MAPS

NUMMER 44 • VOORJAAR 2013

REDACTIE

Taco Hoekwater, hoofdredacteur
Wybo Dekker
Frans Goddijn

NTG NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

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

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

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

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

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

Productie. De Maps wordt gezet met behulp van een LATEX class file en een ConTEXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.40.11 en luatex 0.70.1 draaiend onder Linux 2.6. De gebruikte fonts zijn Linux Libertine, het niet-proportionele font Inconsolata, schreefloze fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

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

Inhoudsopgave

Redactioneel

Dit is het laatste redactioneel van mijn hand. Het werken aan de Maps is al een tijd niet meer te combineren met mijn andere bezigheden, met als gevolg dat de deadlines nu al een paar issues lang niet gehaald worden. Sommige artikelen in deze Maps lagen zelfs al meer dan twee jaar op de plank, waarvoor mijn nederige excuses.

Hoog tijd om de stok door te geven aan iemand anders. Met ingang van de volgende reguliere Maps is er een nieuwe hoofdredacteur, Michael Guravage. Hoewel ik wel ‚in de buurt' zal blijven voor de technische kanten van de productie, zal Michael alle redactionele activiteiten gaan coördineren. Ik heb er het volste vertrouwen in dat volgende Maps-en onder zijn leiding goed gevuld en vooral beter op tijd zullen verschijnen.

De Maps die nu voor u ligt is flink gevarieerd. Zoals de laatste tijd gebruikelijk is zijn er weer artikelen van Kees van der Laan over PostScript; zijn er artikelen over nieuwe modules voor ConTEXt van zowel Hans Hagen als Hans van der Meer; is er aandacht voor de ConTEXt wiki (door Sietse Brouwer); een boek review van Koen Wybo; C.M. Fortuin legt uit hoe een kegelsnede benaderd moet worden; en er is een kleine MetaPost bijdrage van mijzelf.

We beginnen met een bespiegelend stuk van Hans Hagen over de toekomst van TEX, en we eindigen met een verslag van de vijfde ConTEXt bijeenkomst door Michael Guravage. Dat betekent dat het laatste artikel van deze Maps geschreven is door degene die in de volgende Maps als eerste aan het woord is, en het eerste artikel in deze Maps is van de persoon die waarschijnlijk als allerlaatste nog aan het woord gaat zijn in de verre toekomst.

Wat mij betreft is dat een gepast einde van mijn carriere als hoofdredacteur. Rest mij nog om Michael veel succes toe te wensen bij het voorbereiden van de volgende Maps-en, en voor u als lezer:
veel leesplezier toegewenst!

Taco Hoekwater

Does TEX have a future

Introduction

Making the transition from ConTEXt MkII to MkIV took a lot of time. In the process all kinds of code were evaluated, improved and, occasionally, removed. To some extent, the frozen state of MkII reflects the requirements of automated typesetting of the past two decades. Today, LuaTEX is advancing automated typesetting beyond what was previously possible. But do we really need it? In this article I will describe several issues we faced while rewriting the code, the choices, and compromises, we made. I will not attempt to answer the question whether TEX has a future, but merely offer you my own observations and thoughts.[1]

Media

It is not hard to extrapolate the advance of e-books, and the demise, in some countries, of paper books. Less demand for printed books means less need for typesetting. Of course, real-time rendering is also typesetting. But since there is no one format compatible with all e-book readers, publishers are unlikely to produce multiple device-specific versions. To what extent is a shift in the way documents are encoded important for TEX development? And as publishers cut quality and costs in an attempt to stay alive, who will want high quality output? Personally, I think more and more authors will turn to self-publishing. In this respect we might see a revival of TEX and more complex typesetting. It all depends on how important a particular look and feel is, and what price you are willing to pay to achieve it. Nevertheless, we cannot deny the fact that times are changing, and that technological developments will influence how TEX-like systems evolve.

From the start ConTEXt could produce very complex interactive documents. But apart from its inclusion in several projects, this functionality has never been in any serious demand by the publishing world. One reason for this is that compared to the printed product, interactivity is seen as an additional 'free' feature. As we enter the age of electronic books, we see that the features commonly used are only a portion of those available. Nevertheless, all this accumulated functionality is available in MkIV. When a typesetter has an eye for quality, interesting typographic and navigational details will appear.

Application

It is quite usual to find ConTEXt hidden in a larger publication workflow. In such cases the input comes from a database or some online editing environment. The layout, and therefore the typesetting, are often relatively simple. A predefined style tells ConTEXt how to transform input to output. The input may be predictable, but the user still has significant influence on the workflow. In this situation, what sets MkIV apart is its ability to analyze and manipulate data sets. MkII can also deal with data, but with Lua on board, MkIV solutions seem more natural. MkII is sufficient for traditional typesetting situations, but MkII is a dead end compared to MkIV.

We often talk of TEX users and user groups, but the more abstract term *usage* might be a better indicator of how much *TEX* is used. The number of TEX users is not growing, but TEX usage might be on the rise. Perhaps counting the number of pages produced with TEX is a better indicator of how prevalent TEX is rather than counting the number of installed TEX systems.

Coding

Another observation is that ConTEXt users often produce more advanced and demanding documents than I do as part of my work. For me, the biggest advantage of MkIV is its support for OpenType. Fonts are easier to install, and all those encodings disappear. Another advantage is that MkIV has a flexible xml processor built in, which can save you time in solving problems. Of course we continue to use and improve basic rendering capabilities, but we often have to simplify solutions when designers fail to see the possibilities of automated typesetting. Stability is often cited as the reason to use older combinations of TEX engines and macro packages. Ease of use and improved maintenance might be sufficient reasons to move on.

1. This text was copy-edited for Maps by Michael Guravage, whom I gratefully thank for helping me express my thoughts.

In the early days of ConTEXt my colleagues and I were its main users. One of the nice things with TEX compared to a word processor —never in my life have I had to use one—is that you can automate things. Imagine that you attend a series of meetings where several hundred learning objectives are identified, described, ordered and grouped. If you are in charge of such a task, it really helps to have a system where numbering and breaking pages comes for free. We were often able to get the adapted documents in the post within a few hours of leaving the meeting. The authors were impressed when, in the next stage of the project, we presented them with multiple professional looking documents derived from the same source. No other application could easily handle 500 floating images on 300 pages without crashing. This was the time that using TEX paid off for us. That was more than 15 years ago.

Such a TEX-based workflow is a sequence of edit, run and preview cycles; steps recognizable to any old-time computer user. However, it is not something that newcomers, like our children, might deem usable. It's not 'what you see is what you get', but the more abstract process of 'what you key is what gets done'. Wrapping TEX with a simpler interface would only hide its power, flexibility and charm. Then, you might as well use a word processor. Let's face it, using TEX directly only pays off when the user can separate coding from rendering, wants to have full control and desires to be independent of hard coded solutions. Try explaining that to a twittering facebooking kid. Regardless of how we move from MkII to MkIV, the route from source to result remains the same, and so does the intended audience. Updating TEX engines and macro packages will not increase TEX usage.

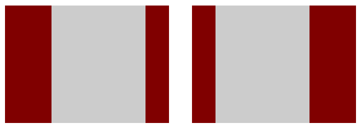
For some of our ConTEXt projects, traditional paper-based books are complemented by content intended for the web. Consequently, the document source is often xml. We could encode documents using a TEX-based coding, which, if they had the freedom to choose, would likely be more comfortable for authors to use. I wager that many authors who have used TEX directly still prefer it as an input language. Though xml is a widely accepted input and storage format, it is not ideal for typesetting. xml is geared toward publishing on the web and is not as expressive as TEX. However, reusing content is rare, so we needn't worry too much about encodings.

Coding in xml has some advantages for processing by TEX. There are no TEX commands for authors to misuse or redefine, and valid xml documents produce no errors. Another advantage is that styling and coding are completely separate. Of course, relieving the author of the responsibility of rendering complex documents can lead to sub-optimal output, unless the author is willing to adapt his content. The advent of xml has made people aware of the benefits of structure. ConTEXt tries to enforce structure, so TEX can fit nicely into modern publication workflows. However, for the quick and dirty one-time documents, the overhead of adding structure might not be worth the effort. So, even if in MkIV we promote using `\startchapter` over `\chapter` and `\startitem` over `\item`, we keep supporting the less demanding coding variants.

The styles I write today are a mixture of TEX, MetaPost and Lua. Solving the same problems with MkII, if possible, would require considerably more effort. Just as the faster Internet has become natural, so has the MkIV mix.

Double-sided

An electronic medium is single-sided. A book is always double-sided, and in the case of magazines and newspapers also multi-column. ConTEXt has quite some code to deal with double-sided layout. Sometimes TEX collects more content than can fit on one page. When this happens we have to keep track of where content should end up. For instance, dimensions and alignment conditions for margin notes must be swapped for odd and even pages.



In a single-sided universe, all the ConTEXt code that deals with `inner` and `outer` positioning and alignment could go away. Headers and footers could also be simplified. By removing the distinction between left and right pages, we could also drop some page synchronization code. Backgrounds wouldn't have to keep track of state either.

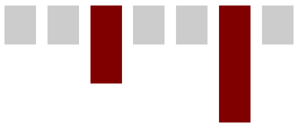
Related to this is page imposition. Page imposition is built into ConTEXt and is rather advanced. New imposition schemes occasionally appear through the effort of Willi Egger who not only typesets but also prints and binds books. The advent of a new folded paper gadget can be the impetus for adding yet another variable to control the position of pages.

Some of our projects require that we produce imposed products as part of an automated workflow. Cover pages, combined with back pages, are on the agenda for future integrated support. Since these features are applied to finalized pages implementing them is relatively easy, and they do not interfere much with existing code.

To separate content within electronic documents, we might end up with all sorts of cover-like pages. After all, additional e-pages are cheap, and color comes for free. This means that we might see more advanced page clustering and numbering schemes in ConTEXt MkIV. For instance, it might be nice if chapters had alternating or unique background colors. It would be even nicer if this property could be implemented without introducing new user commands in the source document.

Paper size

Paper books have standard page sizes; electronic books do not. Splitting tables with spans or large cells is somewhat painful. So why should we split a large table in an e-document when we could just as well scroll?

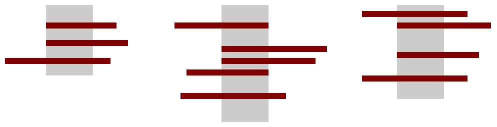


In some way we're going back in time. Long ago scrolls were used as a continuous medium. In that sense, scrolling on a display is not as new as it may look.

The concept of a page is derived from the medium — but what if we ignore this? For instance, if each chapter of a book were a separate entity, we could have one long page per chapter. This is problematic since TEX sets a limit on how high a page can be. But imagine that instead of thinking vertically we go horizontal. Headers and footers go away or get a new meaning, and the edges would give some indication of where we were. Perhaps we need a floating indicator; we've seen stranger things. Would this require a programmable viewer that we could control from our document, or could we anticipate standard features in viewers and viewing devices? Luckily for us we can adapt the TEX backend for either eventuality; at least we have done so for over three decades.

