hss}\hskip\size}%
\kern-\size%setback
   \leaders\hbox{\leaders\copy\cb\hskip\size}\vskip\size}}$$
\bye
```



Candle or faces

PostScript lacks to draw a pleasing curve through given points. The controle points must be specified.



```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -234 -317 234 317
%%BeginSetup
%%EndSetup
newpath -231 -315 moveto
-230.996576 -315 -189.003419 -312.900171 -189 -312.9 curveto
-150.602108 -310.979948 -165.354069 -257.301583 -147 -231 curveto
-132.060075 -209.590923 -99.648395 -209.766718 -84 -189 curveto
-55.898607 -151.707122 -103.284993 -102.741264 -84 -63 curveto
-70.625453 -35.4386 -31.836956 -29.048744 -21 0 curveto
0.145155 56.680143 -81.874447 78.741461 -105 126 curveto
-124.104631 165.04153 -100.718717 211.542235 -115.5 252 curveto
-131.812491 296.648828 -181.870276 315 -231 315 curveto
-231.102539 315 -231.102539 -315 -231 -315 curveto closepath fill
newpath 231 -315 moveto
230.996576 -315 189.003419 -312.900171 189 -312.9 curveto
150.602108 -310.979948 165.354069 -257.301583 147 -231 curveto
132.060075 -209.590923 99.648395 -209.766718 84 -189 curveto
55.898607 -151.707122 103.284993 -102.741264 84 -63 curveto
70.625453 -35.4386 31.836956 -29.048744 21 0 curveto
-0.145155 56.680143 81.874447 78.741461 105 126 curveto
124.104631 165.04153 100.718717 211.542235 115.5 252 curveto
131.812491 296.648828 181.870276 315 231 315 curveto
231.102539 315 231.102539 -315 231 -315 curveto closepath fill
2 setlinewidth -232 -316 454 632 rectstroke
showpage
%%EOF
```

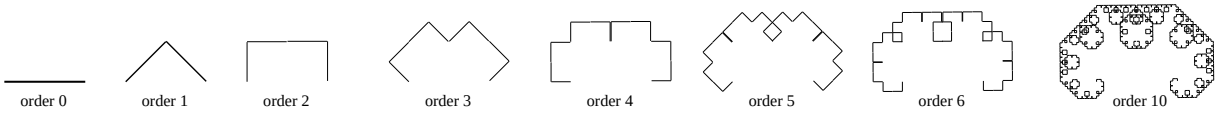
MetaPost code adapted from the old Metafont code.

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

The .eps code was obtained from MetaPost, with the frame added in PostScript.

Lévy fractal

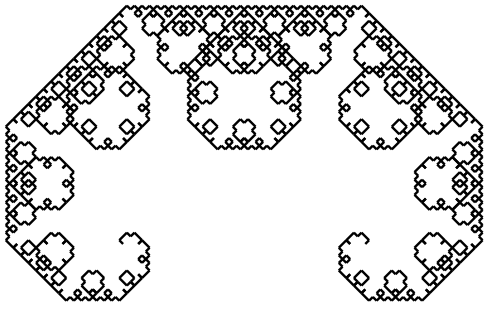
An approximation of the Lévy fractal is also called a C (broken) line of a certain order. The constructive definition of various orders of C lines starts with a straight line, let us call this line C_0. An isosceles triangle with angles 45°, 90° and 45° is built on this line as hypotenuse. The original line is then replaced by the other two sides of this triangle to obtain C_1. Next, the two new lines each form the base for another right-angled isosceles triangle, and are replaced by the other two sides of their respective triangle, to obtain C_2. After two steps, the broken line has taken the appearance of three sides of a rectangle of twice the length of the original line. At each subsequent stage, each segment in the C figure is replaced by the other two sides of a right-angled isosceles triangle built on it. Such a rewriting relates to a Lindenmayer system. After n stages the C line has length $2^{n/2}\times C_0$: 2^n segments each of size $2^{-n/2}\times C_0$.



Production rule

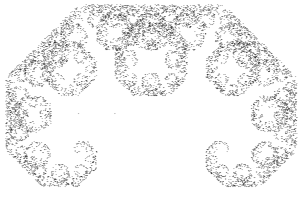
$C_n = [R_{45}S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})}C_{n-1}]\oplus T_{\frac{s}{2}\frac{s}{2}}\,[R_{-45}S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})}C_{n-1}]$, with $C_0 =$ initial line, and C_n the Lévy C curve of order n, $\oplus$ splice operator, meaning add properly, i.e. $T_{(\frac{s}{2},\frac{s}{2})}$, [means store graphics state on the GS stack and open a new one,] means remove current graphics state off the GS stack and recall previous, R_{45} means rotate US 45° in the PS sense $S_{a,b}$ means scale US by a and b, in x- and y-direction $T_{a,b}$ means translate US by a and b, in x- and y-direction.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Levy fractals 0..4. cgl, 2011, kisa1@xs4all.nl
%%BoundingBox: -1 -1 363 65
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/levyC{%on stack: the order ==> C line
       %s = size of line segment (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}%draw line
{1 sub %lower order on stack
  45 rotate levyC%-45 rotate%combine -45 twice into -90
 -90 rotate levyC  45 rotate
 1 add %adjust order on stack
}ifelse
}def
%%EndProlog
/s 20 def             0 levyC pop
s 2 div   0 translate 1 levyC pop
s         0 translate 2 levyC pop
%%EOF
```



Lévy fractal as Iterated Function System

$\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{\text{L}}{=} .5\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}+.5\begin{pmatrix} -1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{\text{R}}{=} .5\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}+.5\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

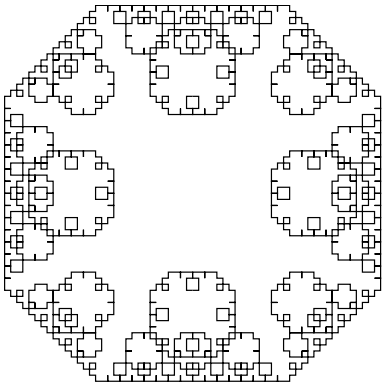


Associated with the Lévy fractal are 2 rotations with rotation centres for L: (-1,0) and for R: (1,0) and contraction $\sqrt{.5} \approx .7$. Amazing, isn't it! Laurier's BASIC program `FRACMC2` and my conversion are given below.

```
REM ***naam: FRACMC2***
KMAX=60000
REM ***Coefficienten***
   A=.5: B=A: C=A : D=-A
   DET1=A*A+B*B : DET2=C*C+D*D
   Q=DET1/(DET1+DET2)
   X=1 : Y=0 : K=0 : KMAX=10000
   DO WHILE K<KMAX AND INKEYS$=""
      R=RND
      IF R<Q THEN
         U=A*X-B*Y-1+A :
           V=B*X+A*Y+B 'rotatie L
      ELSE
         U=C*X-D*Y-1-C :
           V=D*X+C*Y-D 'rotatie R
      ENDIF
      X=U : Y=V
      PSET (X,Y)
   LOOP : BEEP
END
```

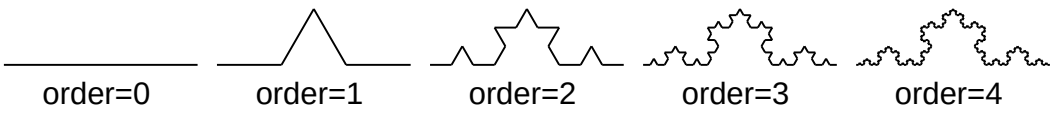
```
%!PS-Adobe-3.0 EPSF-3.0
%%Author: H.A. Lauwerier(1994): Spelen met Graphics en Fractals
%%BoundingBox: -203 -54 205 206
%%BeginSetup
%%EndSetup
%%BeginProlog
/Courier 7 selectfont
/x 0 def  /y 0 def /halfmaxint 2 30 exp  def
/a .5 def  /b a def /c a def /d a neg def
/det1 a dup mul b dup mul add def /det2 c dup mul d dup mul add def
/q det1 det1 det2 add div def
/printxy{x s  y s moveto (.) show}def
%%EndProlog
/s {100 mul}def 10 srand%10 is seed
10000{rand halfmaxint lt
      {/xnew a x mul b y mul sub 1 sub a add def
       /y b x mul a y mul add b add def           /x xnew def%rot L
      }{/xnew c x mul d y mul sub 1 add c sub def
        /y    d x mul c y mul add       d sub def /x xnew def%rot R
      }ifelse
      printxy}repeat
showpage
%%EOF
```

Łévy carpet. C_{10} spliced with its `-1 1 scale`d copy, a Lévy carpet.



von Koch fractal

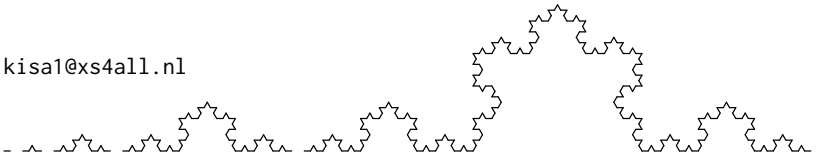
A von Koch broken line and a Lévy C line are related to a Lindenmayer system, also called a rewrite system. For the von Koch broken line the rewrite is: divide a line in 3 pieces and replace the middle piece by an equilateral triangle, with the base omitted. Repeat the process on the 4 line pieces to the required order. It is similar to the defining construction process of the Lévy fractal; the result conveys a different impression, however. Below $K_0 ... K_4$, scaled with increasing order (line thickness is scaled as well).



Lindemayer production rule enriched with PostScript concepts for the von Koch fractal

$K_n = [S_{\frac{1}{3}\frac{1}{3}} K_{n-1}] \oplus T_{\frac{s}{3}0} [S_{\frac{1}{3}\frac{1}{3}} R_{60} K_{n-1}] \oplus T_{\frac{s}{6}\frac{s\sqrt{3}}{6}} [S_{\frac{1}{3}\frac{1}{3}} R_{-60} K_{n-1}] \oplus T_{\frac{s}{6}\frac{-s\sqrt{3}}{6}} [S_{\frac{1}{3}\frac{1}{3}} K_{n-1}]$

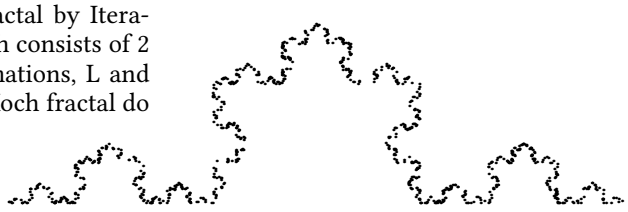
with, K_0 the initial line, K_n the von Koch curve of order n, $\oplus$ splice operator, meaning add properly, i.e. translate, [open a new GS on the GS stack,] remove current graphics state from the GS stack and recall previous, R_{60} means rotate US 60° in the PS sense, $S_{a,b}$ means scale US by a and b, in x- and y-direction $T_{a,b}$ means translate US by a and b, in x- and y-direction.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: vonKoch fractal. cgl, kisa1@xs4all.nl
%%BoundingBox: -1 -1 650 120
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/vonKoch{%on stack order >=0; ==> von Koch
         %s = size of the line segment (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub vonKoch %lower the order on the stack
   60 rotate vonKoch
 -120 rotate vonKoch
   60 rotate vonKoch
 1 add         %reset order
}ifelse}def
%%EndProlog
/s 5 def
0 vonKoch 10 0 translate
1 vonKoch 10 0 translate
2 vonKoch 10 0 translate
3 vonKoch 10 0 translate
4 vonKoch showpage
%%EOF
```

Von Koch-like fractal as Iterated Function System

Lauwerier(1994) in one of his exercises created a von Koch-like fractal by Iterated Function Systems, which consists of 2 contracted, affine transformations, L and R, which both for the von Koch fractal do contraction and mirroring.



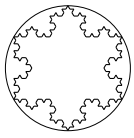
$$\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{\text{L}}{=} \begin{pmatrix} .5 & .289 \\ .289 & -.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -.5 \\ .289 \end{pmatrix} \text{ and } \begin{pmatrix} x' \\ y' \end{pmatrix} \overset{\text{R}}{=} \begin{pmatrix} .5 & -.289 \\ -.289 & -.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} .5 \\ .289 \end{pmatrix}$$

Associated with the von Koch fractal are 2 rotations with mirroring with centres for L: (-1,0) and for R: (1,0) and contraction $\sqrt{.5^2 + .289^2} \approx .58$. Amazing, isn't it! Laurier's BASIC program FRACMC4 and my transcription are given below. (MC is abbreviation for Monte Carlo, meaning alternate L and R by gambling.)

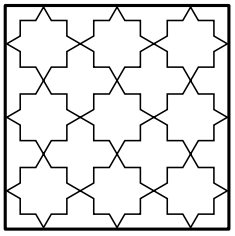
```
REM ***iteratief systeem,
REM    2 spiegelingen, FRACMC4***
REM ***coefficienten***
A=.5 : B=..289 : C=A : D=-B
DET1=A*A+B*B : DET2=C*C+D*D :
  Q=DET1/(DET1+DET2)
X=1 : Y=0 : K=0 : KMAX=1000
DO WHILE K<KMAX AND INKEY$=" "
   R=RND
   IF R<Q THEN
      X1=A*X+B*Y-1+A :
        Y1=B*X-A*Y+B 'spiegeling L
   ELSE
      X1=C*X+D*Y+1-C :
        Y1=D*X-C*Y-D 'spiegeling R
   END IF
   X=X1 : Y=Y1
   PSET (X,Y),10
   K=K+1
LOOP : BEEP
END

Other values of the parameters
a=.5 b=.5 c=.6667 d=0 %bebladerde tak
a=.5 b=.289 c=.5  d=-.289 %von Koch
a=.5 b=.5   c=.5  d=0     %kale tak
a=.5 b=.5   c=.6  d=-.2
a=0  b=.64  c=0   d=-.64  %tegelpatroon
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: fracmc4
%%Author: H.A. Lauwerier(1994): Spelen met Graphics en Fractals
%%BoundingBox: -100 -1 103 60
%%BeginSetup
%%EndSetup
%%BeginProlog
/Courier 7 selectfont
/x 1 def /y 0 def /halfmaxint 2 30 exp def/maxint 2 31 exp 1
sub
def
/a .5 def  /b .289 def /c a def /d b neg def
/det1 a dup mul b dup mul add def /det2 c dup mul d dup mul add
def
/q det1 det1 det2 add div def
/printxy {x s y s moveto (.) show}def
%%EndProlog
10 srand%10 is seed
/s {100 mul }def%scaling
1000{rand maxint div q lt
     {/xnew a x mul b y mul add 1 sub a add def
       /y b x mul a y mul sub b add def /x xnew def%mirror L
     }
     {/xnew c x mul d y mul add 1 add c sub def
      /y d x mul c y mul sub d sub def /x xnew def%mirror R
     }ifelse
     printxy
 }repeat
showpage
%%EOF
```

A von Koch island is a closed splicing of von Koch fractals; at right a von Koch tile (van der Laan(1997)).



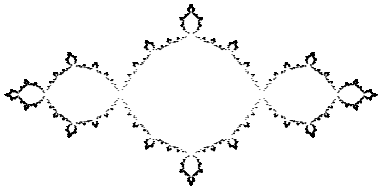
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: von Koch (triangular) island
%%...
/s 100 def
gsave .5 s mul dup neg exch translate 3 vonKochfractal pop
grestore
gsave .5 s mul dup translate
      -120 rotate 3 vonKochfractal pop grestore
gsave 0  -.366 s mul translate
      -240 rotate 3 vonKochfractal pop grestore
.001 setlinewidth 0 21 57.8 0 360 arc stroke
showpage
%%EOF
```



Julia fractals

The Julia set for $z_i = z_{i-1}^2 + c, \quad i = 1, 2, 3, ...$ is a repeller; the Julia set for the inverse iteration is an attractor. JULIAMC implements inverse iteration. The parameters of the JULIAMC are: the real and imaginary part of the problem parameter c, i.e. a and b, and the number of points of the fractal to be generated, n. It is the simplest Julia fractal generator.

$$z_i = z_{i-1}^2 + c, \quad i = 1, 2, 3, ... \qquad \rightarrow$$
$$\text{Inverse:} \quad z_{i-1} = \pm\sqrt{z_i - c}, \quad i = n + 10, ... , 1, \quad |z_{n+10}| \leq 1.$$



```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -165 -85 165 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/JULIAMCdict 11 dict def  %local dictionary
/JULIAMC{%stack: a, b, maxk.
         %a+ib complex constant c,  maxk maximal iterations
         % ==> Julia set of z^2 + a + ib
JULIAMCdict begin%open the local dictionary
/kmax exch def /b exch def /a exch def  /Courier 7 selectfont
/nextpoint{/x1 x a sub 2 div def /y1 y b sub 2 div def
            x1 dup mul y1 dup mul add sqrt dup%R
           /y exch x1 sub sqrt y1 0 lt{neg}if def
           /x exch x1 add sqrt def
}bind def
/printxy{x s y s        moveto (.) centershow
    x s neg y s neg     moveto (.) centershow   %point symmetry
    b 0 eq y 0 ne and
       {x s y s neg moveto (.) centershow
        x s neg y s moveto (.) centershow}if%symmetry x- and y-axes
}bind def
/s {100 mul} def                          %scaling
/nrand rand 2147483647 div def            %random number in [0,1]
/x nrand def /y nrand def                 %start values in [0,1]
10{nextpoint}repeat                       %discard 10 iterations
kmax{nextpoint
     rand 1073741823 gt {/x x neg def /y y neg def}if
     printxy
    }repeat
end}bind def
%%EndProlog
-1    0   5000 JULIAMC showpage  %SanMarco
-.59 -.34 5000 JULIAMC showpage  %cloud
%%EOF
```

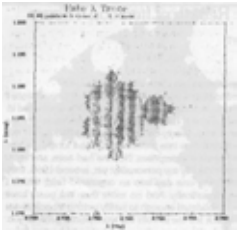
PSlib.eps contains moreover defs JULIABS, which implements the boundary scan method, JULIAF, which generates a filled fractal, JULIAP, which implements the pixel method, JULIAD, which is based on the distance formula.

Chaos and Fractalus are packages, which create fractals with rich colour possibilities.

Mandelbrot's fractal

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

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox:  -210 -135 85 135
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/MANDELdict 20 dict def
/MANDEL%==>Mandelbrotfractal in coloured bands
{MANDELdict begin /Courier 12 selectfont
/colours [/white /blue /lightblue /green /lightgreen
          /red /lightred /brown /yellow] def
/step .01 def /sc {100 mul} def
-2.1 step .85{/a exch def /asc a sc def
   0 step 1.35 {/b exch def /bsc b sc def
     /u 4 a dup mul b dup mul add mul def
     /v u 2 a mul sub .25 add def
     u 8 a mul add -3.75 le           %exclude cardioid
     v v sqrt sub 2 a mul add .5 le or%exclude circle
     {/l 0 def}%inside white, do nothing
     {/x a def /y b def /k 0 def
         {%loop over k
          /z x def
          /x x dup mul y dup mul sub a add def
          /y 2 z mul y mul b add def
          /s x dup mul y dup mul add def
          /k k 1 add def
           s 100 gt k 50 eq or{exit}if
         }loop%k
       k 40 lt{/l k 8 mod def}{/l 0 def}ifelse
       colours l get cvx exec
       k 3 gt{asc bsc     moveto (.) show
              asc bsc neg moveto (.) show}if
     }ifelse
   }for%j
}for%i
end}bind def
%%Endprolog
MANDEL
%respectively
%MANDELzw         %see PSlib.eps
%MANDELzwcontour  %see PSlib.eps
showpage
%%EOF
```



Mandelbrot's first M-fractal

Mandelbrot in 1980 answered the question:

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

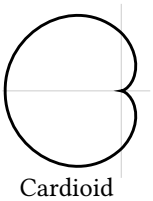
He was surprised, but ... realized the relevance.

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

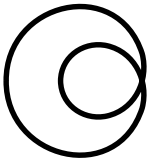
Form and size of M-fractal

M-cardiod

A Cardioid is defined in polar coordinates by $r = 2a(1 + \cos\theta)$, $\theta \in [0,2\pi]$. A version of the more general Limaçon, with parameter b=.25, $r = 2a(b + \cos\theta)$, $\theta \in [0,2\pi]$ is included at right.


Cardioid

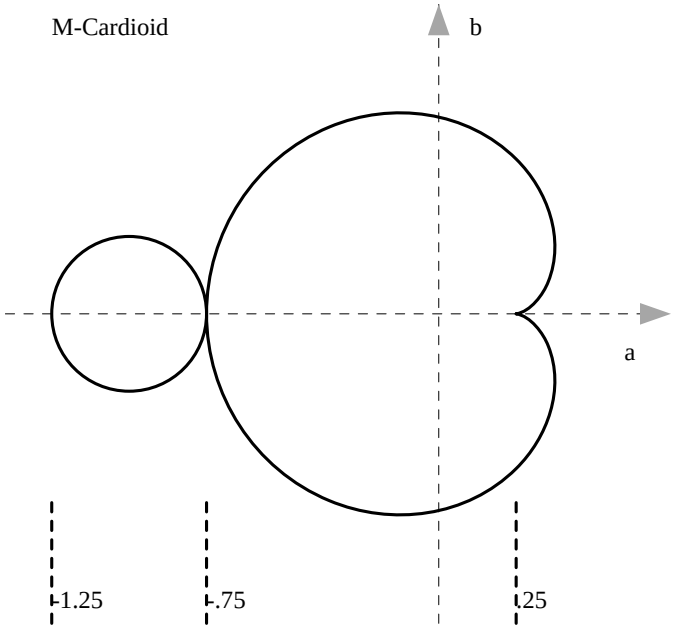
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Cardioid in Polar Coordinates
%%BoundingBox: 0-30 -41 30
%%BeginSetup
%%EndSetup
/t 0 def /2a -20 def 2 2a mul 0 moveto
120{3 rotate /t t 3 add def
    2a 1 t cos add mul 0 lineto}repeat
stroke showpage
```


Limaçon

The equations for the cardioid in Cartesian coordinates, parametric in $\varphi \in [0, 2\pi]$, read

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

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



Lauwerier(1995) contains a BASIC program for drawing a cardiod, `CARDIO`.

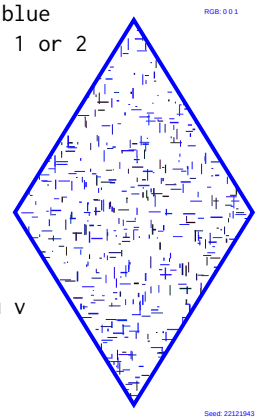
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: M-cardioid. cgl 2012
%%BoundingBox: -140 -105 80 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/m-cardioid{% Math cardoid of mandelbrot fractal
/Times-Roman 8 selectfont
/s {100 mul} def .01 setlinewidth [3] 0 setdash
-1.4 s 0 moveto .75 s 0 lineto
0 -1 s moveto 0 1 s lineto stroke
1 setlinewidth [] 0 setdash
.6 s -15 moveto (a) show
10 .9 s  moveto (b) show
.65 setgray%arrowheads coordinate axis
0 90 0 100 1 7 10 arrow fill %y-axis
65 0 75 0 1 7 10 arrow fill  %x-axis
0 setgray
.25 s 0 moveto
.1 .1 360.05{/phi exch def
    2 phi mul cos -4 div phi cos -2 div sub s
    2 phi mul sin 4 div phi sin 2 div sub s lineto
   }for
-.75 s 0 moveto
-1 s 0 .25 s 0 360 arc stroke
[3] 0 setdash
-1.25 s -1 s moveto -1.25 s -.6 s lineto stroke
-1.25 s -.95 s moveto (-1.25) show
-0.75 s -1 s moveto -0.75 s -.6 s lineto stroke
-0.75 s -.95 s moveto (-.75) show
0.25 s -1 s moveto 0.25 s -.6 s lineto stroke
0.25 s -.95 s moveto (.25) show
%
-1.25 s .9 s moveto (M-Cardioid) show
}bind def
%%EndProlog
m-cardioid showpage
%%EOF
```

Wim W. Wilhelm communicated his compact specification for drawing the cardioid in Mathematica, using Cartesian coordinates.

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

Mondriaan's line pieces.

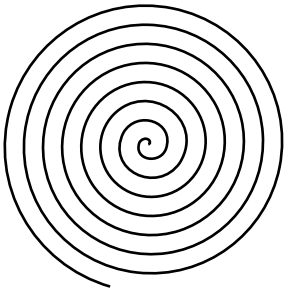
Since *à la Mondriaan* the PostScript program has been extended by a square and circle as frame. In the paper the MetaPost program as well as the PostScript program have been developed and discussed. Because of the seed the pictures are unique. It illustrates the use of PostScript's random number generator. The whole page is used.



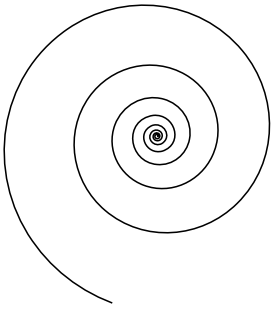

```
%!PS-Adobe-3.0 EPSF-3.0
(c:\\PSlib\\PSlib.eps) run
/mondrian% 2014 extended by circle and square
%birthday: ddmmyyyy, a number as seed for srand
%three numbers from the closed interval [0, 1], for the rgb-color values: red green blue
%number for the kind of frame (0=rectanglere 1=Oval 2=Lozenge 3=square 4=circle): 0, 1 or 2
%==> generated Mondrian-alike
{0 begin gsave %savety for not changing the graphics state outside
  % used from the library: unifrmdef, maxrandom, ellipse
/form exch def
/b exch def /g exch def /r exch def /date exch def
date srand% start random generator with (birthday date) seed
100 50 translate
%wired-in parameters
/u 420 def /v u 1.618 mul def /hu u 2 div def /hv v 2 div def%BB of rectangle: 0 0 u v
/maxrandom 500 def /maxlength 20 def /maxwidth 3 def /eps 0.1 def
/hx  {u unifrmdev}            def
/hy  {v unifrmdev}           def
/l    {maxlength unifrmdev}def
/w  {maxwidth unifrmdev} def
/spread {2 unifrmdev mul }def                 %(0, 2)
/color{r 0 eq {eps}{r} ifelse spread g 0 eq {eps}{g} ifelse spread b 0 eq {eps}{b} ifelse spread} def
form 0 eq {/contour {0 0 moveto u 0 lineto u v lineto 0 v lineto closepath} def} if      %rectangle
form 1 eq {/contour {hu hv hu hv 0 360 ellipse} def} if    %oval
form 2 eq {/contour {hu 0 moveto u hv lineto hu v lineto 0 hv lineto closepath} def} if%lozenge
form 3 eq {/contour {0 0 moveto u 0 lineto u u lineto 0 u lineto closepath} def} if      %square
form 4 eq {/contour {hu hv hu hu 0 360 ellipse} def} if    %circle
 %oval
gsave contour clip%random pattern will only show up in (is clipped to) contour
maxrandom{%draw pattern in loop confined to contour
/xaux hx def /yaux hy def%position in (0, u) x (0, v) rectangle
/laux l 2 div def
xaux laux sub yaux moveto  xaux laux add yaux lineto w setlinewidth color setrgbcolor stroke%h-line
/xaux hx def /yaux hy def
/laux l 2 div def
xaux yaux laux sub moveto  xaux yaux laux add lineto w setlinewidth color setrgbcolor stroke%v-line
}repeat
grestore %end clipping path
contour 7 setlinewidth r g b setrgbcolor stroke%original color od choice
H12pt setfont  /nstr 8 string def %0 0 0 setrgbcolor
u 85 sub v 10 add moveto (RGB: ) show
  r nstr cvs show ( ) show  g nstr cvs show ( ) show b nstr cvs show
u 85 sub -20 moveto (Seed: ) show  date nstr cvs show
grestore end}def%end Mondrian
/mondrian load 0 26 dict put
22121943 0 0 1 2 mondrian showpage %example of use
%%EOF
```

Spirals: Archimedian and Growth

The Archimedes Spiral is in polar coordinates defined by $r_\theta = k\theta, \quad 0 \leq \theta < \infty$.
The Growth Spiral is in polar coordinates defined by $\ln r_\theta = k\theta$ or $r_\theta = \mathrm{e}^{k\theta}, \quad -\infty < \theta < \infty$.

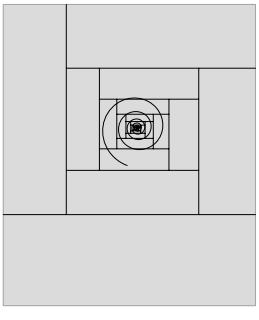


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Archimedes Spiral
%%BoundingBox: -56 -59 54 55
%%BeginSetup
%%EndSetup
/archimedesspiral{0 0 moveto
/f .1 def /r 0 def
555{5 rotate /r r f add def
    r 0 lineto}repeat stroke
}bind def
%%EndProlog
archimedesspiral showpage
%%EOF
```



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth Spiral
%%BoundingBox: -100 -110 75 86
%%BeginSetup
%%EndSetup
/growthspiralpc{1 0 moveto%off the 0
/f 2.718 .0085 exp def /r 1 def
555{5 rotate /r r f mul def
    r 0 lineto}repeat stroke
}bind def
%%EndProlog
growthspirapc
```

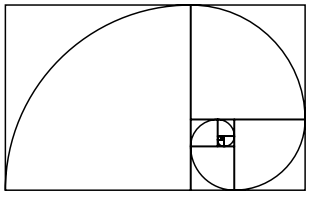
The Gyre Open font Type activity logo



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Imitation Gyre-logo, cgl 2012
%%BoundingBox: -2 -2 722 862
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run % contains growthspiral
%%EndProlog
/gyrelogodict 8 dict def
/gyrelogo{gyrelogodict begin
/ux 720 def /uy 860 def
gsave .86 setgray 0 0 ux uy rectfill
  3 setlinewidth .65 setgray 0 0 ux uy rectstroke%background
grestore
2 setlinewidth
/xl{ux 180 720 div mul}def /xu{ux xl .9 mul sub}def
/yl{uy 260 860 div mul}def /yu{uy yl .7 mul sub}def
7{xl yl moveto xl uy lineto 0 yl moveto  ux yl lineto %low
  xu yl moveto xu yu lineto xl yu moveto ux yu lineto %up
```

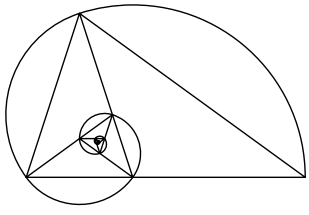
```
  stroke
  xl yl translate /ux xu xl sub def /uy yu yl sub def
}repeat
%growth spiral
ux uy translate growthspiralpc
end}def
gyrelogo showpage
%%EOF
```

Variant growth spiral



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth spiral. cgl 1997
%%BoundingBox: -16 -22 201 126
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/growthspiraldict 6 dict def
/growthspiral{growthspiraldict begin %tilLiv
0 setlinejoin 1 setlinecap
/x 200 def /y .618 x mul def
/square{0 y lineto y y lineto y 0 lineto
  y 0 y 180 90 arcn
  y y translate}def
12{0 0 moveto square -90 rotate
  /aux x def /x y def /y aux y sub def
 }repeat stroke
end}def
%%EndProlog
growthspiral showpage
%%EOF
```

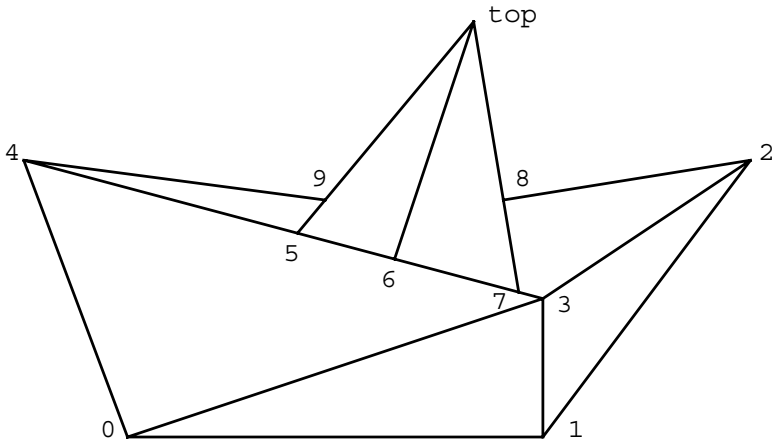
Variant growth spiral



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth spiral II. cgl 1997
%%BoundingBox: -16 -22 201 126
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/growthspiralIIdict 3 dict def
/growthspiralII{growthspiralIIdict begin %tilLvi
0 setlinejoin 1 setlinecap
/x 200 def /xg{.618 x mul}def
/tri{x xg sub 0 moveto
     currentpoint translate
     0 0 moveto 0 0 xg 0 108 arc
     xg 0 lineto stroke
    108 rotate /x xg def
}def
12{tri}repeat
end}def
%%EndProlog
growthspiralII showpage
%%EOF
```

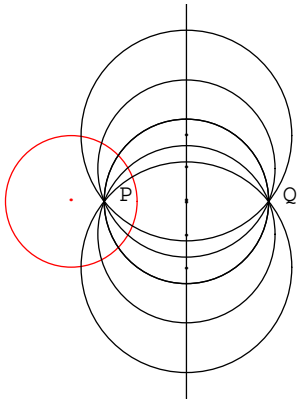
Gust battleship logo

The programming is interesting because the intersection point of straight lines has to be calculated.




```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: GUST Battleship, 1996
%%BoundingBox: -41 -1 231 151
%%BeginSetup
%%EndSetup
%%BeginPrologue
(C:\\PSlib\\PSlib.eps) run
/battleshipdict 13 dict def
battleshipdict begin
/s 50 def 2 setlinejoin
/p0{0 0}def
/p1{3 s mul 0}def
/p2{4.5 s mul 2 s mul}def
/p3{3 s mul s}def
/p4{-.75 s mul 2 s mul}def
/top{2.5 s mul 3 s mul}def
/p5{p0 top p3 p4 intersect}def
/p6{p0 p1 mean top p3 p4 intersect}def
/p7{top p1 p3 p4 intersect}def
/p8{p2 p5 top p1 intersect}def
/p9{p8 dup 0 exch top p0 intersect}def
end
%
/battleship{battleshipdict begin
p0 moveto p1 lineto p2 lineto p3 lineto p0 lineto
p1 moveto p3 lineto p4 lineto p0 lineto
top moveto p5 lineto
top moveto p6 lineto
top moveto p7 lineto
p2 moveto p8 lineto
p4 moveto p9 lineto
stroke
end}def
%%EndPrologue
battleship showpage
```

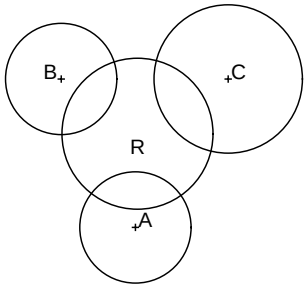
Orthogonal circles II



The problem is given a circle and a point P inside the circle construct a circle bundle through P which cut the original circle orthogonally. (Q is the inverse of P).