Floats

Floats are nice for paper. It is interesting to notice that in ConTEXt's early years floats were very prevalent in the documents we produced. In fact, they were a selling point. In educational documents especially, graphics need to appear near to where they are mentioned in the text. In a purely electronic document we don't need to struggle with fitting graphics on a page. Relaxing this requirement would simplify designs. Removing the corresponding ConTEXt code would definitely make the codebase leaner and meaner. But don't worry, we have no plans to delete anything.



What if we combine the previously mentioned vertical layout with horizontal extensions? Again with a finger we swipe our way down the page, where we run into an indicator denoting a larger image. Swiping our finger to the left displays the image; which might be accompanied by texts, images or animations. Another swipe and we're back in the main thread. It is amazing that we can do this with TEX. In fact we can proceed to multi-dimensional or even parallel documents. I remember turning the MetaFun manual into a QuickTime 360 movie. I must have a ConTEXt presentation style somewhere that implements this one page presentation where clicking on areas exposes different parts of the page. TEX is and will always be a fine playground for such concepts. MkIV with Lua and MetaPost makes it even finer.

Margins

The first step from a paper document to, for example, an e-book device is to get rid of margins. Due to technical limitations all devices shipped around 2012 have rather hard-coded physical margins. Perhaps one day we will have devices that have matte displays running from edge to edge. Imagine a device without buttons, logos or stickers proudly mentioning the internal chip sets or operating system.

The current tendency is to remove margins. In the near future we might see them coming back. Margins provide structure, and also room for various indicators and navigation aids. This is a good thing. Support for putting things in margins is quite important. In MkIV we already go further than in MkII and more will come.

Accessibility

A table of contents still makes sense in an electronic document, but what about an index? An index's usefulness is proportional to how carefully it was prepared. In many cases a search option works just as well. The concept of a table of contents can be expanded to include local tables and navigation aids that help the reader find what he wants. Similarly, we can collect information in multiple indexes. We added multiple interactive indexes to ConTEXt while involved in a project that produced quality assurance manuals. In another project we needed index entries arranged in a linked list, which is why this functionality exists in MkII. This cross-linked variant is not yet available in MkIV – simply because I don't know anybody who needs it. Interestingly, implementing it in MkIV is far easier than in MkII.

A great deal of functionality, some of it even documented, is there because we once needed it. Take, for instance, flow charts. We can make really big ones. Selected cells can become hyperlinks — allowing us to jump through the document. Again, this functionality was a side effect of making those interactive QA manuals.

Mechanisms like these have always been part of ConTEXt, even when they make no sense for paper documents. They are more coding issues than demanding typographical challenges. They do not interfere with other typographical components, so simplifying or removing this functionality has no benefits. We can do much more in MkIV, but sometimes I get the feeling that less is more.

A lot of code in ConTEXt deals with structure. It makes sense to think about ways to improve how we gain access to it: linked lists, pop ups, summaries, reading routes, etc. MkII has several mechanisms that make controlled reading possible, but they never took off. In MkIV most mechanisms that structure data also retain part of it for re-use. Because we store data for use in a second or subsequent typesetting pass, information can be used multiple times.

Some mechanisms also support user data. For instance, when starting a chapter, besides setting its title, you can also name a variable that stores the name of an image — a sort of visual title. As this name is carried around, the image can appear as an icon in the table of contents and on the first page of the chapter. We needed this a long time ago in MkII. This is one reason why in MkIV we can now set user variables in commands that start chapters and sections.

In one project we participate in, a free math method, the content is first published on the web. Given the nature of electronic documents, it went unnoticed that, when typeset for the printed page, the document was quite large. Selective use of content, multiple products, and efficient typesetting are solutions to this. The e-book version is not constrained by the number of pages. Information can be repeated when needed; complemented with the necessary navigational aids. I'm confident that ConTEXt can deal with both variants.

There has been a time, probably due to the fact that I gave presentations showing pdf on a projector, that ConTEXt was promoted as a system for creating electronic documents in pdf format. This is just one feature, but interaction has always been integrated in the core — never an add-on. However, there is a fundamental difference between interaction in MkII and MkIV. Using different techniques in MkIV, we no longer have interfering status nodes. This makes the whole mechanism more robust, although internally it has become pretty complex.

Columns

Columns make sense in broadsheet newspapers and journals where one wants to put as much as possible on a page. But I wonder if columns make sense in electronic documents. After all, electronic pages are cheap, and getting rid of multi-columns makes typesetting much easier. In TEX the mechanisms that deal with columns, e.g., page builder, floats and notes, are often complex. The code can be pretty messy. It would be nice to get rid of this legacy.

A good application of columns can be found in parallel bible translations. Not only must the text be synchronized in multiple columns, it also has to be broken across pages in a reasonable way. Footnotes are another complication.

Will such products be made in the future? The production of printed encyclopedias has already stopped, and concordances might soon follow. On the other hand, the fact that Thomas Schmitz typesets sophisticated documents for tablets, notebooks, projectors, and paper indicates that, for critical editions, the future is not yet determined. And I know several TEXies who typeset catalogs for conferences and festivals where a proper paper version is the only way to provide an effective overview. All these documents share a mixture of one column, multi-column and specially composed pages.

ConTEXt currently has two mechanisms that deal with columns. The first mixes well with single column mode, the second is more powerful and encapsulated. In MkIV the pluggability of the output routine has been improved; so if needed we can support yet unforeseen page building schemes. Parallel streams are first on the agenda.

Move on

If we consider only paper documents, do we anticipate needing more typesetting functionality than we already have? Does it make sense to develop macro packages any further? Of course, it is not difficult to make a wish list including more support for complex critical editions and parallel typesetting of translations. For those who use a simple input format such as Markdown, existing ConTEXt functionality is more than sufficient. In fact, as long as we can deal with the concepts found in html we're okay. Most of these documents consist only of running text, tables, images, a bit of sectioning, itemized lists and maybe descriptions.

TEX is over thirty years old. It is still maintained and kept up-to-date. It provides users with a lot of freedom. It has an active user community. It is often chosen for long term use. It is boringly stable. With these attributes, we can safely assume that TEX will be around for a while. The same is true for macro packages. They will stay and evolve. But how will TEX change along the inexorable path from paper to electronic media? Typesetting habits change slowly, so we still have some time to ponder these questions.

On the other hand, look at how quickly the web is evolving, and how quickly younger generations adapt to new electronic devices. When using TEX it is natural to think in book-related categories. But just as a computer desktop is not a real desktop, an e-book is not a real book. Real books have a physical presence; we can hold them in our hands and turn each page as we read. E-books try to mimic these physical characteristics with ridiculous results. For example, you can choose an e-book from an e-bookshelf, and turning pages is simulated by showing the binding and a moving cut edge. But will we want or need these visual clues in the future when we have instant access to everything from anywhere on any device we choose? Why carry around a book when we can have its contents projected on our retina, or hear it spoken in our ear? Regardless of how TEX and its attendant macro packages evolve, we'd best refrain from predicting the future, let alone promote TEX as the ultimate and last word on typography. We can only hope that future hardware and software will allow us to TEX like we allow printers to use printing presses.

Hans Hagen

CD and DVD labels

TIMTOWTDI

Abstract
How to make CD and DVD labels by PostScript, to be printed on prefab glued paper, assisted by Photoshop for the conversion of an illustration into EPSF, is explained.

Keywords
Adobe, afii, CD-ROM, ConTEXt, DVD, EPSF, extending encoding vector, label, lightscribe, minimal encapsulated PostScript, numero sign, OTF, Photoshop, plain TeX, PSlib, TEXworks

Introduction

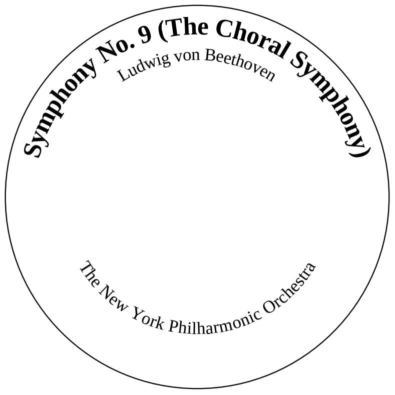
At the EuroTEXConTEXt 2009 meeting I attended the tutorial DECORATING CD-ROMs AND DVDs by Willi Egger. I was surprised that it was all about how to use ConTEXt for the purpose. Nothing wrong with that, but ... TIMTOWTDI to quote Larry Wall, the author of PERL. The first NTG secretary, Gerard van Nes, used to say that all tastes should be catered for, material for a broad audience should be communicated. Moreover, we should be aware of competing tools for the purpose.

I agreed with Willi to come up with an alternative approach in PostScript, well ... combined with Photoshop and for the printing the use of prefab glued paper.

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.

Within the TEX community the TEXCollection DVD and NTG's MAPS1-24 CDROM are well-known. On the other hand there is the example in Adobe's Blue Book of 1985 — Symphony No 9 — to illustrate the typesetting along circular arcs in PostScript.


Symphony No. 9 (The Choral Symphony)
Ludwig von Beethoven
The New York Philharmonic Orchestra

We see from the example labels that the tool must be able to handle

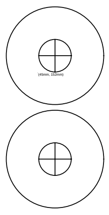
□ text along circular arcs, in general arbitrary placement of text
□ various fonts
□ background colour or illustration

and not to forget

□ to print the label on the right place on the prefab glued paper.

In this note I'll show how we can enrich the 1985 label of Adobe by an illustration. Nothing new, I just use what is already available. The example program can be adapted to your circumstances.

← Prefab glued paper



template for →
printing on glued paper

How?

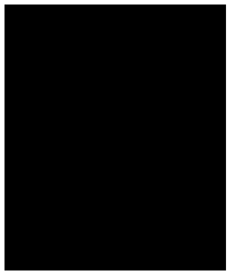
Photoshop 9 is used to scale and convert an illustration: scale to 12cm by 12cm say (one can enrich the illustration if one wishes or add text in a fancy font) and convert from `.jpg or ...` into `.eps`. The resulting EPSF, with extension `.eps`, is embedded by `run` into the PostScript label job. The total is converted into `.pdf`, as usual, and printed on the prefab glued paper, et voilà.

Conversion of '.jpg' into '.eps'

When I agreed with Willi I had no idea how to do the conversion.
It turned out to be very, very simple, because it has been done already by Adobe: open the picture in Photoshop, scale to the size of the disc or a little larger — 12cm by 12cm say — adjust for the required pixels, and save as (transparent, tho not relevant when the illustration is used as filling background) `.eps`, i.e. Encapsulated PostScript with the illustration cropped to the bounding box with the lower left corner at the origin i.e. left under on the paper. That is all. One only has to be aware of it.