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Orthogonal circle bundle. cgl 2014
%%BoundingBox: -105 -150 125 150
(c:\\PSlib\\PSlib.eps) run
/orthogonalcirclebundledict 11 dict def
/orthogonalcirclebundle{orthogonalcirclebundledict begin
/r 50 def /Courier 18 selectfont
/px -25 def/py 0 def
/qx r r mul  px neg div def /qy 0 def%inverse
gsave 1 0 0 setrgbcolor -50 0 r 0 360 arc stroke
-50 0 moveto (.)centershow  %(C) show
grestore
px py moveto (.) centershow ( P) show
qx qy moveto (.) centershow ( Q) show
/m px qx add 2 div def%middle
m 150 moveto m -150 lineto stroke%middleline
2{/r1 m px sub def
m 0 r1 0 360 arc stroke gsave m 0 moveto (.)centershow grestore
/r2 m px sub dup mul 625 add sqrt def
m 25 r2 0 360 arc stroke gsave m 25 moveto (.)centershow grestore
/r3 m px sub dup mul 2500 add sqrt def
m 50 r3 0 360 arc stroke gsave m 50 moveto (.)centershow grestore
1 -1 scale}repeat
end}def
orthogonalcirclebundle showpage
%%EOF
```

Orthogonal circles III. Radical circle



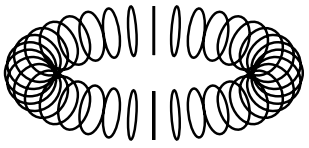
The problem is given three circles construct a circle which cuts the original circles orthogonally.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Radical circle. CGL april2010
%%BoundingBox: -190 -185 235 215
(C:\\PSlib\\PSlib.eps) run %PS library
/r 100 def /mr r neg def  H14pt setfont
/Ax 0 def  /Ay mr  def           /A {Ax Ay} def  /Ar .75 r mul def
/Bx mr def /By r  def            /B {Bx By} def   /Br .75 r mul def
/Cx 1.25 r mul def  /Cy r def   /C {Cx Cy} def  /Cr r def
A plus B plus C plus
newpath A Ar 0 360 arc stroke A moveto  2 0 rmoveto (A) show
newpath B Br 0 360 arc stroke  B moveto -12 0 rmoveto (B) show
newpath C Cr 0 360 arc stroke C moveto   2 0 rmoveto (C) show
Ax Ay Ar Bx By Br Cx Cy Cr radical
/radr exch def /rady exch def /radx exch def
newpath radx rady radr 0 360 arc stroke
radx rady plus
radx 3 add rady 5 sub moveto (Radical) show showpage
%%EOF
```

An ill-posed subproblem — touching point of 2 circles — has been reformulated in a well-posed problem.

Toroid

Lauwerier in *Meetkunde met de microcomputer* explained and demonstrated the picture.

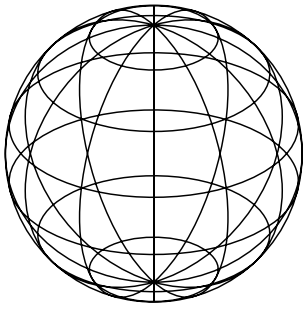


```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -61 -28 61 28
%%BeginProlog
(c:\\PSlib\\Pslib.eps) run
/toroiddict 16 dict def
toroiddict begin%locals
/R 50 def /r 10 def /phi 0 def /theta 20 def
/a1{R r sub t cos mul   R r sub t sin mul   0         ptp}def
/a2{R r sub t cos mul   R r sub t sin mul   .55 r mul ptp}def
/a3{R r sub .55 r mul add t cos mul R r sub .55 r mul add t sin mul r ptp}def
/a4{R t cos mul    R t sin mul              r         ptp}def
/a5{R .55 r mul add t cos mul R .55 r mul add t sin mul r ptp}def
/a6{R r add t cos mul   R r add t sin mul   .55 r mul ptp}def
/a7{R r add t cos mul   R r add t sin mul   0         ptp}def
/a8{R r add t cos mul   R r add t sin mul  -.55 r mul ptp}def
/a9{R .55 r mul add t cos mul R .55 r mul add t sin mul r neg  ptp}def
/a10{R  t cos mul       R  t sin mul         r neg  ptp}def
/a11{R .55 r mul sub t cos mul  R .55 r mul sub t sin mul r neg  ptp}def
/a12{R r sub t cos mul   R r sub t sin mul  -.55 r mul ptp}def
end
/toroid{toroiddict begin
%used from library: ptp
0 10 360{/t exch def a1 moveto a2 a3 a4 curveto a5 a6 a7 curveto
  a8 a9 a10 curveto a11 a12 a1 curveto closepath
}for stroke
end}def
%%EndProlog
toroid showpage
%%EOF
```

A direct translation of Lauwerier's BASIC code

```
%!PS-Adobe-3.0 EPSF=3.0  %Torus, cgl Jan 07
%%BoundingBox: -300 -100 300 300
%%Creator: H Lauwerier. Meetkunde met de microcomputer. Adapted to PS cgl Jan2007
/PI 3.141593 def /C .8 def  /C1 .7071 1 C C mul sub sqrt mul def
/r1 235 def /r2 40 def
/ptp2{/z exch def/y exch def/x exch def %point to pair projection
   y x sub .7071 mul
   C z mul x y add C1 mul sub}def
/n 75 def 300 300 translate
0 1 n 1 sub{/k exch def
   /A k n div 360 mul cos def
   /B k n div 360 mul sin def
   0 10 360{/t exch def
            B neg r1 r2 t cos mul add mul %x stacked
            A     r1 r2 t cos mul add mul %y stacked
            r2 t sin  mul                 %z stacked
            ptp2                          %u,v stacked xyz consumed
            t 0 eq{moveto}{lineto}ifelse
   }for stroke %less memory requirement than stroke on the end
}for
showpage
```

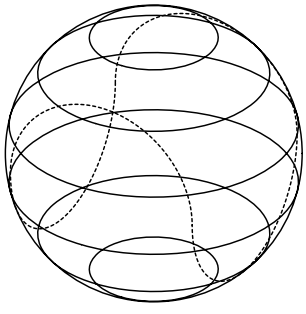
Sphere with meridians and latitude circles



```
%%!PS-Adobe-3.0 EPSF-3.0
%%Title: Spere with meridans,latitude circles
%%Author: H A Lauwerier(1987): Meetkunde met de microcomputer, Epsilon
%%Transcriptor: Kees van der laan, kisa1@xs4all.nl,  2012
%%BoundingBox: -101 -102 101 102
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/spheremeridianslatitudesdict 25 dict def
/spheremeridianslatitudes{spheremeridianslatitudesdict begin
/m 6 def /n 6 def /r 100 def /-r r neg def
/latitudecircles{1 1 m{/i exch def /t i 180 mul m 1 add div def
 /x r t sin mul def /y 0 def /z r t cos mul def
 x y z ptp moveto
 1 1 100{/j exch def  /s j 180 mul 50 div def
         /x{r s cos t sin mul mul}def
         /y{r s sin t sin mul mul}def
         x y z ptp lineto
         }for %j
}for %i
stroke}def%latitudecircles
/meridians{1 1 n{/i exch def /s i 180 mul n div def
  /x{0}def /y{0}def /z{r}def
 x y z ptp moveto
 1 1 100{/j exch def  /t j 180 mul 50 div def
         /x{r s cos t sin mul mul}def
         /y{r s sin t sin mul mul}def
         /z{r t cos mul}def
         x y z ptp lineto
        }for %j
}for %i
stroke}def%meridians
r 0 moveto  0 0 r 0 360 arc stroke%circle in projection plane
latitudecircles
meridians
end} bind def
%%Endprolog
/phi 30 def /theta 30 def
spheremeridianslatitudes
showpage
%%EOF
```

With tennisball curve dashed

(note scaling $\sqrt{2}$, because tennisball curve is for sphere with radius $\sqrt{2}$.)

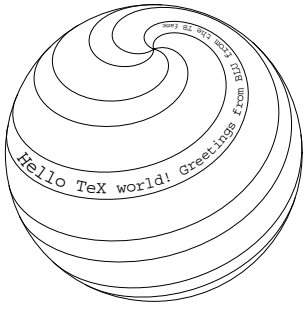


```
%tenniscurve
/tennisball{/r r 2 sqrt div def /-r r neg def
%Tennisball part {t:(1, sin t, cos t)}
r -r 0 ptp moveto
-89 1 90{/t exch def
 r  t sin r mul  t cos r mul  ptp lineto
```

```
}for
%Tennisball part {t:(-1, sin t, cos t)}
-r   -r   0  ptp moveto
-89 1 90{/t exch def
-r   t sin r mul  t cos r mul  ptp lineto
}for
%Tennisball part {t:(-sin t, 1, -cos t)}
r   r   0 ptp moveto
-89 1 90{/t exch def
t sin neg r mul   r   t cos neg r mul    ptp lineto
}for
%Tennisball part {t:(-sin t, -1, -cos t)}
r   -r   0 ptp moveto
-89 1 90{/t exch def
t sin neg r mul    -r    t cos neg r mul   ptp lineto
}for
[2] 1 setdash stroke}def%tennisball
```

Sphere with spiral and text

Interesting is the diminishing font.

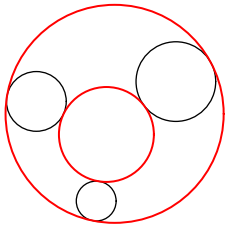


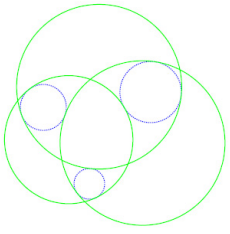

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Spere with belts and text, kisa1@xs4all.nl, 2012
%%BoundingBox: -101 -102 101 102
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
/Courier 14 selectfont
pathtextdict begin /size 14 def
/linetoproc
  { /oldx newx def /oldy newy def
       /newy exch def /newx exch def
       /dx newx oldx sub def
       /dy newy oldy sub def
       /dist dx dup mul dy dup mul add sqrt def
       dist 0 ne
         { /dsx dx dist div ovr mul def
           /dsy dy dist div ovr mul def
           oldx dsx add oldy dsy add transform
           /cpy exch def /cpx exch def
           /pathdist pathdist dist add def
              {
/size size .08 sub def
/Courier findfont [size 0 0 size 0 0] makefont setfont
                 setdist pathdist le
                { charcount str length lt
                     {setchar} {exit} ifelse }
                { /ovr setdist pathdist sub def exit }
                ifelse
              } loop
      } if
  } def
end
/sphereandspiraldict 30 dict def
/sphereandspiral{sphereandspiraldict begin /r 100 def
/c .7 def /a2 1 1.41421 div def /b1 a2 1 c c mul sub sqrt mul def
/c1 c a2 mul def /c3 1 c c mul sub sqrt def
/spiral{-90 1 +90{/thetaj exch def
 /phij thetaj 4 mul phi0 add def%windings
 /x r phij cos thetaj cos mul mul def
 /y r phij sin thetaj cos mul mul def
 /z r thetaj sin mul def
 /w c1 x y add mul c3 z mul add def
 /u a2 y x sub mul def
 /v c z mul b1 x y add mul sub def
 w 0 lt{u v moveto}{u v lineto}ifelse
}for %thetaj
}def%spiral
gsave
0 c r mul moveto /phi0   0 def spiral
0 c r mul moveto /phi0  45 def spiral stroke
grestore
```

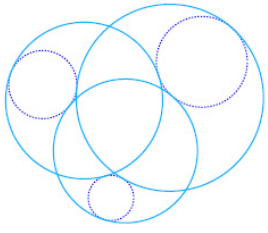
```
gsave
0 c r mul moveto /phi0 120 def spiral
0 c r mul moveto /phi0 165 def spiral stroke
grestore
gsave
0 c r mul moveto /phi0 240 def spiral stroke
0 c r mul moveto /phi0 285 def spiral stroke
0 c r mul moveto /phi0 275 def spiral
( Hello TeX world! Greetings from BLU from the TB fame) 20 pathtext
grestore
%circle in projection plane
r 0 moveto
1 1 100{/k exch def  /t k 180 mul 50 div def
        r t cos mul r t sin mul lineto
}for %k
stroke end} bind def
```

Apollonius problem

Given three disjunct circles find the circles touching the circles. In *Circle Inversion* the problem has been solved and all 8 solutions have been programmed in MetaPost and PostScript. A universal PostScript definition `apollonius` has emerged. The definition for a slight variant for the case when one given circle contains the two other ones is called `apollonius2`. The inversion method for solving Apollonius problem has also been explained and illustrated in the article.



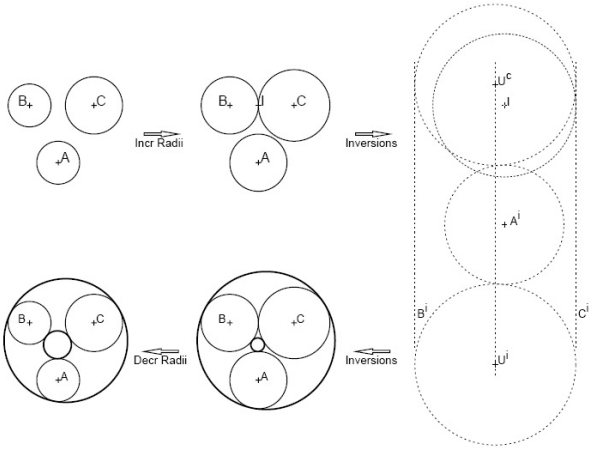