Illustration

As background I selected part of the (original) score of the 9th Symphony — Ode an die Freude, also known as, The Choral Symphony (a picture of Beethoven is not suited because of the hole in the centre of the label.)



Courtesy score: `http://beethoven.staatsbibliothek-berlin.de/index.html`

Take note of the supplied BoundingBox upper values in the converted `.eps` file — `uux uuy` (in pnts) — which reflect the size of the picture.
The inclusion of the illustration — `partituur9e.eps` — is done by

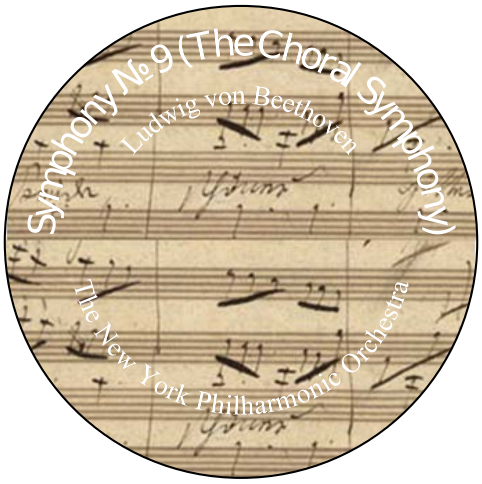
```
(C:\\CD-DVD-labels\\partituur9e.eps) run
```

but ... we have to position the illustration over the place of the disc on the template of the prefab paper before the above by the invoke $-.5uux$ $-.5uuy$ `translate`, i.e. transform the origin of PostScript's User Space, appropriately.

The result

In the script the order of invokes is important because of the opaque nature of the PostScript painting operators. In the case at hand the illustration is painted first on the clipped area, the upper disc on the template of the prefab paper.

At the end the texts along circular arcs are typeset by `outsidecircletext` respectively `insidecircletext` onto the disc and painted to the current page, as prompted by Program 10 of Adobe's Blue Book.


Symphony № 9 (The Choral Symphony)
Ludwig von Beethoven
The New York Philharmonic Orchestra

```
%!PS-Adobe-3.0
%%Title: Blue Book label with score illustration
%%Creator: Kees van der Laan, kisa1@xs4all.nl
%%CreationDate: March 2011
%%BeginProlog
(C:\\PSlib\\Bluebook.eps)
  run %inclusion of BlueBook library
%
/toplabel {
   /mm {72 25.4 div mul} def
   105 mm 212 mm translate  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
%
/Numerovec [8#43 /afii61352]
  def %numero sign mod in encoding vector
%%EndProlog
```

```
%
%---Program--- the script
%
/MinionPro-Regular
  /MPR Numerovec ReEncodeSmall %Courtesy Adobe BlueBook Program 18, p207
%
3 setlinewidth toplabel gsave stroke grestore eoclip
%
gsave
   -170 -170 translate %reposition origin
   (C:\\CD-DVD-labels\\partituur9e.eps) run %embed illustration
grestore
%
%annotations as prompted by Program 10 of Adobe's Blue Book
1 setgray     %but ... with annotations in white
/size 27 def  %and larger
/MPR size
  selectfont %LRM3 concatanation of findfont scalefont setfont
(Symphony No. 9 (The Choral Symphony)) size  90 135 outsidecircletext
/size 20 def
/Times-Roman size
  selectfont %LRM3 concatanation of findfont scalefont setfont
(Ludwig von Beethoven)                 size  90 118 outsidecircletext
(The New York Philharmonic Orchestra)  size 270 118 insidecircletext
showpage
%%EOF
```

There is nothing new in the design of the jewelcase inlay. The (vertical) text at the back can be obtained easily in PostScript after rotating User Space by 90°.

If outlines are wanted for the texts use oshow, outline show, from my PSlib library, or create an outline font variant as prompted by Program 16 of Adobe's Blue Book.

Related work

John Deubert's Acumen J. nov 2004 explains in detail how to include graphic files into (handwritten) PostScript. Useful. (His editing of EPSF is no longer necessary.)

A quote from John Deubert

> "If you do variable data or any other handwritten PostScript code, EPS is the greatest thing since coffee. It allows you to effortlessly incorporate into your PostScript output graphics and images from any professional software."

Don Lancaster in 1997 provided his own filter and hoped it will work generally. Well … when conversion is done by an Adobe tool it works.

Acknowledgements

Thank you Willi Egger for your inspiring tutorial which gave rise to this work. Thank you Adobe for your maintained, adapted to version 3 since 1997, good old, industrial standard PostScript and `Acrobat` to view it, Don Knuth for your stable plain TEX, Jonathan Kew for the TEXworks IDE, Hàn Thế Thành for your pdfTEX&Co.

Karel Horák drew my attention to `No`, the numero symbol, which is typografically not correct typeset in the Adobe example, as admitted by Adobe, says Karel, although Wikipedia mentions variations in use. As bonus I have adjusted the flaw.[1]

Thank you MAPS editors for improving my use of English, Jos Winnink for your opinion and suggestions, and Taco Hoekwater for procrusting my plain TEX note into MAPS format.

Notes

1. The TEXnique is to associate the PostScript `8#43` code of the `#`-key with the (Adobe) name `afii61352` for the numero character which on its turn is associated with the numero glyph in the `MinionPro-Regular` font. A nice application of the `ReencodeSmall` procedure as given in Program 18 p.207 of the Blue Book.

My case rests, have fun and all the best.

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

Review of *LaTEX and Friends* by Marc van Dongen

Abstract
Zeggen dat LaTEX niet gemakkelijk is om aan te leren, is een open deur intrappen. Met een degelijk boek als LaTEX and Friends word je meer dan behoorlijk op weg gezet.

Auteur Marc van Dongen doceert aan de faculteit Computerwetenschappen aan de universiteit van Cork. Dit is niet onbelangrijk omdat hij deze doelgroep van informaticastudenten voor ogen houdt bij het opmaken van dit boek. Er wordt dan ook aandacht gespendeerd aan wat deze studenten nodig hebben om een mooie thesis met beelden, diagrammen, mathematische formules, dataplots, enz. te kunnen afleveren.
De auteur start vanaf nul maar houdt er de pas goed in. Informatie wordt meestal maar eenmaal aangereikt, dus opletten en goed lezen is de boodschap. Op het eind van het boek kan je met behulp van LaTEX alle mogelijke obstakels bij het maken van een complex document als een thesis overwinnen. Alles wat je hierbij aan inzicht, pakketten, commando's, tools, enz. nodig kan hebben, is in het boek aan bod gekomen. Hierbij heeft hij – in tegenstelling tot sommige andere auteurs – niet zomaar alles op een hoopje gegooid waardoor je bij hem niet je eigen weg nog moet zoeken. Marc van Dongen is een docent 'pur sang' en weet heel gestructureerd, overzichtelijk, vlot leesbaar en doorspekt met voorbeelden het gebruik van LaTEX uit te leggen.
Deze invalshoek van het boek maakt het uniek: je hebt een specifieke doelgroep en een specifieke doelstelling. Dit wil niet zeggen dat anderen er geen gebruik van kunnen maken. We verduidelijken even.

De structuur wijkt af van andere boeken en dat is duidelijk een voordeel.
Deel 1 spendeert de auteur aan de diverse componenten waaruit een complex document is opgebouwd. Front, main en back matter, labels, document management, bibliografie, inhoudsopgave, classfiles, enz. vormen samen een mooi geheel en deze kun je met gemak met LaTEX realiseren. Hij last af en toe een 'intermezzo' in om naast de technische kant ook stil te staan bij esthetische aspecten. Zo ontstaat er een symbiose tussen techniek en schoonheid in een document.
De verdere delen zijn degelijke en goed gestructureerde inleidingen op 5 topics:

- ☐ tekst en lijsten (pag. 41–72)
- ☐ visualisatie van data in tabellen, diagrammen en dataplots (pag. 75–141)
- ☐ wiskunde en algoritmes (pag. 143–192)
- ☐ automatisatie: door de gebruiker gedefinieerde commando's en omgevingen, branching (pag. 195–222)

- diverse: presentaties met beamer, aan de slag met een eigen class, gebruik van opentype fonts. (pag. 225–264)

Het boek brengt meestal de bekende ingrediënten om aan de slag te gaan. Marc van Dongen verliest zich niet in het voorstellen van diverse pakketten die elk hun voor- en nadelen hebben. Integendeel, hij maakt een bewuste selectie van de voor hem juiste tools. Zo is bij het hoofdstuk over diagrammen niets te vinden over postscript of metapost maar wordt resoluut gekozen voor Tikz. Niet dat de auteur niet op de hoogte is van deze, integendeel[1]. Hij selecteert juist dit pakket omwille van controle, stijlconsistentie en zijn vlot omgaan met tekst en wiskundige symbolen als ook vlotte integratie met de beamerclass. En dit typeert weer een ander facet van dit boek: rechttoe rechtaan kennis, de juiste pakketten die de job snel en degelijk kunnen klaren. Dit alles met oog voor stijl en goed design.

Marc van Dongen stelt zijn boek voor op zijn website: `http://csweb.ucc.ie/~dongen/LAF/LAF.html`. Deze moet je gewoon bezoeken. Niet enkel omdat er een mooi video-interview met de auteur terug te vinden is maar ook voor enkele interessante extra's.

- Een **aanvullend hoofdstuk** over de installatie van LaTEX met behulp van TeXlive op zowel Microsoft Windows en linux-unix. In dit hoofdstuk stelt hij ook twee programma's voor: jabref als bibliografie-editor en texworks als LaTEX-editor. Directe link: `http://csweb.ucc.ie/\~dongen/LAF/LAF.pdf`
- Ook zijn er interessante **presentaties** te vinden die Marc van Dongen gegeven heeft aan de universiteit. Hierin legt hij het gebruik van LaTEX uit. Deze presentaties volgen de indeling van het boek en zijn bovendien overladen met voorbeelden. Ook zeker de moeite om door te nemen.

Voor de beginner is LaTEX and friends een absolute aanrader: het zet je snel op weg maar je moet wel aandachtig lezen en bij de les blijven.
Als je reeds vertrouwd bent met LaTEX , zijn er geen grote ontdekkingen te vinden in dit boek. Het weet wel te verrassen met details die er echt toe doen. Door de structuur en de nauwgezette info die in dit boek aanwezig zijn, is het ook een mooi naslagwerk om het geheugen op te frissen en zelfs om je eigen kennis compleet te maken.

LaTEX and Friends, Marc van Dongen, Springer, 2012, 299 p. 162 illus, ISBN 978-3-642-23815-4 (hardcover) en ISBN 978-3-642-23816-1 (eBook)

Noten

1. Zie zijn uitgebreide website over metapost op `http://csweb.ucc.ie/\~dongen/mpost/mpost.html`

Koen Wybo

Kegelsneden benaderen

Abstract
Kegelsneden kunnen systematisch worden benaderd door hoekpunten van een (deel van een) omgeschreven veelhoek. Een algoritme wordt ontwikkeld voor het bepalen van de steunpunten van een iteratieve derdegraads 'Bézier' benadering van kegelsneden. Voor de start zijn drie punten nodig. Het algoritme is onafhankelijk van de stand van de kegelsnede.

Inleiding

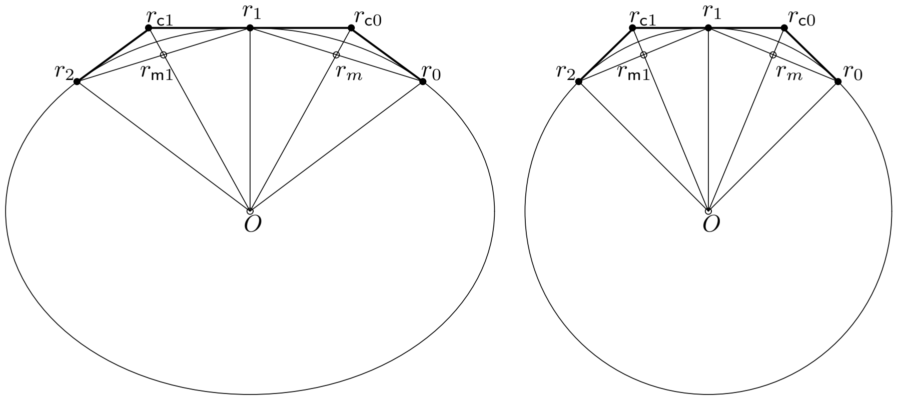
We stellen ons de vraag hoe een cirkel, meer algemeen een ellips, en nog algemener een kegelsnede, op het scherm, of op een printer, te tekenen. Daarvoor moeten vele punten worden berekend. Als steunpunten voor de berekening gebruiken we hoekpunten van een ingeschreven veelhoek en hoekpunten van een omgeschreven veelhoek. Het idee om cirkels te benaderen met veelhoeken komt oorspronkelijk van de oude Grieken. Archimedes heeft rond het jaar 240 voor Christus de waarde van het getal π benaderd met behulp van in- en omgeschreven veelhoeken van de cirkel, en bewees dat $3{10/71} < \pi < 3{1/7}$ $(3,1408 < \pi \approx 3,1416 < 3,1429)$. Hij gebruikte de twaalfhoek, en via verdubbeling achtereenvolgens de veelhoek met vierentwintig, achtenveertig en zesennegentig hoek(punt)en. (Archimedes, Cirkelmeting, -240)?)

De lengte van de zijde van de veelhoek was voor hem het middel om de omtrek, en daarmee π, te bepalen. Voor ons zijn de hoekpunten zelf een benadering van de kromme van de cirkel. We zullen laten zien hoe een goede benadering van de cirkel, of algemener de ellips, kan worden verkregen met behulp een algoritme gebaseerd op de hoekpunten van veelhoeken. De hoekpunten worden daarbij opgevat als vectoren in een nog ongedefiniëerd assenstelsel voor het platte vlak. De gedachte om deze punten te gebruiken bij het tekenen van krommen zien we reeds bij de Fransman De Casteljau (Courbes à Pôles,1959)?), praktisch leidend tot kwadratische of kubische interpolatie.

Een kanttekening. De ellips is niets anders dan een cirkel die eenzijdig is uitgerekt. Onder die uitrekking blijven rechten rechten, blijft evenwijdig evenwijdig, blijft raken raken. Kortom, zolang we naar de geometrie kijken kunnen we ons beperken tot de cirkel, de ellips zonder voorkeursassen, door de ellips langs de kortste as uit te rekken tot die evenlang is als de langste as. Meer in het algemeen kunnen we ons beperken tot de eenvoudigste vorm van kegelsneden met eenheidsassen (ellips wordt eenheidscirkel, hyperbool wordt eenheidshyperbool, parabool wordt eenheidsparabool)

Verband tussen in- en omgeschreven veelhoek bij ellips

Figuur 1 illustreert duidelijk het verband tussen de ellips, met de langste as horizontaal, en de cirkel. In de tekening is de ellips de horizontaal uitgerekte cirkel. Van de ingeschreven veelhoek liggen de hoekpunten op de ellips. Ze zijn opvolgend genummerd r_0, r_1, r_2, ... De hoekpunten van de omgeschreven veelhoek liggen op een parallele ellips, die we aanduiden met de letter 'c' ('circumscribed'), en zijn bijbehorend genummerd r_{c0}, r_{c1}, r_{c2}, ... Er is een duidelijk verband tussen opeenvolgende



Figuur 1.

hoekpunten van de omgeschreven veelhoek, zeg r_{c0}, r_{c1}, en het bijbehorende hoekpunt van de ingeschreven veelhoek, r_0. Het hoekpunt van de ingeschreven veelhoek (punt van de ellips) ligt precies halverwege de twee hoekpunten, het is het gemiddelde van de bijbehorende hoekpunten van de omgeschreven veelhoek:

$$r_1 = (r_{c0} + r_{c1})/2 \Leftrightarrow r_{c1} - r_1 = (r_1 - r_0) - (r_{c0} - r_1) \tag{1}$$

Omgekeerd is er een verband tussen het hoekpunt r_{c0} van de omgeschreven veelhoek met de bijbehorende hoekpunten r_0 en r_1 van de ingeschreven veelhoek. Nu is het hoekpunt van de omgeschreven veelhoek, ***vanuit het middelpunt*** O van de ellips gezien, een veelvoud $1/\lambda$ van het gemiddelde r_m van de bijbehorende hoekpunten van de ingeschreven veelhoek:

$$r_{c0} = r_m/\lambda \quad \text{vanuit } O \tag{2}$$
$$r_m = (r_0 + r_1)/2 \tag{3}$$

De grootte van de factor volgt door te kijken naar de rechthoekige driehoeken Or_1r_{c1} en Or_mr_1. De hoek bij het O is de helft van de veelhoek-hoek ϕ. Gemeten vanuit O is $|r_m| = R\cos(\phi/2)$, de straal van de ingeschreven cirkel van de ingeschreven veelhoek. Anderzijds is $|r_{c0}| = R/\cos(\phi/2)$, de straal van de omgeschreven cirkel van de omgeschreven veelhoek. De punten r_m en r_{c0} zijn vanuit O gezien een onderschatting respectievelijk een overschatting van het punt $r_{1/2}$ van de ellips in dezelfde richting. Merk op dat, vanuit O gemeten:

$$r_{c0} = r_{1/2}/\mu \quad \mu = \cos(\phi/2) \tag{4}$$
$$r_m = \mu r_{1/2} = \lambda r_{c0} \quad \lambda = \mu^2 \tag{5}$$

Algoritme met tweedegraads Bézier benadering ellips

We gaan uit van het eerste interval met de punten r_0, r_{c0} en r_1, die samen een driehoek vormen. De punten van het eerste interval bepalen de punten van het volgende interval r_1, r_{c1} en r_2. Voor het gemak van het scalaire vermenigvuldigen kiezen we O als oorsprong. We maken voor de tussenberekeningen ook gebruik van het middenpunt r_m en van de drie relatieve waarden (de reden hiervoor wordt duidelijk uit de vorm van de Bézier benadering):

(6)
$$\begin{aligned}(r_{c1}-r_1) &= (r_1-r_0)-(r_{c0}-r_0) \quad \text{volgens (1)}\\ r_1 &= r_0+(r_1-r_0)\\ r_{c1} &= (r_{c1}-r_1)+r_1\\ r_m &= \lambda r_{c1} \quad \text{volgens (2)}\\ r_2-r_1 &= 2(r_m-r_1) \quad \text{volgens (3)}\end{aligned}$$

Dit algoritme kan worden voortgezet in de volgende hoek van de veelhoek om zo steeds een punt van de ingeschreven veelhoek en bijbehorend punt van de omgeschreven veelhoek te bepalen. Merk op, dat op deze manier de sinus en cosinus functie alleen nodig zijn bij het bepalen van de beginpunten. Het algoritme bevat slechts scalaire vermenigvuldiging (waarin de halve hoek zit) en vectorverschillen.

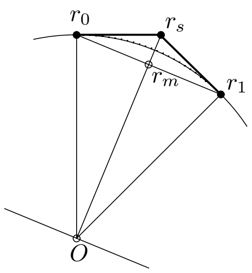
Met behulp van de twee hoekpunten van ingeschreven cirkel r_0 en r_1 en het bijbehorende hoekpunt $r_{c0} = r_s$ van de omgeschreven veelhoek kunnen we de ellips benaderen met een tweedegraads Bézier. De benaderings parameter t voor de Lagrange vorm is in dit geval de relatieve doorlopen hoek $t \approx (\varphi - \varphi_0)/\phi$. Vereenvoudiging geeft het gebruik van relatieve punten, ten op zichte van r_m, en daarmee van de relatieve parameter ten opzichte van φ_m, herschaald naar de helft van de doorlopen hoek ϕ, zodat $u = 2(t - 1/2)$.

De tweedegraads Bézier benadering is:

(kwadratisch Bézier) $r_{B2} = (1-t)^2 r_0 + 2t(1-t)r_s + t^2 r_1$

(veelterm t) $= r_0 + t(2(r_s - r_0) + t((r_1 - r_0) - 2(r_s - r_0)))$

(veelterm u) $= r_m + u(r_1 - r_0)/2 + u^2(r_m - r_s)/2$



De coëficiënten van de veelterm in u zijn symmetrisch rond het midden:

1. $a_0 = (r_m + r_s)/2$ het punt op halve hoogte van de driehoek $r_0 r_1 r_s$;
2. $a_1 = (r_1 - r_0)/2$, de numerieke eerste afgeleide in koorderichting;
3. $a_2 = -(r_s - r_m)/2$, de halve numeriek tweede afgeleide in radiële richting.