```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Apollonius. Three circles ->circum-/inscribed circle. cgl 2010
%%BoundingBox: -71 -50 100 121
(C:\\PSlib\\PSlib.eps) run
%given circles
/r 15 def /r1 r  def /x1 0 def /y1 r -2 mul def
/r2 r 1.5 mul def /x2 r -3 mul def  /y2 x2 neg def
/r3 r 2 mul def /x3 r 4 mul  def  /y3 x3 def
newpath%show the given circles
x1 y1 r1 0 360 arc stroke
x2 y2 r2 0 360 arc stroke
x3 y3 r3 0 360 arc stroke
%inscribed circle
x1 y1 r1
x2 y2 r2
x3 y3 r3  Apollonius /ri exch def /yi exch def /xi  exch def
red newpath xi yi ri  0 360 arc stroke%inscribed
%circumscribed circle
x1 y1 r1 neg
x2 y2 r2 neg
x3 y3 r3 neg Apollonius /rout exch def /yout exch def /xout exch def
newpath  xout yout rout 0 360 arc stroke%circumscribed
showpage
%%EOF
```

Note that the radius is specified negatively when the solution circle should contain the given circle. So the blue case has one negative radius specified and the green case two.

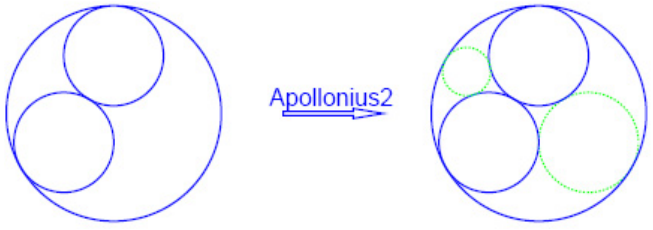
Below a picture of the inversion method.



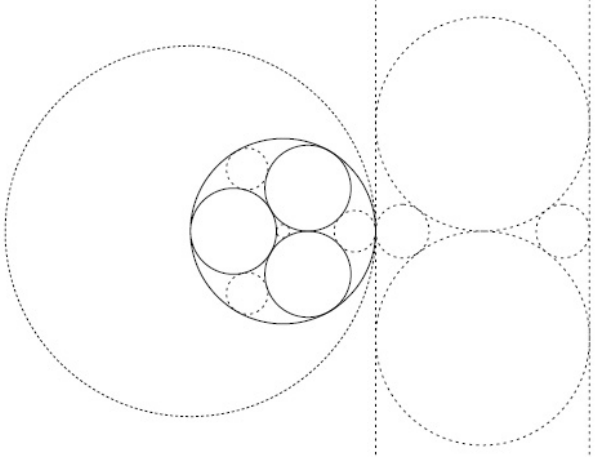
Apollonius problem, variant non-disjunct circles

Given three non-disjunct circles, one circle contains the other two, find the circles touching the circles. In *Circle Inversions* the problem has been solved and all 8 solutions have been programmed in MetaPost and PostScript. A universal PostScript definition `apollonius2` for this case has emerged.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Rerich Circles. CGL 2010
%%BoundingBox: 40 240 370 360
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/r 25 def /y r 3 sqrt div dup add def  /x 0 def
/x1 x def /y1 y def /r1 r def
/ri y r sub def %center  (0,0)
/R y r add def %center  (0,0)
x y -120 rot    %rotate center
/y2 exch def   /x2 exch def
/y3 y2 def      /x3 x2 neg def

100 300 translate%left figure
 0 0 R 0 360 arc stroke
gsave 2{newpath x y r 0 360 arc stroke 120 rotate}repeat
grestore

85 0 translate newpath 0 0 50 0 2 5 15 arrow stroke
                    -9 +4 moveto H12pt setfont (Apollonius2) show

128 0 translate%right figure
newpath 0 0 R 0 360 arc stroke
gsave 2{newpath x y r 0 360 arc stroke 120 rotate}repeat
grestore

x1 y1 r
x3 y3 r
0 0 R neg   Apollonius2
/rsnd1 exch def /ysnd1 exch def /xsnd1 exch def
/rsnd2 exch def /ysnd2 exch def /xsnd2 exch def
newpath xsnd2 ysnd2 rsnd2 0 360 arc stroke
newpath xsnd1 ysnd1 rsnd1 0 360 arc stroke
showpage
%%EOF
```



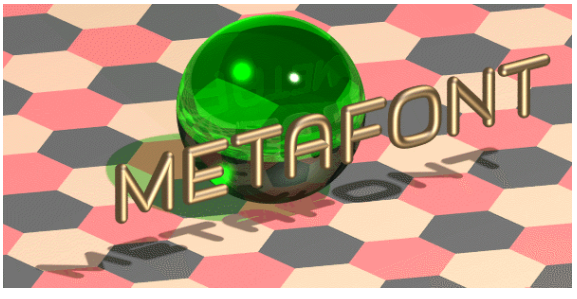
Tiling a circle by circles via inversion, tedious.



Jacko's Toruń98 logo.
The above is my best mimic in PostScript of Jackowski's Toruń98 dynamic masterpiece logo.

Metafont&TEX can be used to create beautiful artistic results with fonts.

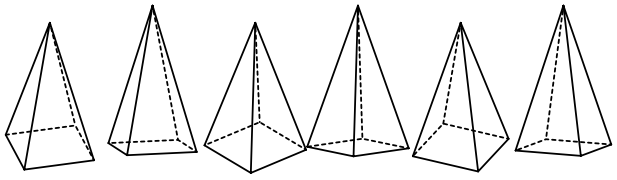
This has been shown in the 90-ies by Bogusław Jackowski and Marek Ryćko. Non-scalability is not relevant for pieces of art.



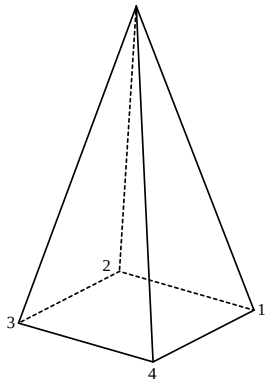
I could not reproduce these TEX-Metafont pictures in PostScript. Maybe the TEXnique has become outdated in view of the OpenTypeFonts activity. A challenge for the reader to realize this with OTFs?

Pyramids

Hobby showed in his MetaPost manual one full-page pyramid in MetaPost. Below are pyramids specified in 3D and projected with varying viewing angles.



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pyramids in projection, cgl 2009
%%BoundingBox: -25 -20 315 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/pyramidsdict 9 dict def
pyramidsdict begin
/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
1 setlinecap 1 setlinewidth
/z1{r neg r   0 ptp}def
/z2{r neg dup 0 ptp}def
/z3{r r neg   0 ptp}def
/z4{r r       0 ptp}def
/top{0 0 r 4 mul ptp}def
%
/pyramid{z1 moveto z2 lineto z3 lineto
   [2]1 setdash stroke
z3 moveto z4 lineto z1 lineto
   []0 setdash stroke
top moveto z1 lineto
top moveto z3 lineto
top moveto z4 lineto
%   -3 -10 rmoveto (4)show
stroke
top moveto z2 lineto
[2]1 setdash stroke
%  -10 0 rmoveto (2)show
} def  %end pyramid
end%pyramidsdic
/pyramids{pyramidsdict begin
15  25  65{/a exch def
30  -20 10{/b exch def
   pyramid 57 0 translate
}for}for
end}def
%%endProlog
pyramids  showpage
%%EOF
```

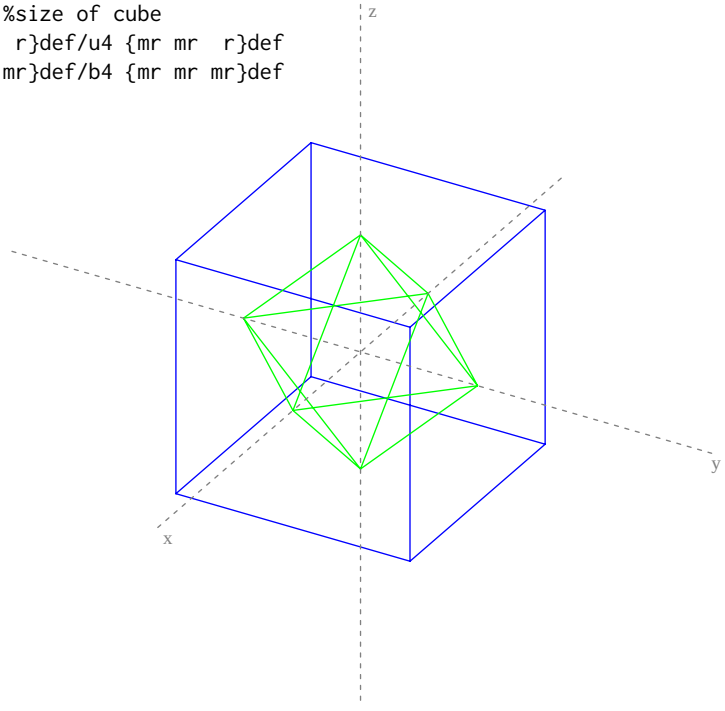



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pyramid in projection, cgl 1997
%%BoundingBox: -90 -40 90 190
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/pyramid{/ptp{/z exch def/y exch def/x exch def
                       x -.6 mul y .8 mul add
                        -4 x mul -3 y mul add
                                                12 z mul add 13 div}def
 /Times-Roman findfont 10 scalefont setfont
 /r 50 def /top{0 0 r 4 mul ptp}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
 z1 moveto z2 lineto z3 lineto  [2]1 setdash stroke
 z3 moveto z4 lineto z1 lineto  []0  setdash stroke
 top moveto z1 lineto    2  -3 rmoveto(1)show
 top moveto z3 lineto    -7  -3 rmoveto(3)show
 top moveto z4 lineto   -3 -10 rmoveto(4)show  stroke
 top moveto z2 lineto
   -10  0 rmoveto(2)show
 [2]1 setdash stroke  }def
%%EndProlog
pyramid showpage
%%EOF
```

Octaeder in cube

Essentially an old program. Nowadays I would use `ptp` in the definition of the points.

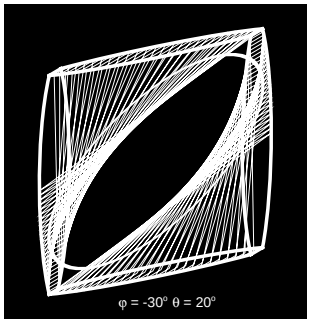
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Cube-Oktaeder
%%BoundingBox: -280 -280 280 280
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/cubedict 12 dict def
/cube{cubedict begin/r 100 def /mr r neg def %size of cube
/u1 { r mr  r}def/u2 { r  r  r}def/u3 {mr  r  r}def/u4 {mr mr  r}def
/b1 { r mr mr}def/b2 { r  r mr}def/b3 {mr  r mr}def/b4 {mr mr mr}def
u1 ptp moveto %u1%upper
u2 ptp lineto %u2
u3 ptp lineto %u3
u4 ptp lineto %u4
u1 ptp lineto %u1
b1 ptp moveto %b1 %lower
b2 ptp lineto %b2
b3 ptp lineto %b3
b4 ptp lineto %b4
b1 ptp lineto %b1
%edges (vertical)
u1 ptp moveto %u1
b1 ptp lineto %b1
u2 ptp moveto %u2
b2 ptp lineto %b2
u3 ptp moveto %u3
b3 ptp lineto %b3
u4 ptp moveto %u4
b4 ptp lineto %b4
end}def
/octaederdict 12 dict def
/octaeder{octaederdict begin
/r 100 def /mr r neg def  /2r r 2 mul def  /m2r 2r neg def %size of octaeder
top {0 0  r  }def/v4  {0 mr 0  }def /v1  {r 0  0  }def /v2  {0  r 0   }def
/v3  {mr 0  0 }def /nadir {0 0  mr}def
top ptp moveto %top
v4 ptp lineto
v1 ptp lineto %v1
v2 ptp lineto %v2
v3 ptp lineto %v3
v4 ptp lineto %v4
%
top ptp moveto %top
v1  ptp lineto %v1
top ptp moveto %top
v2  ptp lineto %v2
top ptp moveto %top
v3  ptp lineto %v3
%
nadir ptp moveto %nadir
```

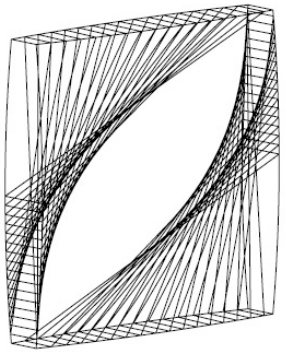


```
v4 ptp lineto %v4
nadir ptp moveto
v1 ptp lineto %v1
nadir ptp moveto
v2 ptp lineto %v2
nadir ptp moveto
v3 ptp lineto %v3
}def
/cubeoktaederdict 5 dict def
/cubeoktaeder{cubeoktaederdict begin  /Times-Roman 14 selectfont
cube     0 0 1 setrgbcolor stroke
octaeder 0 1 0 setrgbcolor stroke
0.5 0.5 0.5 setrgbcolor
%x-y-z coordinate axis
/r r 3 mul def /mr r neg def
mr 0 0 ptp moveto r  0 0 ptp lineto 4 -12 rmoveto (x) show
0 mr 0 ptp moveto 0  r 0 ptp lineto 0 -12 rmoveto (y) show
0 0 mr ptp moveto 0 0  r ptp lineto 6 -12 rmoveto (z) show [3 5]6 setdash stroke
end}def
/phi 30 def/theta 30 def cubeoktaeder
showpage
%%EOF
```

Emulation Gabo's linear construction no1

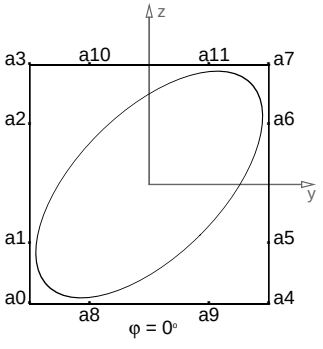
For more Gabo emulations, see Gabo's Torsion.

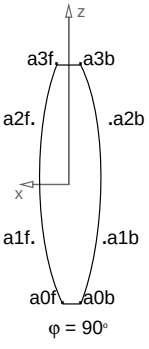




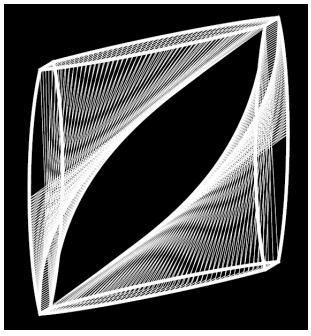
```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -61 -28 61 28
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\Pslib.eps) run
/linearconstructionno1dict 50 dict def
linearconstructionno1dict begin
/reversevideo {-130 -135 260 270 rectfill} def %Mimics BoundingBox
%Data
/origin { 0 0 } def
/a0f  { .1 s  -2 s  -2 s ptp  }def%y constant
/a1f  { .6 s  -2 s  -1 s ptp  }def
/a2f  { .6 s  -2 s   1 s ptp  }def
/a3f  { .2 s  -2 s   2 s ptp  }def

/a4f  { .2 s   2 s  -2 s ptp  }def%y constant
/a5f  { .6 s   2 s  -1 s ptp  }def
/a6f  { .6 s   2 s   1 s ptp  }def
/a7f  { .2 s   2 s   2 s ptp  }def

/a8f  { .6 s   -1 s -2 s ptp  }def%z constant
/a9f  { .6 s    1 s -2 s ptp  }def

/a10f { .6 s   -1 s  2 s ptp  }def%z constant
/a11f { .6 s    1 s  2 s ptp  }def

/a0b  { -.2 s  -2 s -2 s ptp  }def%y constant
/a1b  { -.6 s  -2 s -1 s ptp  }def
/a2b  { -.7 s  -2 s  1 s ptp  }def
/a3b  { -.2 s  -2 s  2 s ptp  }def

/a4b  { -.2  s  2 s -2 s ptp  }def%y constant
/a5b  { -.6  s  2 s -1 s ptp  }def
/a6b  { -.6  s  2 s  1 s ptp  }def
/a7b  { -.2  s  2 s  2 s ptp  }def

/a8b  { -.6 s  -1 s -2 s ptp  }def%z constant
/a9b  { -.6 s   1 s -2 s ptp  }def

/a10b { -.6 s  -1 s  2 s ptp  }def%z constant
/a11b { -.6 s   1 s  2 s ptp  }def

/sampleellipsestroke{gsave
-45 rotate
.6 1.2 scale%kind of ellipse
1 1 360{/t exch def
2 s t cos mul 2 s t sin mul }for %sample the skewed ellipse
count /n exch def
0 3 1 roll  % 0 y z
ptp         % u v projected point
moveto
n 2 div 1 sub cvi {0 3 1 roll  % 0 y z
             ptp         % u v projected point
```