In de koorderichting is de benadering slechts lineair, in de radiële richting kwadratisch. De tweedegraads benadering is daarom een parabool, die precies raakt in de punten r_0 en r_1 en precies gaat door het punt $(r_m + r_s)/2$, halverwege de hoogte van de intervaldriehoek $r_0 r_1 r_s$.
Voor een preciezere beschrijving kiezen we een geschikt assenstelsel dat bestaat uit de lijn door O evenwijdig aan de basiskoorde $r_1 - r_0$, voor de x-as, en de symmetrielijn van het interval door O, r_m en r_s, voor de y-as. De assen staan, bij de cirkel, loodrecht op elkaar. De punten hebben coördinaten $r_{B2} = (x_{B2}, y_{B2})$: $r_1 = (\sin(\phi/2), \cos(\phi/2), r_s = (0, 1/\cos(\phi/2))$. Merk op, dat $y_m = y_1 = \cos(\phi/2)$ en $y_s = 1/y_1$. De tweedegraads Bézierbenadering componentsgewijs is:

(7)
$$\begin{aligned}x_{B2} &= ux_1\\ y_{B2} &= (y_1 + y_s)/2 - u^2(y_s - y_1)/2 = y_1 + (1-u^2)(1/y_1 - y_1)/2\end{aligned}$$

Eenvoudig is te zien dat de parabool buiten de (eenheids)cirkel (ellips) valt omdat de afstand tot de oorsprong groter dan 1 is. Radiëel is:

$$\begin{aligned} r_{B2}^2 &= u^2x_1^2 + y_1^2 + (1-u^2)(1-y_1^2) + ((1-u^2)(1/y_1 - y_1)/2)^2 \\ &= 1 + u^2(x_1^2 + y_1^2 - 1) + ((1-u^2)(1/y_1 - y_1)/2)^2 \\ &= 1 + ((1-u^2)(y_s - y_m)/2)^2 = 1 + (y_{B2} - y_m)^2 \end{aligned}$$

De grootste afwijking $\max(e_{B2})$ tussen ellips en parabool is in het midden, waar $u = 0$ en $r_{B2} = (0, y_{B2})$. Voor de ellips geldt dan:

(8)
$$\begin{aligned} \max(e_{B2}) &= (r_c + r_m)/2 - r_{1/2} = (1/2(1+\mu^2) - \mu)r_c \\ &= 1/2(1-\mu)^2 r_c = 1/2(1-\mu)^2 r_{1/2}/\mu \\ &= 1/2(1-\cos(\phi/2))^2 r_{1/2}/\cos(\phi/2) \\ &= 2\sin^4(\phi/4)/\cos(\phi/2) r_{1/2} \approx 1/128\phi^4 r_{1/2} \end{aligned}$$

Voor kleine hoeken ϕ, en die gebruiken we, is de sinus praktisch gelijk aan de hoek. De fout is verrassend van de vierde orde in de hoek, waar derde orde, vanwege het aantal punten, normaal zou zijn: dit is het gevolg van de symmetrie.

Iteratie van de 2degraads Bézier ellips

Wat gebeurt er met de fout als we de veelhoek verdubbelen? Als het aantal hoeken verdubbelt wordt de hoek ϕ gehalveerd. De fout wordt, vanwege de vierde macht, een factor $2^4 = 16$ kleiner. Bijvoorbeeld, uitgaande van een 12 hoek, met $\phi = 2\pi/12 = \pi/6 = 30°$, wordt $|e_2|/|r| \approx 0,0006$ dus na de verdubbeling naar een 24 hoek wordt $|e_2|/|r| \approx 0,00004$. Bij een straal van 5cm betekent dat een afwijking van ten hoogste 4.10^{-4}cm of 0,07pt.

Het is duidelijk dat bij zoveel hoekpunten de nauwkeurigheid van het herhaalde berekenen een grote rol gaat spelen. Iedere berekening geeft een fout, zodat de iteratie een cumulatie van fouten wordt. Het algoritme bevat 1 scalaire vermenigvuldiging en 5 vectoriële sommen. Zeg dat dat 10 standaardfouten geeft. Bij een 24 hoek zijn voor een rondgang totaal 240 standaardfouten mogelijk. We verwachtten dat dat maximaal 0,07pt is. Een standaardfout mag dan niet meer zijn dan 0,003pt zijn. Omgekeerd bepaalt de maximale fout de hoek:

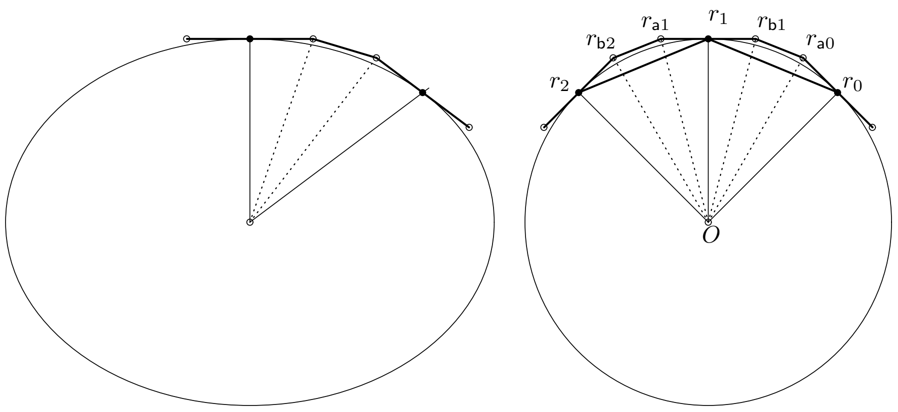
(9)
$$\phi \approx (128|e_2|/|r_{a0}|)^{-1/4}$$

Een 0,1mm [0,01mm] fout op een straal van 1cm [5cm] geeft een hoek van $1,06 = 60°$ $[0,161 = 9°]$.

Algoritme met derdegraads Bézier ellips

Om tot een derdegraads Bézier benadering te komen verdelen we de hoek in drieën, en laten de raaklijnen in r_0, r_1 enzovoort verder lopen tot $\phi/3$, een derde van de totale hoek (figuur 2). Op de omgeschreven veelhoek krijgen we op het interal van r_0 tot r_1 achtereenvolgens de punten r_0, r_{a0}, r_{b1} en r_1. Ze gaan tegen de klok in, zoals gebruikelijk in de wiskunde, en vormen een trapezium met de ellipspunten r_0 en r_1 als basis en de raaklijnpunten r_{a0} en r_{b1} als top. De toplijn $r_{a0} - r_{b1}$ raakt niet aan de cirkel, zodat de omgeschreven veelhoek nu slechts om en om raakt aan de cirkel. Dit is een gevolg van driedeling van de hoek, wat derdegraads Bézier suggereert. De afstand tot het midden van de toplijn is:

$$r_{a0} + r_{b1} = 2\alpha r_{1/2} \quad \alpha = \cos(\phi/6)/\cos(\phi/3)$$



Figuur 2.

Uit de posities van de punten van het interval, r_0, $r_{\text{a}0}$, $r_{\text{b}1}$ en r_1, bepalen we de posities van de punten in het volgende interval: r_1, $r_{\text{a}1}$, $r_{\text{b}2}$ en r_2. We gebruiken daarbij opnieuw de symmetrie ten opzichte van de straal naar het gemeenschappelijke punt r_1, waardoor het gemiddelde van twee gespiegelde punten op die straal ligt op een afstand $\cos(\alpha)$, waarbij α de projectiehoek is in O. We krijgen achtereenvolgens de symmetrie eigenschappen:

$$\begin{aligned} r_0 + r_2 &= 2\gamma r_1 \quad \gamma = \cos(\phi) \\ r_{\text{b}1} + r_{\text{a}1} &= 2r_1 \\ r_{\text{a}0} + r_{\text{b}2} &= 2\beta r_1 \quad \beta = \cos(2\phi/3)/\cos(\phi/3) \end{aligned}$$

Door combineren volgt daaruit het algoritme voor de iteratie. We zullen echter eerst ingaan op de benadering om te zien welke nodig zijn. Met behulp van de twee hoekpunten r_0 en r_1 op de ellips en de twee hoekpunten $r_{\text{a}0}$ en $r_{\text{b}1}$ van de omgeschreven veelhoek kunnen we de ellips benaderen met een derdegraads Bézier. Die benadert de ellips met een kromme die bepaalt wordt door een derdegraadsveelterm. De kromme raakt de ellips in de punten r_0 en r_1. Voor de Lagrange benadering is de parameter t gelijk aan de relatieve doorlopen hoek $t = (\varphi - \varphi_0)/(\phi)$. Voor de benadering relatief het midden, met de parameter $u = (\varphi - \varphi_m)/(\phi/2) = 2(t - 1/2)$, is de derdegraads Bézier benadering:

(Lagrange)

$$r_{b3} = (1-t)^3 r_0 + 3t(1-t)^2 r_{\text{a}0} + 3t^2(1-t) r_{\text{b}1} + t^3 r_1$$

(veelterm) $$= r_0 + t(3(r_{\text{a}0} - r_0) + t(3((r_{\text{b}1} - r_0) - 2(r_{\text{a}0} - r_0)) + {} + t(((r_1 - r_0) - 3(r_{\text{a}0} - r_0)) - 3((r_{\text{b}1} - r_0) - 2(r_{\text{a}0} - r_0))))$$

(tov midden)

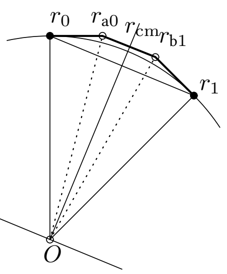
$$\begin{aligned} &= 1/8(r_0 + r_1 + 3(r_{\text{a}0} + r_{\text{b}1})) + 3/8u(r_1 - r_0 + r_{\text{b}1} - r_{\text{a}0}) + \\ &\quad + 3/8u^2(r_0 + r_1 - (r_{\text{b}1} + r_{\text{a}0})) + 1/8u^3(r_1 - r_0 - 3(r_{\text{b}1} - r_{\text{a}0})) \end{aligned}$$

De coëficiënten van de veelterm in u, symmetrisch rond het midden, zijn:

1. $a_0 = 1/8((r_1 + r_0) + 3(r_{\text{b}1} + r_{\text{a}0})) = 1/4(r_m + 3r_{am})$, het punt ven de benadering op de symmetrieas;

2. $a_1 = 3/8((r_1 - r_0) + (r_{\mathrm{b}1} - r_{\mathrm{a}0}))$, numeriek eerste afgeleide tangentiëel evenwijdig aan koorde;
3. $a_2 = 3/8((r_1 + r_0) - (r_{\mathrm{b}1} + r_{\mathrm{a}0})) = 3/4(r_m - r_{\mathrm{a}m})$, 1/2 numeriek tweede afgeleide radiëel;
4. $a_3 = 1/8((r_1 - r_0) - 3(r_{\mathrm{b}1} - r_{\mathrm{a}0}))$, 1/6 numeriek derde afgeleide tangentiëel.

In de radiële richting is de benadering van de tweede graad, in de tangentiële koorderichting van de derde graad. Het snijpunt van de diagonalen van het trapezium $r_1, r_0, r_{\mathrm{a}0}, r_{\mathrm{b}1}$, op 3/4 hoogte, is praktisch het punt $(r_m + 3r_{\mathrm{a}m})/4$, nog net buiten de ellips.
We werken de cirkel hier verder uit. De punten hebben daarin coördinaten $r_{\mathrm{B}3} = (x_{\mathrm{B}3}, y_{\mathrm{B}3})$: $r_1{=}(\sin(\phi_1), \cos(\phi_1))$, $r_{\mathrm{b}1}{=}(\sin(\phi_1{-}\phi_B), \cos(\phi_1{-}\phi_B))/\cos(\phi_B)$ met $\phi_1{=}\phi/2$ en $\phi_B{=}\phi/3$. Merk op, dat $|r_{\mathrm{b}1} - r_1| = \tan(\phi_B)$.
De derdegraads Bézierbenadering componentsgewijs is:



$$\begin{aligned} x_{\mathrm{B}3} &= 3/4u(x_1 + x_{b1}) + 1/4u^3(x_1 - 3x_{b1}) \\ &= ux_1 - 1/4u(1-u^2)(x_1 - 3x_{b1})) \\ y_{\mathrm{B}3} &= 1/4(y_1 + 3y_{b1}) + 3/4u^2(y_1 - y_{b1}) \\ &= y_1 - 3/4(1-u^2)(y_1 - y_{b1}) \end{aligned} \tag{10}$$

Voor kleine intervallen is de coëfficiënt van de derdegraadsterm in u zeer klein. Immers, bij verwaarlozing van de kromming is $3x_{b1} = x_1$. Daardoor is de tangentiële x-component in tweede orde lineair en de radiële y-component kwadratisch: de derdegraads benadering is in tweede orde een parabool, die in derde orde iets wordt uitgebold.
Preciezer uitgewerkt:

$$\begin{aligned} 1/4(x_1{-}3x_{b1}) &= 1/4\sin(\phi_1) - 3/4\sin(\phi_1{-}\phi_B)/\cos(\phi_B) \\ &= 1/4\sin(\phi_1) - 3/4\sin(\phi_1) + 3/4\cos(\phi_1)\tan(\phi_B) \\ &= 3/4\cos(\phi_1)\tan(\phi_B) - 1/2\sin(\phi_1) \\ 1/4(y_1 + 3y_{b1}) &= 1/4\cos(\phi_1) + 3/4\cos(\phi_1{-}\phi_B)/\cos(\phi_B) \\ &= 1/4\cos(\phi_1) + 3/4\cos(\phi_1) + 3/4\sin(\phi_1)\tan(\phi_B) \\ &= 3/4\sin(\phi_1)\tan(\phi_B) + \cos(\phi_1) \\ 1{\mp}1/4(y_1 + 3y_{b1}) &= 1 \mp \cos(\phi_1) \mp 3/4\sin(\phi_1)\tan(\phi_B) \\ &= 2(\sin^2[\cos^2](\phi_1/2) \mp 3/4\sin(\phi_1)\tan(\phi_B) \\ &= 3/4\sin(\phi_1)\left(4/3\sin^2[\cos^2](\phi_1/\sin(\phi/2)\cos(\phi/2) \mp \tan(\phi_B)\right) \\ &= 3/4\sin(\phi_1)\,(4/3\tan[\cot](\phi_1/2) \mp \tan(\phi_B)) \end{aligned}$$

Als $\phi_1{=}\phi/2$ en $\phi_B{=}\phi/3$ dan zijn eerste en vierde linkerlid nul in eerste orde, en wel van de orde ϕ^3 respektievelijk ϕ^4. Preciezer gezegd $1/4(x_1 - 3x_{b1}) \approx 3/4((1/3)^3 - 2/3(1/2)^3)\phi^3/3 = -5/432\phi^3$ en $\max(e_{\mathrm{B}3}) = 1/4(y_1 + 3y_{b1}){-}1 \approx 3/4(\phi/2)(4/3(1/4)^3{-}(1/3)^3)\phi^3/3 = 7/3456\phi^4$. Deze derdegraads Bézier fout is van dezelfde orde als de tweedegraads Bézier fout—slechts 4 keer kleiner.

We zullen aan de hand van de cirkel laten zien dat de gehele benadering buiten de cirkel, dus ook buiten de ellips, ligt. P_1 ligt op de cirkel, $y_1^2 = 1{-}x_1^2$, en P_{r1} op de raaklijn loodrecht op de straal: $y_1(y_1{-}y_{b1}) + x_1(x_1{-}x_{b1}) = 0$, zodat tezamen $y_1 y_{b1}{+}x_1 x_{b1} = 1$. Voor de radiële r^2fout geldt enerzijds en anderzijds:

$$
\begin{aligned}
r_{\mathrm{B3}}^2 - 1 &= (1+e_{\mathrm{B3}})^2 - 1 = e_{\mathrm{B3}}(2+e_{\mathrm{B3}}) = (x_{\mathrm{B3}}^2 + y_{\mathrm{B3}}^2) - 1 \\
&= \bigl(ux_1 - 1/4u(1-u^2)(x_1 - 3x_{b1})\bigr)^2 + \bigl(y_1 - 3/4(1-u^2)(y_1 - y_{b1})\bigr)^2 - 1 \\
&= u^2x_1^2 + y_1^2 - 1/2u^2(1-u^2)x_1(x_1-3x_{b1}) - 3/2(1-u^2)y_1(y_1-y_{b1}) + \\
&\quad + \bigl(1/4u(1-u^2)(x_1-3x_{b1})\bigr)^2 + \bigl(3/4(1-u^2)(y_1-y_{b1})\bigr)^2 - 1 \\
&= (u^2-1)\left(x_1^2 + 1/2u^2x_1(x_1-3x_{b1}) - 3/2x_1(x_1-x_{b1})\right) + \\
&\quad + 1/16(1-u^2)^2\left(u^2(x_1-3x_{b1})^2 + 9(y_1-y_{b1})^2\right) \\
&= 1/16(1-u^2)^2\left(8x_1(x_1-3x_{b1}) + u^2(x_1-3x_{b1})^2 + 9(y_1-y_{b1})^2\right) \\
&= 1/16(1-u^2)^2\left(-16 + (y_1+3y_{b1})^2 + u^2(x_1-3x_{b1})^2\right) \\
&= (1-u^2)^2\left(((1/4(y_1+3y_{b1}))^2 - 1) + u^2(1/4(x_1-3x_{b1}))^2\right)
\end{aligned}
$$

De factor $(1-u^2)^2$ zorgt voor de perfecte aansluiting op de cirkel bij de hoekpunten. De term $(1/4(y_1+3y_{b1}))^2 - 1$ is precies de fout van r^2 op de symmetrieas. Zolang het punt op de symmetrieas buiten de cirkel is, is het rechterlid positief, dus $e_{\mathrm{B}} \geq 0$: dan is ook de benadering buiten de cirkel.
Merk op, dat de exacte positie van de omgeschreven hoekpunten niet is gebruikt in de afleiding van de radiële fout. De hoekpunten kunnen nog aangepast door niet exact 1/3 van het interval te nemen! De fout wordt kleiner als we $1/4(y_1{+}3y_{b1}) = 1$ maken. Kies daartoe r_{b1} zo, dat $1/4(r_m{+}3r_{bm})$ exact op de cirkel (ellips) ligt. De benadering ligt dan nog steeds geheel buiten de cirkel maar heeft nu ook perfecte aansluiting op de cirkel in het midden. Daarvoor is nodig dat $\tan(\phi'_B){=}4/3\tan(\phi/4)$. Voor kleine hoeken is $\phi_{B'} \approx 1/3\phi$, maar voor grotere hoeken is $\phi_{B'} \approx 1/3\phi(1{-}7/432\phi^2)$. Voor 90 graden is $\phi_{B'} = 0,3212\phi$, dat is minder dan 4 procent onder 1/3. De fout is maximaal voor die u^2-waarde waarbij $|e|$ maximaal is. Dat is: als $u^2 = 1/3$, waarbij de maximale fout is

$$
\begin{aligned}
\max(e_{\mathrm{B'3}}) &\approx 1/2(1-1/3)^2(1/3(1/4(x'_1 - 3x'_{b1}))^2) = 2/27(1/4(x'_1 - 3x'_{b1}))^2 \\
&= 2/27(\cos(\phi/2)\tan(\phi/4) - 1/2\sin(\phi/2))^2 \\
&= 2/27(\sin^2(\phi/4)\tan(\phi/4))^2 \approx 1/55296\phi^6
\end{aligned}
$$

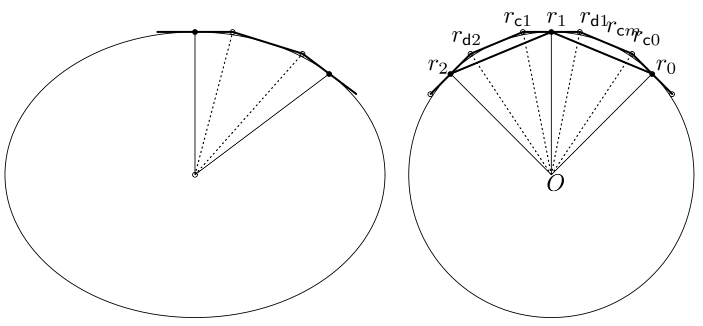
Door het oneven karakter van tan en sin wordt de fout nu van de orde ϕ^6, twee orden (!) beter dan bij de tweedegraads Bézier.
De fout kan zelfs verder verkleind worden door de hoek nog kleiner te nemen, zodat de benadering deels binnen, deels buiten de cirkel ligt met gemiddeld $< e_{\mathrm{B3}} >= 0$. Bedenkend dat het gemiddelde van een (even) macht van u is $< u^n >= 1/(n+1)$, geldt bij de optimale hoek:

$$
\begin{aligned}
(1-2/3+1/7)(1-(1/4(y_1+3y_{b1}))^2) &= (1/3-2/5+1/7)(1/4(x_1-3x_{b1}))^2 \\
1-(1/4(y_1+3y_{b1}))^2 &= 4/25(1/4(x_1-3x_{b1}))^2
\end{aligned}
$$

Het blijkt dat de oplossingshoek daarmee nauwelijks verandert.

Aangepaste derdegraads Bézier: 4 intervallen

Omdat de klassieke derdegraads Bézier 3 intervallen heeft, komt de omgeschreven veelhoek niet goed uit: de verbindingslijn van de binnenpunten raakt niet aan de cirkel. Door de omgeschreven veelhoek met halve hoek en dubbel aantal hoekpunten te nemen zal de verbindingslijn van de extra punten r_{c0} en r_{d1} raken aan de cirkel. De hoek bevat nu vier intervallen met intervalhoek $\phi/4$ (figuur 3).



Figuur 3.