```
            lineto
           }repeat closepath
stroke grestore} def
%
/approxellipsestroke{gsave
-45 rotate
.6 1.2 scale%kind of ellips
/a0 {0 -2 s    0     ptp} def
/a1 {0 -2 s    1.1 s ptp} def
/a2 {0 -1.1 s  2 s   ptp} def
/a3 {0  0      2 s   ptp} def
/a4 {0  1.1 s  2 s   ptp} def
/a5 {0  2 s    1.1 s ptp} def
/a6 {0  2 s    0     ptp} def
/a7 {0  2 s   -1.1 s ptp} def
/a8 {0  1.1 s -2 s   ptp} def
/a9 {0  0     -2 s   ptp} def
/a10{0 -1.1 s -2 s   ptp} def
/a11{0 -2 s   -1.1 s ptp} def
a0 moveto a1  a2  a3 curveto
          a4  a5  a6 curveto
          a7  a8  a9 curveto
          a10 a11 a0 curveto stroke
grestore} def
/frame{
a0f moveto
  a1f  a2f  a3f curveto
  a10f a11f a7f curveto
  a6f  a5f  a4f curveto
  a9f  a8f  a0f curveto
a0b moveto
  a1b  a2b  a3b curveto
  a10b a11b a7b curveto
  a6b  a5b  a4b curveto
  a9b  a8b  a0b curveto
a0f moveto a0b lineto
a3f moveto a3b lineto
a4f moveto a4b lineto
a7f moveto a7b lineto
} def
/dostringing{
0.02 .02 .5001{/t exch def
t       a0f a1f  a2f  a3f tOnSpline moveto
2 t mul a3b a10b a11b a7b tOnSpline lineto
2 t mul a3f a10f a11f a7f tOnSpline lineto
t       a0b a1b  a2b  a3b tOnSpline lineto
closepath
t       a7f a6f  a5f  a4f tOnSpline moveto
2 t mul a4b a9b  a8b  a0b tOnSpline lineto
2 t mul a4f a9b  a8f  a0f tOnSpline lineto
t       a7b a6b  a5b  a4b tOnSpline lineto
closepath
}for } bind def
.1 setlinewidth stroke
/annotation
```





```
{-32 -125  moveto
 S12pt setfont (j) show ( = ) H12pt setfont show phi (   ) cvs show
   gsave 0 5 rmoveto (o) H7pt setfont show grestore
8 0 rmoveto
 S12pt setfont (q) show ( = ) H12pt setfont show theta (  ) cvs show
         0 5 rmoveto (o) H7pt setfont show
} def
end


/linearconstructionno1
{linearconstructionno1dict begin
 /phi 30 def /theta 5 def /scalingfactor 50 def
reversevideo 1 setgray%white
3  setlinewidth frame        stroke
2  setlinewidth approxellipsestroke %sampleellipsestroke
.5 setlinewidth dostringing  stroke
annotation
end}def
%%Endprolog
linearconstructionno1
%%EOF
```



Conclusions

The use of PostScript pictures in AllTEX documents has a history of 20 years already, to start with the use of DVIPS. With pdfTEX the use of direct PostScript inclusion was hampered. .eps has to be converted into .pdf. Adobe seems to have stopped the support of the run command for library inclusion, which is a big loss. It seems that the use of PostScript pictures is still possible in TEX documents, but the future does not look promissing. This holds also for MetaPost and PSTricks pictures, because MetaPost is just a preprocessor for PostScript and PSTricks uses PostScript under the hood. For the future we need a better and more stable graphics tool to cooperate with TEX&Co. On the other hand: is it too optimistic to expect that the PostScript programs will be read and used?

Acknowledgements

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

Hàn Thế Thành for pdf(La)TEX,

Thank you Jos Winnink and Henk Jansen for proofing. MAPS editors for improving my use of English.

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

Spirals in PostScript

— *Polar Coordinates and PostScript are mates* —

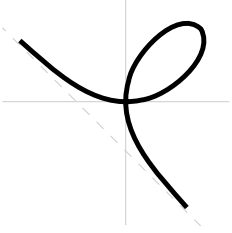
Abstract
Curves specified in Polar Coordinates can be elegantly programmed in PostScript with the rotate command; which performs rotations in User Space. This has been shown for the Cardioid, the Limaçon, the Lemniscate, the Archimedes and the Growth spiral. The Gyre-logo has been analyzed and imitated in PostScript. Printing of text along spiral-like belts on a sphere in the projection plane has been done, yielding poor man's typesetting text on a sphere in projection.

Keywords
Acrobat Pro, Adobe, art, BLUe, Cardioid, ConTEXt, Cornu, EPSF, Escher, FIFO, folium Descartes, function-grapher previewer, Gyre-logo, hidden lines, IDE (Integrated Development Environment), Jackowski, Lauwerier, Lemniscate, LIFO, Limaçon, Mathematica, MetaPost, Metafont, minimal encapsulated PostScript, minimal mark-up, minimal plain TeX, Photoshop, polar coordinates, projection, PSlib, PSTricks, PSView, rotation of US, Ryćko, Spirals, spherical spiral, TEXworks, Voss.

Prologue

When a curve is specified by Polar Coordinates, $\{\theta, r\}$, the curve is usually drawn by connecting the Cartesian data $(x_i, y_i) = (r_{\theta_i} \cos\theta_i, r_{\theta_i} \sin\theta_i)$, parametric in θ_i.

Folium of Descartes is specified by the equation $x^3 + y^3 + 3axy = 0$.
When I considered how to plot this folium, it was not clear to me how to begin. Using the Cartesian data $(x_i, y_i) = (r_{\theta_i} \cos\theta_i, r_{\theta_i} \sin\theta_i)$, parametric in $\theta_i \in [-30, 120]$, yielded the compact one-sweep drawing.



```
%!PS-Adobe-3.0 EPSF-3.0
...
/foliumDescartes{/3a 30 def
/r {3a t sin t cos  mul mul
       t sin dup dup mul mul
       t cos dup dup mul mul add div} def %radius vector
/t -30 def r t cos mul  r t sin mul moveto%start left up
-30  2 120{/t exch def                    %to right under
    r t cos mul r t sin mul lineto}for stroke
} bind def
```

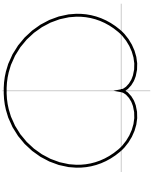
Polar Coordinates

$$r_\theta = \frac{3a \sin\theta \cos\theta}{\sin^3\theta + \cos^3\theta}$$

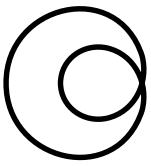
Asymptote $x + y + a = 0$.

A curve specified by polar coordinates can sometimes be programmed in PostScript without the calculation of the intermediate Cartesian coordinates, $\{x, y\}$, exploiting the fact that PostScriptperforms rotations in User Space.

Examples A Cardioid is defined in polar coordinates by $r = 2a(1+\cos(\theta))$, $\theta \in [0,2\pi]$. A version of the more general Limaçon, with parameter b=.25, $r = 2a(b+\cos(\theta))$, $\theta \in [0,2\pi]$ is included at right.


Cardioid

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Cardioid in Polar Coordinates
%%BoundingBox: 0-30 -41 30
%%BeginSetup
%%EndSetup
/t 0 def /2a -20 def 2 2a mul 0  moveto
120{3 rotate /t t 3 add def
    2a 1 t cos add mul 0 lineto}repeat
stroke showpage
```

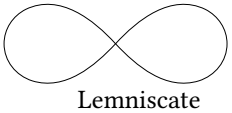

Limaçon

Without the use of the rotation of US facility the coding of the loop would read

```
0 3 360{/theta exch def /r 2a 1 t cos add mul def r theta cos mul r theta sin mul lineto}for
```

The difference is '`r theta cos mul r theta sin mul lineto`' versus '`r 0 lineto`'. Irrelevant details?[1]

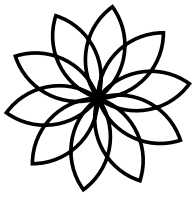
The Lemniscate is defined in polar coordinates by $r^2 = 2a^2\cos(2\theta), |\theta| \leq \pi/4$, a a parameter.[2]


Lemniscate

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Lemniscate in Polar Coordinates
%%BoundingBox: -15 -6 15 6
%%BeginSetup
%%EndSetup
/lemniscate{2{gsave -45 rotate /asqrt2 14.142 def  0 0  moveto
-84 6 84{/2t exch def 3 rotate asqrt2 2t cos sqrt mul 0 lineto}for
0 0 lineto stroke grestore -1 1 scale}repeat%left branch mirrored
}bind def
```

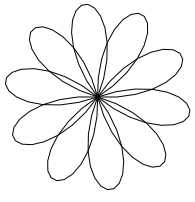
In my 'Julia fractals'-paper, EuroTEX2012 I showed that the Cardioid could also have been drawn by Mathematica via `ParametricPlot`. Mathematica has plot variants: `Pot`, `PolarPlot`, `ParametricPlot`, `Plot3D`, `PlotContour`, ...
In Mathematica the Lemniscate can be plotted by `PolarPlot[2 Sqrt[Cos[2t]], {t,0,}$\pi$\type{/4}]`.[3]
My MF code for the Lemniscate of the mid-90s is longer.
In the Archimedes and the Growth spiral the next radius vector is obtained by only one addition casu quo one multiplication.

Stylistic and trigonometric stylistic flower I consider Stylistic and trigonometric stylistic flowers nice examples of using PS' US rotation. I have overlooked the influence `sin` has on the radius vector in Polar Coordinates. At right the picture with $\sin(6\theta)$. I have decoupled the number 6 from the number of petals.


Stylistic flower

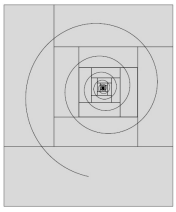
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Stylistic flower
%%BoundingBox: -26 -26 26 26
%%BeginSetup
%%EndSetup
/flower{/r 18 def
10 {0 r r 270 360 arc
    r 0 r  90 180 arc
    36 rotate} bind
    repeat stroke
} bind def
```

```
%!PS-Adobe-3.0 EPSF-3.0
0 0 moveto /dtheta 2 def /a 100 def%scaling
10{gsave%one leaf
  dtheta dtheta 30{%points per leaf;
  /6theta exch 6 mul def%30*6=180
  a 6theta sin mul 0 lineto
  dtheta 2 mul rotate%2 thickness leaf
  }for stroke
  grestore
  36 rotate%number of leaves 36*10=360
  }repeat showpage
```


Stylistic trig. flower

For Lauwerier's BASIC code of an intriguing trigonometric stylistic flower, and my PS translation, see Appendix 2. Rotation symmetric Julia fractals may also yield stylistic flowers, see Julia Fractals in PS, EuroTEX2012. Stylistic flowers enriched by colours have been shown in my 'Recreational use of TEX', EuroTEX2012.

Introduction


TEX-Gyre-logo

While catching up with font developments in the (TEX) world in the 21^{st} century, I learned about the OTF TEX-Gyre-project for fonts to be used by AllTEX engines. The TEX-Gyre-logo drew my attention, especially the spiral.
In a comment on my 'Recreational use of TEX&Co'-paper Jackowski reminded me of the Toruń1998-logo, which displays a belt with text on a sphere.
Enough for paying attention to Spirals in PostScript.


Toruń1998-logo

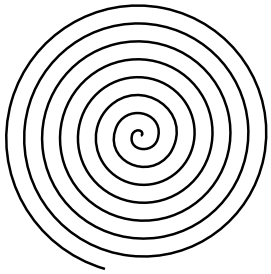
Archimedes and Growth Spiral

The Archimedes Spiral, in polar coordinates, is defined by $r_\theta = k\theta, \quad 0 \le \theta < \infty$.

The Growth Spiral is in polar coordinates defined by $\ln r_\theta = k\theta$ or $r_\theta = \mathrm{e}^{k\theta}$, $-\infty < \theta < \infty$.

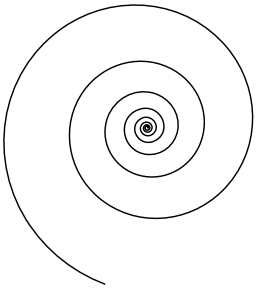
If we draw the curves by piecewise straight lines connecting the $\{\theta_i, r_i\}_{i=1}^n$, with θ_i equidistant, then we have the following properties for the discrete radius vectors $\{r_i\}_{i=1}^n$.[4]

Archimedes Spiral: $r_{i+1} = r_i + k\Delta\theta$, $i = 1, ...\infty$, an arithmetic series. The spiral for finite θ_{max} is finite. The grooves are equidistant.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Archimedes Spiral
%%BoundingBox: -56 -59 54 55
%%BeginSetup
%%EndSetup
/archimedesspiral{0 0 moveto
/f .1 def /r 0 def
555{5 rotate /r r f add def
    r 0 lineto}repeat stroke
}bind def
%
archimedesspiral
```

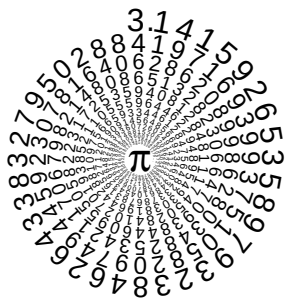
Growth Spiral: $r_{i+1} = r_i\ \mathrm{e}^{k\Delta\theta}$, $-\infty < i < \infty$, a geometric series. The spiral for finite θ_{max} is infinite. Rotation yields the same spiral; the spiral is self-similar.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth Spiral
%%BoundingBox: -100 -110 75 86
%%BeginSetup
%%EndSetup
/growthspiral{1 0 moveto%off the 0
/f 2.718 .0085 exp def /r 1 def
555{5 rotate /r r f mul def
    r 0 lineto}repeat stroke
}bind def
%
growthspiral
```

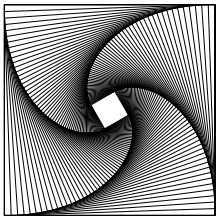
Appendix 1 contains Lauwerier's BASIC programs for these spirals using Cartesian coordinates.

My most beautiful Growth spiral is the π-decimals picture. The decimals are printed along an implicit growth spiral because of the scaling and the rotation of User Space. The disk of radius 1 has surface π.



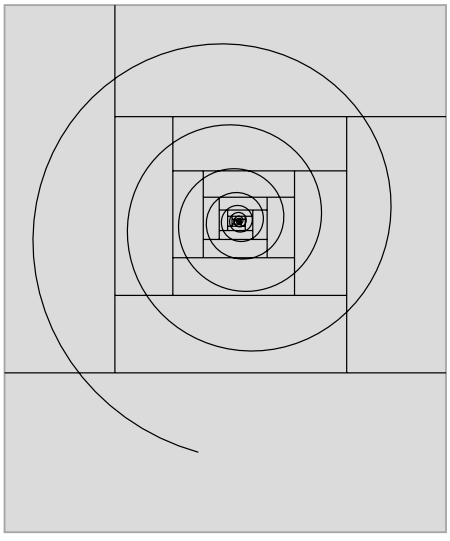
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Pi-decimals along a  Spiral, cgl 2012
%%BoundingBox: -80 -100 100 90
%%BeginSetup
%%EndSetup
/Symbol 26 selectfont
1 -18 moveto (p) show%p denotes pi in symbol font
/Helvetica 20 selectfont
0 70 moveto (3) show 1 0 rmoveto (.) show -2 0 rmoveto
-10 rotate .995 dup scale
{pop pop -10 rotate 3 0 rmoveto .995 dup scale}
(3.14159265358979323846264338327950288419716939937\
 ...) kshow showpage
```

More nice Growth spirals appear in the shrinking squares picture.



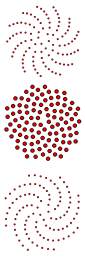
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Wervel (HA Lauwerier)
/wervel{/square {-1 s -1 s 2 s  2 s rectstroke} def
 /s {100 mul} def
 /b 12 6.2831 div def
 /c 1 b sin b cos add div def
 64{square b rotate c c scale}repeat
}bind def
```

The Gyre-logo appears to contain a growth spiral. My imitation of the Gyre-logo, with the recursive lattice[5] underneath and with the growth spiral,[6] has been programmed in PostScript as follows.