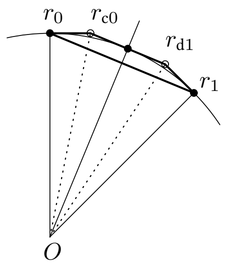
Deze omgeschreven veelhoek is dezelfde als die we zouden hebben gekregen wanneer we de hoek van de tweedegraads benadering zouden hebben gehalveerd. Het algoritme van de tweedegraads Bézier zal iets moeten worden aangepast. Uitgaande van de tweedegraadspunten r_0, r_{c0}, r_{cm} vinden we de punten van de tweede helft r_{cm}, r_{c1}, r_1:

$$\begin{aligned}(r_{d1} - r_{cm}) &= (r_{cm} - r_0) - (r_{c0} - r_0) \quad \text{relatief}\\ r_{d1} &= (r_{d1} - r_{cm}) + r_{cm} \quad \text{absoluut}\\ r_{cm} &= r_0 + (r_{cm} - r_0) \quad \text{absoluut}\\ r_m &= \lambda r_{d1} \quad \lambda = \cos^2(\phi/4) \quad \text{absoluut}\\ r_1 - r_{cm} &= 2(r_m - r_{cm}) \quad \text{relatief}\end{aligned} \tag{11}$$

Zo voortgaande is het algoritme voor de iteratie af te leiden.
In plaats van het aanpassen van de hoek zullen we nu de raaklijntijden aanpassen. De met raaklijntijden t=1/4 en t=3/4 *aangepaste derdegraads* Bézier benadering is:

(Lagrange)
$$r_{B3} = (1-2t)(1-t)^2 r_0 + 4t(1-t)^2 r_{c0} + 4t^2(1-t) r_{d1} + (2t-1)t^2 r_1$$
(veelterm t)
$$= r_0 + 4(r_{c0} - r_0)t + (3(r_1 + r_0) - 8(r_{c0} - r_0) + 4(r_{d1} - r_1))t^2 + (2(r_1 - r_0) - 4(r_{d1} - r_{c0}))t^3$$
(u tov midden)
$$= 1/2(r_{d1} + r_{c0}) + (1/4(r_1 - r_0) + 1/2(r_{d1} - r_{c0}))u + {} + (1/2(r_0 + r_1) - 1/2(r_{d1} + r_{c0}))u^2 + (1/4(r_1 - r_0) - 1/2(r_{d1} - r_{c0}))u^3$$

De coëficiënten van de veelterm in u zijn symmetrisch rond het midden r_{cm}:



1. $a_0 = r_{cm} = 1/2(r_{d1} + r_{c0})$ het raakpunt aan de cirkel;
2. $a_1 = 1/4(r_1 - r_0) + 1/2(r_{d1} - r_{c0})$, de numerieke eerste afgeleide in tangentiële koorderichting;
3. $a_2 = -(r_{cm} - r_m)$, de halve numeriek tweede afgeleide in radiële richting.
4. $a_3 = 1/4(r_1 - r_0) - 1/2(r_{d1} - r_{c0})$, een zesde numeriek derde afgeleide in tangentiële richting;

Ook hier: in de radiële y-richting is de benadering van de tweede graad, in de tangentiële x-richting van de derde graad, met sterk lineair deel:

$$\begin{aligned} x_{\mathrm{B3}} &= u(1/2x_1{+}x_{d1}) + u^3(1/2x_1{-}x_{d1}) = ux_1 - u(1{-}u^2)(1/2x_1{-}x_{d1}) \\ y_{\mathrm{B3}} &= 1 - u^2(1 - y_1) = y_1 + (1{-}u^2)(1 - y_1) \end{aligned} \tag{12}$$

De puntcoördinaten zijn: $r_1{=}(\sin(\phi_1), \cos(\phi_1))$, $r_{\mathrm{d}1}{=}(\tan(\phi_1 - \phi_B), 1)$ met hoeken $\phi_1{=}\phi/2$ en $\phi_B{=}\phi/4$. Voor de radiële fout in het kwadraat van de afstand vinden we voor de cirkel, door op dezelfde manier als bij de klassieke derdegraads Bézier gebruik te maken van $x_1^2 + y_1^2 = 1$ en $x_1 x_{d1} + 1.y_1 = 1$:

$$\begin{aligned} r_{\mathrm{B3}}^2 &= (y_{\mathrm{B3}}^2 + x_{\mathrm{B3}}^2) = (y_1^2 + u^2x_1^2) + u^2(1-u^2)^2(1/2x_1 - x_{d1})^2 + \\ &+ (1-u^2)(2y_1(1-y_1) + (1-u^2)(1-y_1)^2) - 2u^2(1-u^2)x_1(1/2x_1 - x_{d1})) \\ &= 1 + u^2(1-u^2)^2(1/2x_1 - x_{d1})^2 + (1-u^2)(-x_1^2 + 2y_1 - 2y_1^2 + (1-y_1^2)) - \\ &- (1-u^2)u^2(x_1^2 - 2x_1x_{d1} + (1-y_1)^2) = 1 + u^2(1-u^2)^2(1/2x_1 - x_{d1})^2 \end{aligned}$$

Niet toevallig is $1/2x_1{-}x_{d1} = 1/2\sin(\phi_1){-}\tan(\phi_1/2)$ gelijk aan de 'klassieke' aangepaste afstand $1/4(x_1'{-}3x_{b1}') = \cos(\phi_1)\tan(\phi_1/2){-}1/2\sin(\phi_1)$:

$$\max(e_{\mathrm{B3}}) = 2/27(\sin^2(\phi/4)\tan(\phi/4))^2 \approx 1/55296\phi^6$$

Merk op, dat de ordefout bij deze Béziermethode, met aangepaste raaklijntijd 1/4, veel eenvoudiger wordt verkregen dan bij de klassieke Béziermethode met raaklijntijd 1/3. Omgekeerd vinden we de aanpassingshoek: ga van raaklijntijd 1/4 naar 1/3 door de raaklijn P_1P_{d1} met een factor 4/3 te verlengen tot P_1P_{b1}. Gevolg: $\tan(\phi_B') = 4/3\tan(\phi/4)$, voilà!

In- en omgeschreven veelhoek bij hyperbool

Naast de cirkel en ellips is ook de hyperbool een kegelsnede. We zullen nu analytisch aantonen dat de veelhoeken constructie ook toepasbaar is bij de hyperbool, zij het met enige aanpassing. Gedachtig de mogelijkheden van herschaling en symmetrie, zoals beschreven in de inleiding, gaan we, zonder verlies van algemeenheid, uit van de standaardhyperbool in de parameter beschrijving vanuit de oorsprong (ook dat is nog lastig genoeg!):

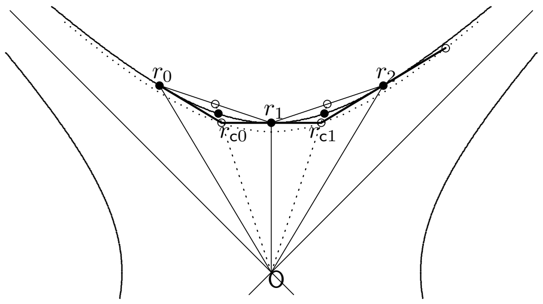
$$x = \cosh t \quad y = \sinh t \Leftrightarrow x^2 - y^2 = 1 \tag{13}$$

Bij de parameter t behoort geometrisch de richting(scoëfficiënt) m van de straal

$$m = y/x = \tan\varphi = \tanh t \tag{14}$$

$$x^2 = \frac{1}{1-m^2} \quad y^2 = \frac{m^2}{1-m^2} \tag{15}$$

Naar analogie van de constructie van de veelhoeken bij de cirkel gaan we, in plaats van de hoek φ, uit van de parameter t. In dit geval leidt dat tot oneindig veel hoekpunten om de hyperbool te beschrijven. Uitgaande van een punt van de hyperbool r_0, bij parameter t_0, of richting m_0, vinden we de hoekpunten $r_1, r_2, \ldots$ van de ingeschreven veelhoek als de punten met hoeken die veelvouden van veelhoek-hoek ϕ verder liggen. De snijpunten van de raaklijnen in de hoekpunten van de ingeschreven veelhoek vormen de hoekpunten r_{c} van de omgeschreven veelhoek (figuur 4). Het is eenvoudig in te zien dat, als de richting van de straal m is, de richting van de raaklijn $1/m$ is. Immers door differentiëren van de hyperboolvergelijking in x, y:



Figuur 4.

$$2x - 2yy' = 0 \Rightarrow y' = x/y$$

Bezie nu de driehoek van punten $r_0 r_1 r_c$, waarin $r_0 r_1$ een koorde van de hyperbool is en r_a het snijpunt van de raaklijnen in r_0 en r_1. Op de hyperbool is een punt $r_{1/2}$ 'halverwege' waar de raaklijn evenwijdig loopt aan de koorde. We zullen eerst laten zien dat, net als bij de ellips, de straal naar het raakpunt iedere lijn evenwijdig aan de raaklijn in de hyperbool doormidden deelt. Stel dat de koorde en raaklijn richting $1/m_{1/2}$ hebben, zodat in het raakpunt de straal een richting $m_{1/2}$ heeft. De straal naar $r_{1/2}$ snijdt de koorde in het halverwege punt $r_m = (r_0 + r_1)/2$, omdat de eindpunten van de koorde op de hyperbool liggen:

$$x_0^2 - y_0^2 = x_1^2 - y_1^2 = 1 \Leftrightarrow (y_1^2 - y_0^2) = (x_1^2 - x_0^2)$$
$$(y_1 - y_0)(y_1 + y_0) = (x_1 - x_0)(x_1 + x_0) \Leftrightarrow (y_1 + y_0) = m_{1/2}(x_1 + x_0)$$
$$(16) \quad r_m = \mu r_{1/2}$$

wat beduidt dat het midden van de koorde r_m op de straal naar $r_{1/2}$ ligt. In het bewijs is gebruikt dat de koorderichting $1/m_{1/2}$ is. Later zullen we de constante waarde van μ bepalen. Inderdaad is $t_{1/2}$ halverwege t_0 en t_1:

$$\tanh t_{1/2} = \frac{\sinh t_0 + \sinh t_1}{\cosh t_0 + \cosh t_1}$$
$$= \frac{2\sinh((t_0 + t_1)/2)\cosh((t_0 - t_1)/2)}{2\cosh((t_0 + t_1)/2)\cosh((t_0 - t_1)/2)} = \tanh((t_0 + t_1)/2)$$
$$(17) \quad t_{1/2} = (t_0 + t_1)/2 = t_m$$