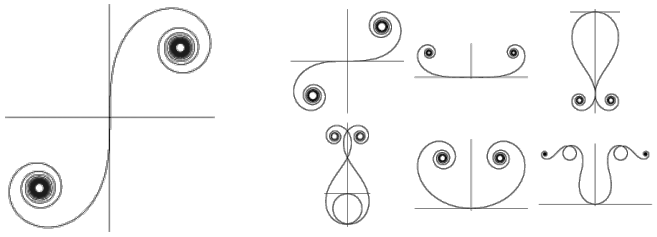
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Imitation Gyre-logo, cgl okt 2012
%%BoundingBox: -2 -2 722 862
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run % contains growthspiral
%%EndProlog
/ux 720 def /uy 860 def
gsave .86 setgray 0 0 ux uy rectfill
  3 setlinewidth .65 setgray 0 0 ux uy rectstroke%background
grestore
2 setlinewidth
/xl{ux 180 720 div mul}def /xu{ux xl .9 mul sub}def
/yl{uy 260 860 div mul}def /yu{uy yl .7 mul sub}def
7{xl yl moveto xl uy lineto 0 yl moveto  ux yl lineto %low
  xu yl moveto xu yu lineto xl yu moveto ux yu lineto %up
  stroke
  xl yl translate /ux xu xl sub def /uy yu yl sub def
}repeat
%growth spiral
ux uy translate growthspiral showpage
%%EOF
```

Spirals occur naturally in Nautilus shells, sunflower heads, the arms of spiral galaxies, and abundantly in fractals.

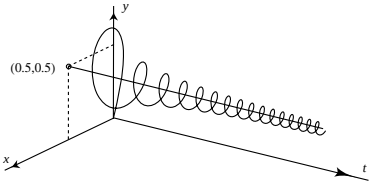


The Nautilus is overlaid with a rectangle with golden-ratio proportions. The rectangle is covered with squares, each time the largest possible. In each square a circular arc has been drawn, yielding the Nautilus 'spiral.'

Cornu's spiral $c(t) = u(t) + iv(t)$, in $\mathbb{C}$, with u and v Fresnel integrals, is famous in Optics for the description of the diffraction of a half-plane. It is also known as a clothoid or Euler's spiral. Advanced, interesting, nice and beautiful.
http://www.mathworld.wolfram/CornuSpiral.html.



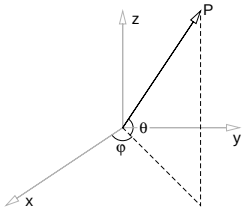
N.M. Temme communicated a 3D Cornu's spiral from
Temme, N.M(1996): Special Functions An introduction to Classical Functions of MathematicaL Physics. John Wiley.



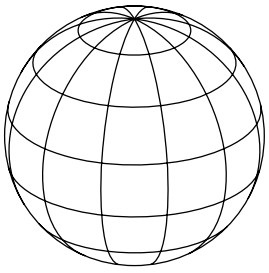
Cornu's spiral
formed from Fresnel integrals
$\{C(t), S(t), t\},\ t \geq 0.$

The Cornu spiral is the projection of the cork-screw in the (x,y)-plane. At any point of the Cornu spiral the quotient of the curvature and the length equals π, Temme(1996, p185)!

Spherical coordinates



Next to the right-handed XYZ Cartesian coordinate system there is the spherical coordinates system with (r, ϕ, θ), where r is the radius vector, ϕ the azimuth angle and θ the latitude angle. In literature the complement of θ, the angle between the polar axis and the radius vector is also used and called polar angle.
We'll draw the sphere with meridians and latitude circles, the sphere with the spherical growth spiral, and the sphere with belts à la Escher and à la the Toruń-logo, but with text.



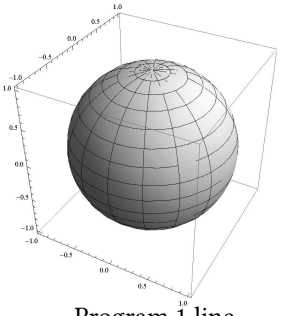
Program 50 lines
.pdf 6kB

The PostScript codes for the spheres with meridians and latitude circles, ≈ 50 lines — translations of Lauwerier H. A.(1987): Meetkunde met de microcomputer, p110-111 — are too lengthy and too boring to be included here, but ... useful. Consult `PSlib.eps`.

← In PS hidden lines suppressed By Mathematica→

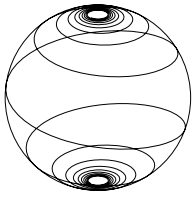
```
ParametricPlot3D[{Cos[t]Cos[u], Sin[t]Cos[u],
Sin[u]}, {t, 0, 2Pi}, {u,-Pi/2,Pi/2}].
```

Using Mathematica is simpler and yields more than programming in PS, at expense of a huge `.pdf` file, which makes previewing slow.



Program 1 line
.pdf 4.5mB, .png 53kB
`Graphics3D[Sphere[]]` 1.2mB

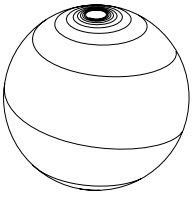
Spherical spiral



The PostScript code is a translation of `bolspira`, Lauwerier H. A(1987): Fractals, p150.

← spherical growth spiral, hidden lines in view
spherical growth spiral, hidden lines suppressed→

http://mathworld.wolfram.com/topics/SphericalCurves.html



What spiral has been used by Escher and in the Toruń-logo? Apparently not the following logarithmic spherical spiral, because of the ∞ number of windings near the poles.
Below is Lauwerier's BASIC program, and my PostScript translation. Parameters: r = radius sphere, a = density of windings , s=$k\pi/50$, k=-500(1)500, P(x,y,x)=P(r, ϕ, θ) point on the sphere, with ϕ = azimuth, $\theta_k = \text{atan}(a\, s_k)$ = inclination.

```
10 REM ***BOLSPIRAAL, S=phi***
50 A=.15 : REM ***spiraal constante***
60 C=.9  : REM ***helling (uv-) projectievlak***
70 P=1/SQR(2) : Q=P*SQR(1-C*C) : REM ***PROJECTIE
80 FOR N=-500 TO 500             REM    GETALLEN***
90 S=N*PI/50 : T=ATN(A*S)
100 X=COS(S)COS(T) : Y=SIN(S)COS(T) : Z=-SIN(T)
110 U=P*(Y-X) : V=C*Z-Q*(X+Y)
120 IF N=-500 THEN PSET(U,V) ELSE LINE -(U,V)
130 NEXT N : END
```

P(x,y,z)=P(r, ϕ, θ)

$$\begin{aligned} x &= r\cos\phi\cos\theta \\ y &= r\sin\phi\cos\theta \\ z &= r\sin\theta \end{aligned}$$

coordinates projection plane

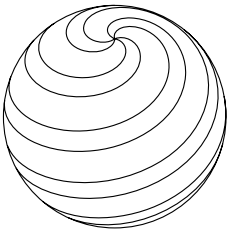
$$\begin{aligned} u &= (-x+y)/\sqrt{2} \\ v &= -(x+y)/\sqrt{2}\sqrt{(1-c^2)} + cz \\ w &= c(x+y)/\sqrt{2} + z\sqrt{(1-c^2)} \quad \perp uv-plane \end{aligned}$$

```
/sphereandspiraldict 25 dict def
/sphereandspiral{sphereandspiraldict begin /r 100 def
/c .9 def /a2 1 1.41421 div def
  /b1 a2 1 c c mul sub sqrt mul def
  /c1 c a2 mul def /c3 1 c c mul sub sqrt def
/spiral{90 -1 -90{/thetaj exch def
 /phij thetaj 10 mul phi0 add def%windings
 /x r phij cos thetaj cos mul mul def
 /y r phij sin thetaj cos mul mul def
 /z r thetaj sin mul def
 /w c1 x y add mul c3 z mul add def
 /u a2 y x sub mul def
 /v c z mul b1 x y add mul sub def
 w 0 lt{u v moveto}%hidden lines suppressed
       {u v lineto}ifelse
}for %thetaj
}bind def%spiral
end} bind def
```

Curve on a sphere

I decided to construct a spiral-like curve on the sphere. Below are given: the relation between a spacial point (ϕ, θ) on a sphere of radius r, the coordinates (u,v) in the projection plane and w the coordinate orthogonal to the projection plane.

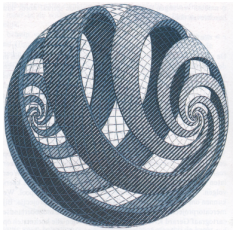
$$\begin{aligned} x &= r\cos\phi\cos\theta \\ y &= r\sin\phi\cos\theta \\ z &= r\sin\theta \end{aligned} \quad \xrightarrow{\text{projection}} \quad \begin{aligned} u &= (-x+y)/\sqrt{2} \\ v &= -(x+y)/\sqrt{2}\sqrt{(1-c^2)} + cz \\ w &= c(x+y)/\sqrt{2} + z\sqrt{(1-c^2)} \end{aligned}$$



The left sphere has three spiral-like belts with winding constant $k = 4$; parameterized in $t \in [-\pi/2, \pi/2]$ with coordinates

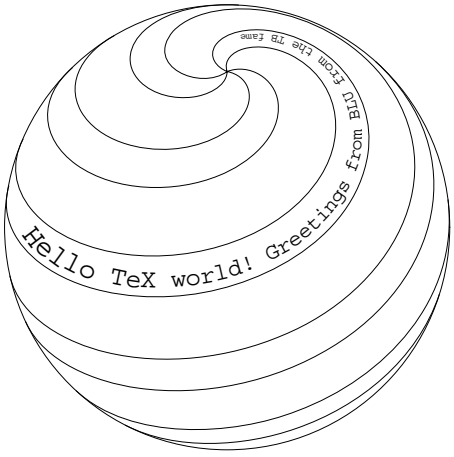
$$\begin{aligned} x(t) &= r\cos 4t\cos t \\ y(t) &= r\sin 4t\cos t \\ z(t) &= r\sin t \end{aligned}$$

Escher spherical spirals →



Text on a sphere

I followed the Toruń logo partly by just printing a line of text in the visible part of a belt, by the use of Adobe's `pathtext` from the Bluebook; no animation nor manipulation of the hidden texts. Too many details.




```
/Courier 14 selectfont
/sphereandspiraldict 25 dict def
/sphereandspiral{sphereandspiraldict begin /r 100 def
/c .7 def /a2 1 1.41421 div def /b1 a2 1 c c mul sub sqrt mul def
/c1 c a2 mul def /c3 1 c c mul sub sqrt def
/spiral{-90 1 +90{/thetaj exch def
 /phij thetaj 4 mul phi0 add def%windings
 /x r phij cos thetaj cos mul mul def
 /y r phij sin thetaj cos mul mul def
 /z r thetaj sin mul def
 /w c1 x y add mul c3 z mul add def
 /u a2 y x sub mul def
 /v c z mul b1 x y add mul sub def
 w 0 lt{u v moveto}{u v lineto}ifelse
}for %thetaj
}def%spiral
%
gsave%belts
0 c r mul moveto /phi0   0 def spiral
0 c r mul moveto /phi0  45 def spiral stroke
grestore gsave
0 c r mul moveto /phi0 120 def spiral
0 c r mul moveto /phi0 165 def spiral stroke
grestore gsave
0 c r mul moveto /phi0 240 def spiral stroke
0 c r mul moveto /phi0 285 def spiral stroke
0 c r mul moveto /phi0 275 def spiral
(Hello TeX world, greetings from BLU from the TB fame) 25 pathtext
grestore %circle in projection plane
r 0 moveto
1 1 100{/k exch def  /t k 180 mul 50 div def
        r t cos mul r t sin mul lineto
       }for stroke
end} bind def
```

Shrinking the textsize was achieved by inserting `/linetoproc` or `/pathtext` at appropriate places

```
/Courier 14 selectfont /size 14 def
/size size .08 sub def
/Courier findfont [size 0 0 size 0 0] makefont setfont
```

For my spiral-like spherical curves I discretized $\theta_j, j = 90, 89, ...0, ... - 89, -90$ and at the same time I chose for $\phi_j = k\theta_j$ with k a winding constant, the pitch.
If we connect the points $\{\theta_j, \phi_j\}_{j=90}^{-90}$ by straight lines in the projection plane, we obtain the impression of a curve on the sphere. The North and South Poles are $(0, 0, \pm r)$ in Cartesian coordinates. Shifting the curve by rotating the starting position yields a belt on the sphere.

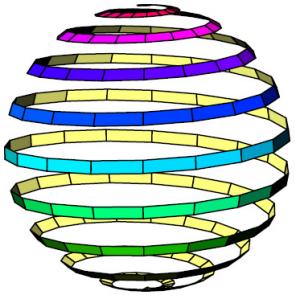
Discussion I found Jackowski's MetaPost code[<reference>] hard to understand.

The various parts are not clearly marked. Moreover, he mentions that he had to kludge around to circumvent limitations, undoubtedly in the parts I have omitted. In PostScript it is straight sailing for the in-principle picture. Jacko's texts are rigid, the characters are fixed in a picture, but the picture shrinks near the poles.

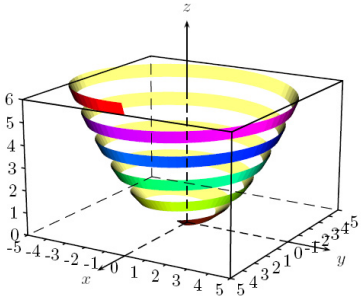
My text is not perfectly typeset either, but it shrinks in size while approaching the poles. Adaptation of textsize is necessary. Hopefully my contribution is also a step forward in the right direction. Much details have to be solved still, maybe for a generation or two after me, with better tools and feedback on the position on the sphere. We could of course inverse (u, v, w) for each character position and obtain the latitude.

Notes Modification of the textsize on the fly is much simpler in PostScript than in TEX&Metafont, although ConTEXt undoubtedly generates fonts on-the-fly.
Experienced readers will discover how to draw the (mirrored) invisible texts, as well as how to manipulate the invisible paths. Animation can be done as well, but requires adjusting of the program.

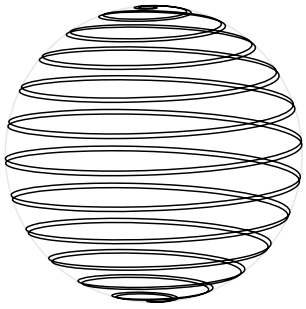
Herbert Voss I am happy to include Herbert Voss's ribbon.tex; which contains the following ribbon pictures. A wink to LATEX-PSTricks users. The process-flow is Script $\xrightarrow{\text{TEX}}$ DVI $\xrightarrow{\text{DVIPS}}$ PS $\xrightarrow{\text{Distiller}}$ PDF. The intermediate `.ps` is 18000 lines, 600kB. I don't know the size of `pst-rubans`.[7]



```
\documentclass{article}
\usepackage{pst-rubans}
\begin{document}
\psset{unit=0.75cm}
\begin{pspicture}(-5,-5)(5,5)
\psset{viewpoint=50 20 10,Decran=50,resolution=180}
\psSphericalSpiral[incolor=yellow!50,R=4,hue=0 1,
                   lightsrc=30 5 17]%
\end{pspicture}

\psset{lightsrc=40 25 17}
\psset{unit=0.75,viewpoint=50 30 20,Decran=50}
\begin{pspicture}(-7,-3)(7,8)
\psSpiralParaboloid[incolor=yellow!50,h=6,hue=0 1,
   resolution=360,spires=5,grid,dZ=0.5]
\gridIIID[QZ=3,Zmin=0,Zmax=6](-5,5)(-5,5)
\end{pspicture}
\end{document}
```



My imitation of Voss' ribbon was achieved by varying the pitch fromfrom 4 into 25, increasing the discretization in θ, reducing the number of belts to one, making the projected circumference gray and changing the viewing angle into `/c1 .99 def` in the belts-on-the-sphere PS code.



A spherical ribbon as variation of the belts-on-a-sphere. If you want to color it or so, the code is included in `PSlib.eps`

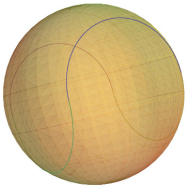
Escher's 'spherical ribbon' →

Tennisball or baseball curve

Different from the spherical spiral but well-known is the curve on the baseball that divides the surface into 2 equal parts.[8] Gabo used the curve for his Spherical object.[9] The four parts of the tennis-ball curve are given by the formulas parameterized in t

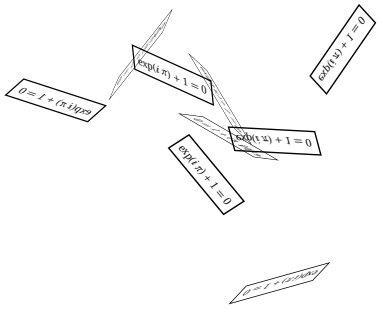
$$\{(x,y,z) \mid (1, \sin t, \cos t)\}$$
$$\{(x,y,z) \mid (-1, \sin t, \cos t)\}$$
$$\{(x,y,z) \mid (-\sin t, 1, -\cos t)\}$$
$$\{(x,y,z) \mid (-1, -\sin t, -\cos t)\}$$

$$t \in (-\frac{\pi}{2}, \frac{\pi}{2})$$



From Wolfram Mathworld Placing Text Randomly.

```
Graphics[Table[
 Inset[Style[Framed[HoldForm[Exp[I Pi] + 1 == 0]], 11],
  RandomReal[{-1, 1}, 2], Center, Automatic,
  RandomReal[{-1, 1}, {2, 2}]], {9}]]
```



Mathematica, basically a formula-manipulation program. It allows users to arbitrarily place text strings and annotate graphics with text labels. The notebooks can also be formatted, no experience as yet. However ... text in a spiral belt on a sphere is decades away, I guess.

However ...

Similar to how I post-processed a symbolic flowers, my wife, Svetlana Morozova, used Photoshop to inscribe a sphere with 'Hello TEX world' to create a nice X-mas card.

Acknowledgements

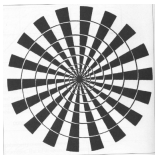
Thank you Adobe for your maintained, adapted to LanguageLevel 3 since 1997, good old, industrial standard PostScript and `Acrobat Pro` (actually DISTILLER) to view it, as well as `pathtext`, for typesetting text along an arbitrary path. Don Knuth for your stable plain TEX, Jonathan Kew for the TEXworks IDE, Hàn Thế́ Thành for pdf(La)TEX,

Thank you Bogusław Jackowski for supplying me with artistic material from GUST, especially sending me the Toruń-logo.

Thank you Herbert Voss for your `ribbon-on-sphere` pictures made by LATEX-PSTricks.

Thank you Nice Temme for your 3D Cornu's spiral.[10]

Thank you Wim W. Wilhelm, Jos Winnink and Henk Jansen for suggestions and proofing, MAPS editors for improving my use of English and last but not least Taco Hoekwater for procrusting my plain TEX preprint note into MAPS format.



Spiral or ...
Peitgen c.s(2004):
Chaos and Fractals

Conclusion

Drawing curves specified in Polar Coordinates works because PostScriptrotation occurs in User Space. The Gyre-logo has been imitated in PS. Belts of text have been drawn on a sphere. The result is less advanced then that of the Toruń-logo; although the effect of making the textsize dependent on the latitude is nice. A ribbon inscribed on a sphere made a nice X-mas card.

My case rest, have fun and all the best.

Appendix 1: Lauwerier's BASIC codes for the Archimedes and Growth spiral

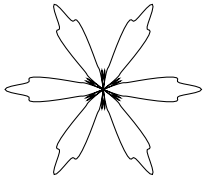
In Lauwerier(1987): Fractals—meetkundige figuren in eindeloze herhaling, Aramith,[11] the Archimedes spiral and the growth spiral have been coded in BASIC, via the usual Polar and Cartesian coordinates.

```
10 REM ***Spiraal van Archimedes***
50 A=.1 : PI=3.141593 : PSET (0,0)
60 FOR T=0 T0 16*PI STEP .1: R=A*T
70    X=R*COS(T) : Y=R*SIN(T) : LINE -(X,Y)
90 NEXT T : END
```

```
10 REM ***Logarithmische Spiraal***
50 A=.1 : B=.1 :  PSET(A,0)
60 FOR T=0.1 T0 35 STEP .1: R=A*EXP(B*T)
70    X=R*COS(T) : Y=R*SIN(T) : LINE -(X,Y)
90 NEXT T : END
```

Appendix 2: Lauwerier's BASIC code

for the trigonometric stylistic flower



```
 A1=1.5 : A2=.2 : PI=3.141593
S1=6 : S2=36 `parameters
T=0 : H=.01  `initialization
DO WHILE T<2*PI
   r=1+A1*COS(S1*T)+A2*COS(S2*T)
   X=R*COS(T) : Y=R*SIN(T)
   IF T=0 THEN PSET(X,Y) ELSE LINE-(X,Y)
   T=T+H
LOOP END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Stylistic flower
%%Formula: r=1+a1*cos(s1*theta)+a1*cos(s2*theta)
%%BoundingBox: -101 -91 101 91
%%BeginSetup
%%EndSetup
/a 35 def%scaling
/a1 1.5 def /a2 .2 def%parameters
/s1 6 def /s2 36 def
/t 0 def /h 1 def %initialization
{ t 360 gt {exit}if
 /r 1 a1 s1 t mul cos mul add
      a2 s2 t mul cos mul add a mul def
 r t cos mul  r t sin  mul t 0 eq{moveto}{lineto}ifelse
 /t t h add def
 }loop closepath
stroke showpage
%%EOF
```

At right my PS translation. Lauwerier, H.A(1992): Computer Simulaties — De wereld als model. Aramith. Lauwerier calls the centre filaments.

Notes

1. During my education I was taught that details matter.
2. In The 8th March note, I drew the Lemniscate in Cartesian coordinates, because the path was reused for printing text along.
3. Abel, M.L, Braselton, J.P, Braselton, L.M(2002): Introduction to Mathematica, p92. Other interesting introductions to Mathematica http://math.georgiasouthern .edu/math/computer/ABGMATHs1., or http://www.science.uva.nl /onderwijs/lesmateriaal/Mathematica/ of 2004 or `Wolfram Mathematica Tutorial Collection` from 2008 onward. It is amazing how much calculus intelligence has been programmed into Mathematica.
4. Questions about the tangent, the arc length, and the curvature stimulated the development of calculus. For formulas see Wolfgram's Mathworld.
5. The lattice can be understood as: draw the 'cross' within the rectangle and repeat the process within the rectangle of the 'cross'.
6. The centre of the growth spiral is given by the coordinates of the last, small, inner rectangle.
7. Processing `TeXnicCentre` yielded that `pst-rubans` was not known, apparently it is not on the TEXlive DVD.
8. Thompson, R.B(1998): Designing a baseball cover. College Math J.
9. Gabo's Torsion, MAPS 42,69–110.
10. Nico used Lauwerier's BASIC 3D drawing program, translated it into PASCAL and obtained `.eps` from there.
11. Translated into Englisch ISBN: 0-691-8551-x.

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

SciTE

Introduction

The SciTE editor is now some 15 years old, and still one of the nicest around. This editor is a wrapper around the scintilla editor framework. It is available for free for Windows and Linux, and there is a relatively cheap version for MacOSX.

Here are a few reasons why I prefer using this editor:

- The footprint is small and (at least on Windows and Linux) installation and updating is easy.
- The editor starts up fast, performs well and the fonts have always been rendered very well.
- Configuration is easy and flexible, and changes to configuration files are reflected immediately (no restart required).
- Lots of file formats are supported by syntax highlighting.
- Processing of files is logged in a fast and efficient log pane.
- There is a Lua interface built-in that permits you to write extensions.

Syntax highlighting

One of the first things that I did when I started using SciTE is to (re)write the TEX and MetaPost lexers so that they were more suitable for ConTEXt. For quite some years ConTEXt has shipped with the relevant files for editing and processing TEX.

Recently the external lexer feature has been extended by an Lpeg based lexer. As a consequence, I wrote a couple of lexers that go beyond the older ones. These are tuned for ConTEXt and are shipped with the ConTEXt distribution.

- The TEX lexer can not only distinguish between tex primitives (including conditionals), low level ConTEXt commands, special registers, ConTEXt user interface commands and special symbols, but can also deal with nested Lua and MetaPost code.
- The Lua lexer has a variant that can recognize some of the ConTEXt MkIV features.
- The xml lexer can recognize some errors in the syntax.
- The Pdf lexer can recognize the relevant objects to some extent.
- The MetaPost lexer can distinguish primitives, MetaFun and user defined commands.

The ConTEXt distribution ships with all the files needed for getting this up and running. In addition to initializing lexers, we also tune some of the menus for use with TEX based workflows. The background of the text areas is set to a light shade of gray and the font defaults to Dejavu Mono; which happens to cover lots of Unicode characters.

The TEX, xml and the yet unmentioned text lexers can do realtime spell checking. As with the lexers, spell checking is more advanced in the Lpeg lexers than in the traditional ones: we recognize rightly spelled words, mark unknown words and also mark words that need case checking. The nice thing is that the regular command highlighting works in parallel. This is shown in figure 1. Normally I use the full (high-res and wide) screen which gives enough room for regular documents as well as the (real-time) log pane. Menus and configured tools adapt themselves automatically to the current file type.

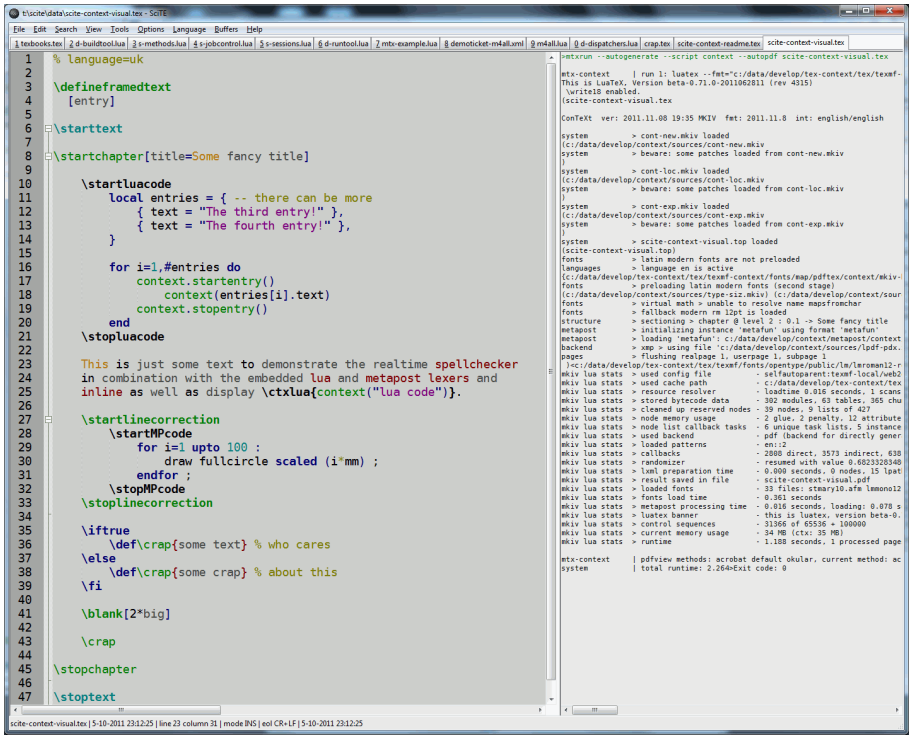


Figure 1. Nested lexers in action.

A side note: currently the MacOSX version does not ship with the library needed for the lpeg lexer. Hopefully this will change one day.

Functionality

Of course, SciTE provides all the regular editing functionality; although in practice one will use only a small subset of features. A real nice feature is the rectangular selection, cut and paste. It's really easy to move columns around. Spacing can be visualized. Structures can be folded.

As an example of a Lua extensions I wrote a word wrapper, simply because I want document sources to look nice as well.

Many files can be open at the same time and are accessed using tabs. The state is remembered and restored at a next startup. Quite handy is the fact that one can collapse all start–ups into one instance.

Word completion is supported (which is quite handy) as is expansion of abbreviation (although I never used that). Also nice is the ability to comment a line or a selected block of text with one key.

Of course, once you are fluent with an editor it stays your favourite no matter what others tell you, but for me it's the only editor that I want to work with.

Side note: a nice alternative is `textadept`, an editor written in Lua using the same scintilla editing component. It shares the Lpeg lexing code but it lacks a realtime log pane and has no tabs.

Processing

Users can easily tweak the `.properties` files to define commands that can be applied to a file. One can add runners to menus and associate them with keys. For TEX compiling, linking et cetera this makes no sense, but checking, processing and

previewing does. For ConTEXt we use `(mtx-)check` for checking, `(mtx-)context` for processing TEX and xml files, `mtxrun` for running Lua, et cetera.

Of course, you can use SciTE for any macro package you like. If you don't use ConTEXt, it provides support for TEX anyway. It's an easy to install editor, so if you're looking for something new it's worth a try.

Hans Hagen

Lua in MetaPost

Introduction

For some years I have been wondering if we could escape to Lua inside MetaPost, or, in practice, to mplib in LuaTEX. The idea is simple: embed Lua code in a MetaPost file that gets run as soon as it's seen. In case you wonder why Lua code makes sense, imagine generating graphics using external data. The capabilities of Lua to deal with that is more flexible and advanced than in MetaPost. Of course we could generate a MetaPost definition of a graphic from data, but it often makes more sense to do the reverse. I finally found time and a reason to look into this, and in the following sections I will describe how it's done.

The basics

The approach is comparable to LuaTEX's `\directlua`. That primitive can be used to execute Lua code, and in combination with `tex.print` we can pipe strings back into the TEX input stream. There, there is a complication in that we have to be able to operate under different so-called catcode regimes: the meaning of characters can differ per regime. We also have to deal with line endings in special ways as they relate to paragraphs and such. MetaPost doesn't have that complication; being perfectly happy with simple strings. For putting back input into MetaPost a mechanism similar to `scantokens` can be used. That way we can return anything (including nothing) as long as MetaPost can interpret it and as long as it fulfils the expectations.

```
numeric n ;
n := scantokens("123.456") ;
```

A script is run as follows:

```
numeric n ;
n := runscript("return '123.456'") ;
```

This primitive doesn't have the word `lua` in its name so, in principle, any wrapper around the library can use it as a hook. In the case of LuaTEX the script language is of course Lua. At the MetaPost end we only expect a string. How that string is constructed is completely up to the Lua script. In fact, the user is completely free to implement the runner any way she or he wants, like:

```
local function scriptrunner(code)
  local f = loadstring(code)
  if f then
    return tostring(f())
  else
    return ""
  end
end
```

This is hooked into an instance as follows:

```
local m = mplib.new {
  ...
  run_script = scriptrunner,
  ...
}
```

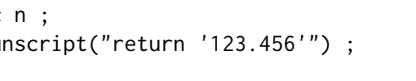
Now, beware, this is not the ConTEXt way. We provide print functions and other helpers, which we will explain in the next section.

Helpers

After I got this feature up and running I played a bit with possible interfaces at the ConTEXt (read: MetaFun) end and ended up with a bit more advanced runner where no return value is used. The runner is wrapped in the `lua` macro.

```
numeric n ;
n := lua("mp.print(12.34567)") ;
draw textext(n) xsized 4cm withcolor maincolor  ;
```

This renders as:



In case you wonder how efficient calling Lua is, don't worry; it's fast enough, especially if you consider suboptimal Lua code and the fact that we switch between machineries.

```
draw image (
  lua("statistics.starttiming()") ;
  for i=1 upto 5000 :
    draw lua ("mp.pair
      (math.random(-74,126),
        math.random(-35,35))" ) ;
  endfor ;
  setbounds currentpicture to
    fullsquare xyscaled (100,20) ;
  lua("statistics.stoptiming()") ;
  draw textext(lua
    ("mp.print(
      statistics.elapsedtime())")
    ) ysized 40 ;
) withcolor maincolor
  withpen pencircle scaled 1 ;
```

Here the part:

```
draw lua ("mp.pair
  (math.random(-74,126),
    math.random(-35,35))" ) ;
```

effectively becomes (for instance):

```
draw scantokens "(25,15)" ;
```

which in turn becomes:

```
draw scantokens (25,15) ;
```

The same happens with this:

```
draw textext (lua
  ("mp.print
     (statistics.elapsedtime())"
   ) ) ...
```

This becomes for instance:

```
draw textext(scantokens "1.23") ...
```

and therefore:

```
draw textext(1.23) ...
```

We can use `mp.print` here because the `textext` macro can deal with numbers. The following also works:

```
draw textext(lua
   ("mp.quoted
    (statistics.elapsedtime())"
   )
) ...
```

Now we get (in MetaPost speak):

```
draw textext(scantokens
  (ditto & "1.23" & ditto) ...
```

Here `ditto` represents the double quotes that mark a string. Of course, because we pass the strings directly to `scantokens`, there are no outer quotes at all, but this is how it can be simulated. In the end we have:

```
draw textext("1.23") ...
```

The decision to use `mp.print` or `mp.quoted` depends on what the expected return value is; an assignment to a numeric can best be a number or an expression resulting in a number.

This graphic becomes:

The runtime on my current machine is some 0.25 seconds without and 0.12 seconds with caching. But to be honest, speed is not really a concern here as the amount of complex MetaPost graphics can be neglected compared to extensive node list manipulation. With LuajitTEX generating the graphic takes 15% less time.[1]

The three print commands accumulate their arguments:

```
numeric n ;
n := lua("mp.print(1) mp.print('+') mp.print(2)")
;
draw textext(n) xsized 1cm
  withcolor maincolor  ;
```

As expected we get:

Equally valid is:

1. Processing a small 8 page document like this takes about one second, which includes loading a bunch of fonts.

```
numeric n ;
n := lua("mp.print(1,'+',2)") ;
draw textext(n) xsized 1cm
   withcolor maincolor  ;
```

This gives the same result:

3

Of course all kind of action can happen between the prints. It is also legal to have nothing returned as could be seen in the 10.000 dot example; there the timer related code returns nothing, so effectively we have `scantokens("")`. Another helper is `mp.quoted`, as in:

```
draw textext
  (lua
    ("mp.quoted
      ('@0.3f',
       " & decimal n & "
      )"
    )
  ) withcolor maincolor  ;
```

This typesets 3.000. Note the `@`. When no percent character is found in the format specifier, we assume that an `@` is used instead.

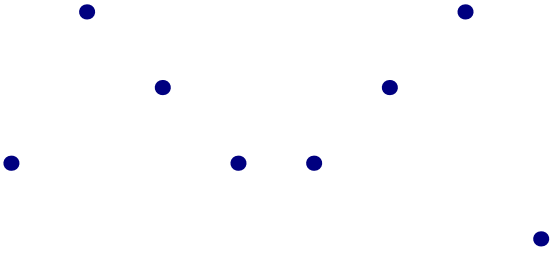
But, the real benefit of embedded Lua is when we deal with data that is stored at the Lua end. First we define a small dataset:

```
\startluacode
table.save("demo-data.lua",
  {
    { 1, 2 }, { 2, 4 }, { 3, 3 },
    { 4, 2 }, { 5, 2 }, { 6, 3 },
    { 7, 4 }, { 8, 1 },
  }
)
\stopluacode
```

There are several ways to deal with this table. I will show clumsy as well as better looking ways.

```
lua("MP = { }
MP.data = table.load('demo-data.lua')"
) ;
numeric n ;
lua("mp.print('n := ',\#MP.data)") ;
for i=1 upto n :
  drawdot
    lua("mp.pair
      (MP.data[" & decimal i & "])"
    ) scaled cm
    withpen pencircle scaled 2mm
    withcolor maincolor  ;
endfor ;
```

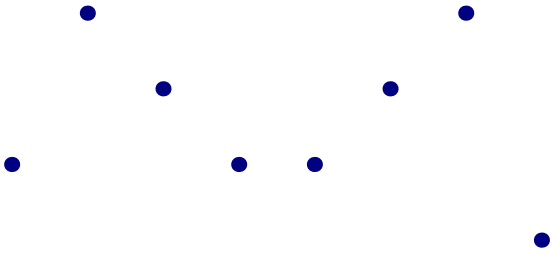
Here we load a Lua table and assign the size to a MetaPost numeric. Next we loop over the table entries and draw the coordinates.



We will stepwise improve this code. In the previous examples we omitted wrapper code but here we show it:

```
\startluacode
  MP.data = table.load('demo-data.lua')
  function MP.n()
    mp.print(#MP.data)
  end
  function MP.dot(i)
    mp.pair(MP.data[i])
  end
\stopluacode

\startMPcode
  numeric n ;
  n := lua("MP.n()") ;
  for i=1 upto n :
    drawdot
      lua("MP.dot
        (" & decimal i & ")"
      ) scaled cm
      withpen pencircle scaled 2mm
      withcolor maincolor  ;
  endfor ;
\stopMPcode
```

So, we create a few helpers in the `MP` table. This table is predefined so, normally, you don't need to define it. You may, however, decide to wipe it clean.



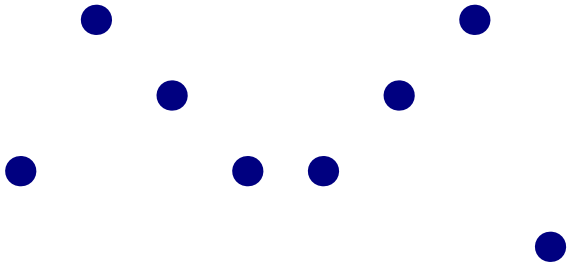
You can decide to hide the data:

```
\startluacode
  local data = { }

  function MP.load(name)
    data = table.load(name)
  end
  function MP.n()
    mp.print(#data)
  end
  function MP.dot(i)
    mp.pair(data[i])
  end
\stopluacode
```

It is possible to use less Lua, for instance in:

```
\startluacode
  local data = { }
  function MP.loaded(name)
    data = table.load(name)
    mp.print(#data)
  end
  function MP.dot(i)
    mp.pair(data[i])
  end
\stopluacode

\startMPcode
  for i=1 upto
    lua
      ("MP.loaded
        ('demo-data.lua')"
      ) :
    drawdot
      lua("MP.dot(",i,")") scaled cm
      withpen pencircle scaled 4mm
      withcolor maincolor  ;
  endfor ;
\stopMPcode
```

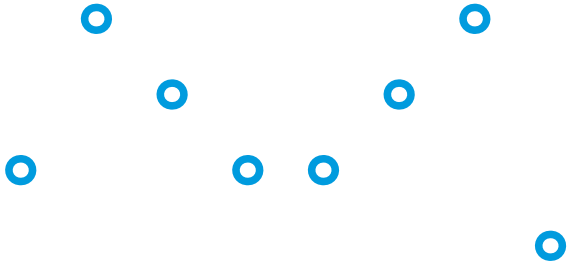
Here we also omit the `decimal` because the `lua` macro is clever enough to recognize it as a number.



By using some MetaPost magic we can even go a step further in readability:

```
\startMPcode{doublefun}
  cmykcolor maincolor;
  maincolor := (1,.15,0,0);
  lua.MP.load("demo-data.lua") ;

  for i=1 upto lua.MP.n() :
    drawdot lua.MP.dot(i) scaled cm
      withpen pencircle scaled 4mm
      withcolor maincolor  ;
  endfor ;

  for i=1 upto MP.n() :
    drawdot MP.dot(i) scaled cm
      withpen pencircle scaled 2mm
      withcolor white ;
  endfor ;
\stopMPcode
```

Here we demonstrate that it also works in `double` mode; which makes sense when processing data from other sources. Note how we omit the `lua.` prefix: the `MP` macro will deal with that.



So in the end we can simplify the code that we started with to:

```
\startMPcode{doublefun}
  for i=1 upto
    MP.loaded("demo-data.lua") :
    drawdot
      MP.dot(i) scaled cm
```

```
      withpen pencircle scaled 2mm
      withcolor maincolor  ;
  endfor ;
\stopMPcode
```

Access to variables

The question with such mechanisms is always: how far should we go. Although MetaPost is a macro language, it has properties of procedural languages. It also has more introspective features at the user end. For instance, one can loop over the resulting picture and manipulate it. This means that we don't need full access to MetaPost internals. However, it makes sense to provide access to basic variables: `numeric`, `string` and `boolean`.

```
draw textext(lua
  ("mp.quoted
    ('@0.15f',
      mp.get.numeric('pi')-math.pi
    )"
  )
)
  ysized .5cm
  withcolor maincolor  ;
```

In double mode you will get zero printed but in scaled mode we definitely get a different result:

-0.000006349878856

In the next example we use `mp.quoted` to make sure that we indeed pass a string. The `textext` macro can deal with numbers, but an unquoted `yes` or `no` is asking for problems.

```
boolean b ;
b := true ;
draw textext(
  lua
    ("mp.quoted(mp.get.boolean('b')
     and 'yes' or 'no')"
    )
)
  ysized 1cm
  withcolor maincolor  ;
```

Especially when more text is involved, it makes sense to predefine a helper in the `MP` namespace. Because MetaPost, currently, doesn't like newlines in the middle of a string, a `lua` call has to be on one line.

yes

Here is an example where Lua does something that would be close to impossible, especially if more complex text is involved.

```
string s ;
s := "" ; % ""
draw textext
  (lua
    ("mp.quoted
      (characters.lower
        (mp.get.string('s')
        )
      )"
    )
  )
  ysized 1cm
  withcolor maincolor  ;
```

As you can see here, the whole repertoire of helper functions can be used in a MetaFun definition.

τεχ

The library

In ConTEXt we have a dedicated runner, but for the record we mention the low level constructor:

```
local m = mplib.new {
  ...
  script_runner = function(s) return
    loadstring(s)() end,
  script_error  = function(s)
    print(s) end,
  ...,
}
```

An instance (in this case `m`) has a few extra methods. Instead you can use the helpers in the library.

`m:get_numeric(name)`	returns a numeric (double)
`m:get_boolean(name)`	returns a boolean (`true` or `false`)
`m:get_string (name)`	returns a string
`mplib.get_numeric(m,name)`	returns a numeric (double)
`mplib.get_boolean(m,name)`	returns a boolean (`true` or `false`)
`mplib.get_string (m,name)`	returns a string

In ConTEXt the instances are hidden and wrapped in high level macros, so there you cannot use these

commands.

ConTEXt helpers

The mp namespace provides the following helpers:

print(...)	returns one or more values
pair(x,y) pair(t)	returns a proper pair
triplet(x,y,z) triplet(t)	returns an RGB color
quadruple(w,x,y,z) quadruple(t)	returns an CMYK color
format(fmt,...)	returns a formatted string
quoted(fmt,...) quoted(s)	returns a (formatted) quoted string
path(t[,connect][,close])	returns a connected (closed) path

The mp.get namespace provides the following helpers:

numeric(name)	gets a numeric from MetaPost
boolean(name)	gets a boolean from MetaPost
string(name)	gets a string from MetaPost

Paths

In the meantime we got several questions on the ConTEXt mailing list about turning coordinates into paths. Now imagine that we have this dataset:

```
10 20 20 20 -- sample 1
30 40 40 60
50 10

10 10 20 30 % sample 2
30 50 40 50
50 20

10 20 20 10 # sample 3
30 40 40 20
50 10
```

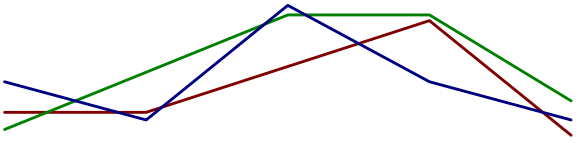
In this case I have put the data in a buffer, so that it can be shown here, as well as used in a demo. Note how we can add comments. The following code converts this into a table with three subtables.

```
\startluacode
  MP.myset =
    mp.dataset
      (buffers.getcontent("dataset"))
\stopluacode
```

We use the MP (user) namespace to store the table. Next we turn these subtables into paths:

```
\startMPcode
  for i=1 upto
    lua("mp.print(mp.n(MP.myset))") :
  draw
    lua("mp.path
      (MP.myset[" & decimal i & "]
      )")
    )
    xysized (HSize,10ExHeight)
    withpen
      pencircle scaled .25ExHeight
    withcolor basiccolors[i]/2 ;
  endfor ;
\stopMPcode
```

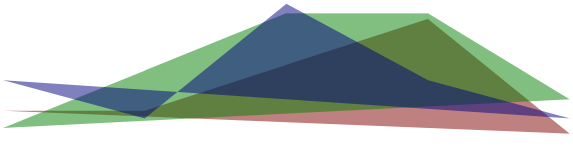
This gives:



Instead, we can fill the path; in which case we also need to close it. The true argument deals with that:

```
\startMPcode
  for i=1 upto
    lua("mp.print(mp.n(MP.myset))") :
  path p ; p :=
    lua("mp.path
      (MP.myset
        [" & decimal i & "],
        true
      )")
    )
    xysized (HSize,10ExHeight) ;
  fill p
    withcolor basiccolors[i]/2
    withtransparency (1,.5) ;
  endfor ;
\stopMPcode
```

We get:

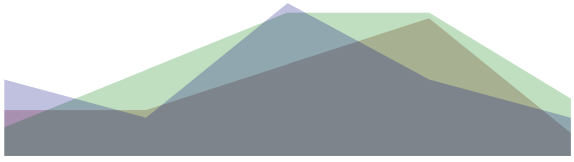


The following makes more sense:

```
\startMPcode
  for i=1 upto
    lua("mp.print
```

```
      (mp.n(MP.myset))"
    ) :
  path p ;
  p := lua("mp.path
    (MP.myset[" & decimal i & "])"
    )
    xysized (HSize,10ExHeight) ;
  p :=
    (xpart llcorner boundingbox p,0)
    -- p --
    (xpart lrcorner boundingbox p,0)
    -- cycle ;
  fill p
    withcolor basiccolors[i]/2
    withtransparency (1,.25) ;
  endfor ;
\stopMPcode
```

So this gives:



This (area) fill is so common, that we have a helper for it:

```
\startMPcode
  for i=1 upto
    lua("mp.size(MP.myset)") :
  fill area
    lua("mp.path
      (MP.myset[" & decimal i & "])"
    )
    xysized (HSize,5ExHeight)
    withcolor basiccolors[i]/2
    withtransparency (2,.25) ;
  endfor ;
\stopMPcode
```

So this gives:

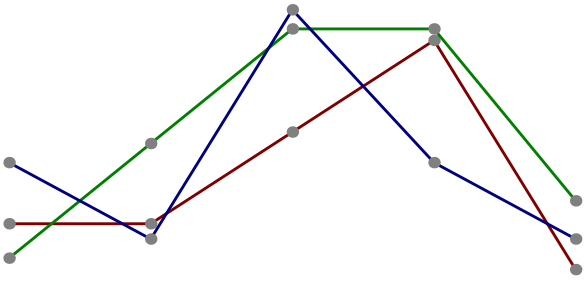
This snippet of MetaPost code still looks kind of horrible, so how can we make it look better? Here is an attempt. First we define a bit more Lua:

```
\startluacode
local data = mp.dataset
    (buffers.getcontent("dataset"))
MP.dataset = {
  Line = function(n) mp.path(data[n]) end,
  Size = function()  mp.size(data)  end,
}
\stopluacode
```

We can now make the MetaPost look more natural. Of course, this is possible because in MetaFun the `lua` macro does some extra work.

```
\startMPcode
  for i=1 upto
    lua.MP.dataset.Size() :
  path p ;
  p := lua.MP.dataset.Line(i)
    xysized (HSize,20ExHeight) ;
  draw
    p
    withpen
      pencircle scaled .25ExHeight
    withcolor basiccolors[i]/2 ;
  drawpoints
    p
    withpen pencircle scaled ExHeight
    withcolor .5white ;
  endfor ;
\stopMPcode
```

As expected, we get the desired result:



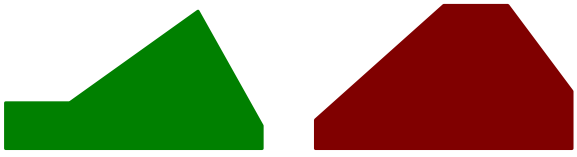
Once we start making things look nicer and more convenient to code, we quickly end up with helpers like those in the next example. First we save some demo data in files:

```
\startluacode
  io.savedata("foo.tmp","10 20 20 20 30 40 40 60
50 10")
  io.savedata("bar.tmp","10 10 20 30 30 50 40 50
50 20")
\stopluacode
```

We load the data in datasets:

```
\startMPcode
  lua.mp.datasets.load("foo","foo.tmp");
  lua.mp.datasets.load("bar","bar.tmp");
  fill area
  lua.mp.datasets.foo.Line()
  xysized (HSize/2-EmWidth,10ExHeight)
  withpen
    pencircle scaled .25ExHeight
  withcolor green/2 ;
  fill area
  lua.mp.datasets.bar.Line()
  xysized (HSize/2-EmWidth,10ExHeight)
  shifted (HSize/2+EmWidth,0)
  withpen
    pencircle scaled .25ExHeight
  withcolor red/2 ;
\stopMPcode
```

Because the datasets are stored by name, we can use them without worrying about them being forgotten:



If no tag is given, the filename (without suffix) is used as a tag, so the following is valid:

```
\startMPcode
  lua.mp.datasets.load("foo.tmp") ;
  lua.mp.datasets.load("bar.tmp") ;
\stopMPcode
```

The following methods are defined for a dataset:

method	usage
Size	the number of subsets in a dataset
Line	the joined pairs in a dataset making a non-closed path
Data	the table containing the data (in subsets, so there is always at least one subset)

Due to limitations in MetaPost suffix handling the methods start with an uppercase character.

Remark

Currently, the features described here are still experimental but the interface will not change. There might be a few more accessors and for sure more Lua helpers will be provided. As usual I need some time to play with it before I make up my mind. It is also possible to optimize the MetaPost–Lua script call a bit, but I might do that later.

When we played with this interface we ran into problems with loop variables and macro arguments. These are internally more or less anonymous. Take this:

```
for i=1 upto 100 : draw(i,i) endfor ;
```

The `i` is not really a variable with name `i` but becomes an object (capsule) when the condition is scanned, and a reference to that object when the body is scanned. The body of the for loop gets expanded for each step, but at that time there is no longer a variable `i`. The same is true for variables in:

```
def foo(expr x, y, delta) =
  draw (x+delta,y+delta)
enddef ;
```

We are still trying to get this right with the Lua interface. Interesting is that when we were exploring this, we ran into quite some cases where we could make MetaPost abort due some memory or stack overflow. Some are just bugs in the new code (due to the new number model) while others come with the design of the system: border cases that never seem to happen in interactive use while the library use assumes no interaction in case of errors.

In ConTEXt there are more features and helpers than shown here but these are discussed in the MetaFun manual.

Hans Hagen