We zullen nu laten zien dat, net als bij de ellips, ook het snijpunt r_c van de raaklijnen aan de eindpunten van de koorde op de straal naar het raakpunt $r_{1/2}$ ligt. Het snijpunt ligt op twee raaklijnen, dus voldoet aan twee vergelijkingen:

$$y_c - y_0 = (x_c - x_0)/m_0 \wedge y_c - y_1 = (x_c - x_1)/m_1$$
$$x_c - m_0 y_c = (1 - m_0^2)x_0 = 1/x_0 \wedge x_c - m_1 y_c = (1 - m_1^2)x_1 = 1/x_1$$
$$(m_1 - m_0)y_c = (1/x_0 - 1/x_1) \wedge 2x_c = (m_0 + m_1)y_c + (1/x_0 + 1/x_1)$$
$$x_0 x_1 (m_1 - m_0) y_c = x_1 - x_0 \wedge x_0 x_1 (m_1 - m_0) x_c = m_1 x_1 - m_0 x_0$$
$$y_c/x_c = (x_1 - x_0)/(y_1 - y_0) = m_{1/2}$$
$$(18) \quad r_\text{c} = (\mu/\lambda) r_{1/2} = r_m/\lambda$$

wat betekent dat het snijpunt van de raaklijnen r_c op de straal naar het raakpunt $r_{1/2}$ ligt. (bewijs moet beter kunnen; meetkundig?)
Liggen de hoekpunten r_c van de omgeschreven veelhoek ook op een hyperbool? Om die vraag te beantwoorden hebben we waarden van μ en λ nodig. In de laatste vergelijkingen zijn x_c en y_c uitgedrukt in x_0, x_1, m_0 en m_1. Deze drukken we uit in de hyperbolische variabele t:

$$\begin{aligned}
x_0x_1(m_1-m_0) &= \cosh t_0 \cosh t_1 (\tanh t_1 - \tanh t_0)\\
&= \sinh t_1 \cosh t_0 - \sinh t_0 \cosh t_1 = \sinh(t_1-t_0)\\
x_1-x_0 &= \cosh t_1 - \cosh t_0 = 2\sinh((t_0+t_1)/2)\sinh((t_1-t_0)/2)\\
y_1-y_0 &= \sinh t_1 - \sinh t_0 = 2\cosh((t_0+t_1)/2)\sinh((t_1-t_2)/2)
\end{aligned}$$

Voor t_1-t_0, het verschil van opvolgende t-waarden, nemen we de constante waarde ϕ. We concluderen dat de hoekpunten r_c van de omgeschreven veelhoek op een evenwijdige, iets grotere, hyperbool liggen met parametervergelijking:

$$x_c = (\mu/\lambda)\cosh t \quad y_c = (\mu/\lambda)\sinh t$$
$$\lambda/\mu = \sinh\phi/2\sinh(\phi/2) = \cosh(\phi/2)$$

De eigenaardige 'straal' μ/λ is naar analogie van de ellips. Let bij de waarde van de 'straal'van de ellips op de overeenkomst met de ellipswaarde.
Omdat de punten van de omgeschreven veelhoek op een hyperbool liggen, geldt de middeneigenschap nu ook voor de raaklijnen in de hoekpunten van de ingeschreven veelhoek, die nu immers koorden zijn in de buitenste hyperbool van r_c.
Het algoritme voor het tekenen van een ellips kan gebruikt worden voor de hyperbool. We missen nog de waarde van $\mu = |r_m|/|r_{1/2}| = y_m/y_{1/2} = x_m/x_{1/2}$:

$$\begin{aligned}
\mu &= \frac{(x_0+x_1)/2}{x_{1/2}} = \frac{\cosh t_0 + \cosh t_1}{2\cosh t_{1/2}}\\
&= \frac{2\cosh((t_0+t_1)/2)\cosh((t_1-t_0)/2)}{2\cosh((t_0+t_1)/2)}
\end{aligned}$$

(19) $$\mu = \cosh(\phi/2) \quad \lambda = \mu^2$$

Merk op, dat het enige verschil met de ellips is, dat we $\sin, \cos$ moeten vervangen door $\sinh, \cosh$.

In- en omgeschreven veelhoek bij parabool

Als het benaderen van de cirkel, ellips en hyperbool goed kan met in-en omgeschreven veelhoek, dan moet dat bij de parabool, de enig overgebleven kegelsnede, ook kunnen! Ook de parabool onderzoeken we analytisch. Welnu, door verplaatsen, vergroten, verdraaien, uitrekken is iedere parabool te zien als de standaard parabool:

(20) $$y = x^2$$

De parameter t van de parabool nemen we (evenredig aan) xt, en dus $t = x$. De parameter van de hoekpunten van de ingeschreven veelhoek heeft stapgrootte ϕ, zodat $x_1 = x_0 + \phi$. Voor de richting(scoëfficiënt) m van de raaklijn in raakpunt r geldt:

$$m = y' = 2x$$

Laat r_0, parameter x_0, en r_1, parameter x_1, twee opeenvolgende hoekpunten zijn van de ingeschreven veelhoek. Dan is $r_{1/2}$ het punt op de kromme met parameterwaarde

$x_{1/2} = (x_0+x_1)/2 = x_m$, waarbij $r_m = (r_0+r_1)/2$ het punt halverwege de koorde r_0r_1. De richting van de raaklijn in $r_{1/2}$ is $m_{1/2} = 2x_{1/2} = 2x_m$ blijkt gelijk aan de richting van de koorde r_0r_1:

(21) $$(y_1 - y_0)/(x_1 - x_0) = (x_1^2 - x_0^2)/(x_1 - x_0) = x_1 + x_0 = 2x_m$$

Dus liggen r_m en $r_{1/2}$ op een lijn evenwijdig aan de as van de parabool, de y-as. Voor de afstand (richting y-as) tussen r_m en het raakpunt $r_{1/2}$ geldt:

(22)
$$y_m - y_{1/2} = (y_1+y_0)/2 - x_{1/2}^2 = (x_1^2+x_0^2)/2 - (x_1+x_0)^2/4 = (x_1-x_0)^2/4 = (\phi/2)^2$$

In woorden: de middens van de koorden van de ingeschreven veelhoek liggen op de parabool evenwijdig aan de kromme op een afstand $(\phi/2)^2$ langs de as hoger.
De raaklijnen in r_0 en r_1 snijden elkaar in het punt r_{c0}, een hoekpunt van de omgeschreven veelhoek. We zullen laten zien dat $r_{1/2}$ juist halverwege r_m en r_{c0} ligt. Immers, voor het snijpunt van de raaklijnen in r_0 en r_1 geldt

$$\begin{aligned}
y_{c0} &= y_0 + m_0(x_{c0} - x_0) = y_1 + m_1(x_{c0} - x_1)\\
x_{c0} &= ((m_1x_1 - m_0x_0) - (y_1 - y_0))/(m_1 - m_0)\\
&= (2(x_1^2 - x_0^2) - (x_1^2 - x_0^2))/2(x_1 - x_0)\\
&= (x_1^2 - x_0^2)/2(x_1 - x_0) = (x_1 + x_0)/2 = x_{1/2}\\
y_{c0} &= y_0 + m_0(x_1 - x_0)/2 = x_0^2 + 2x_0(x_1 - x_0)/2 = x_0x_1\\
y_{c0} + y_m &= x_0x_1 + (x_0^2 + x_1^2)/2 = (x_0 + x_1)^2/2 = 2x_m^2 = 2y_{1/2}
\end{aligned}$$

(23) $$r_{c0} + r_m = 2r_{1/2}$$

Anders gezegd: de hoekpunten van de omgeschreven veelhoek r_c liggen op de parabool evenwijdig aan de kromme op een afstand $(\phi/2)^2$ langs de y-as lager.
Het bijzondere van de parabool numeriek is, dat als we de kwadratische Bézier benadering nemen, de benadering van het eerste interval direct de gehele paraboolkromme geeft. De kwadratische benadering is exact. In het eerste interval, tussen de punten van de kromme r_0, met $t = t_0$, en r_1, met $t_1 = t_0 + \phi$, is alleen een algoritme nodig voor het bepalen van het raaklijnpunt r_{c0}. Aangezien we in het algemeen niet weten waar de as van de parabool is, gebruiken we het punt van de kromme $r_{1/2}$, met $t_{1/2} = t_0 + \phi/2$, en het middenpunt r_m daarvoor:

(24) $$\begin{aligned}
r_m &= (r_0 + r_1)/2 \quad \text{definitie}\\
r_{c0} - r_m &= 2(r_{1/2} - r_m) \quad \text{volgens (23)}
\end{aligned}$$

Merk op, dat de richting van de as van de parabool gegeven wordt door (het tegengestelde van) het laatste verschil.

C.M. Fortuin
van Deldenpad 8
6862DC Oosterbeek NL
c.m.fortuin@hccnet.nl

Pythagoras Trees in PostScript

Fractal Geometry 0

Abstract
Pythagoras Trees are drawn elegantly in PostScript, varied by randomness, colour and the use of curves. Lindenmayer production rules for systematic PS program development are enriched by PS concepts.

Keywords
2.5D, Adobe, ALGOL, art, backtracking, BASIC, CWI, Deubert, EPSF, FIFO, fractal, fractal geometry, IDE, Julia set, Lauwerier, Lévy, LIFO, Lindenmayer, minimal encapsulated PostScript, minimal plain TeX, Pascal triangle, Photoshop, production rule, Pythagoras Tree, PSlib, self-similarity, Sierpiński sieve, sentinel, TEXworks, (adaptable) user space, Word

Contents

Introduction

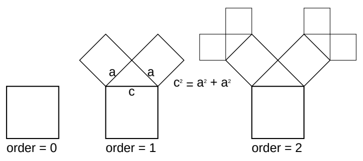
In the 70s, I was still a student, CWI treated its employees and relations on a calendar — human-meets-plotter — with illustrations made by the Calcomp 565 plotter, among others 3 Pythagoras Trees. At the time the Pythagoras Tree was a programming challenge in ALGOL60/68, the programming languages in use at CWI. With Metafont's path data structure in the 80s it was interesting to program the simplified Tree again. (The swapping of parts was very fast!) With PostScript (PS for short) and its powerful adaptable User Space (US for short) functionality, I was curious whether the Pythagoras Tree family could be programmed elegantly in PS as EPSF.

This note is about programming the Pythagoras Tree family recursively in PS with the use of the so-called adaptable User Space facility of PS, biased by production rules. It can be seen as an extension to my BachoTEX 2011 Pearl. In the footsteps of Lauwerier, the reader is invited to experiment with the PS programs, of which `defs` are supplied in my `PSlib.eps` library, which I'll send on request.

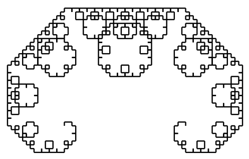
The word FRACTAL did not yet exist for us, although the spirit was all over, endlessly repeated geometric figures buzzed around, apparently, with BASIC programs by Lauwerier(1987).

A few of Lauwerier's BASIC programs: `BOOM3`, `SIER`, `PYTHB1`, `PYTHB3` have been converted into PS `defs`; in the included BASIC programs I have omitted the screen settings. Lauwerier's program `MONDRIAAN` has been elaborated upon in my à la Mondriaan note, MAPS 41 p79–90.

Definition
The Pythagoras Tree is a plane fractal constructed from squares. The Construction of the Pythagoras tree begins with a square. Upon this square are constructed two squares, each scaled by $1/\sqrt{2}$, and rotated $\pm 45°$. The same procedure is repeated on the two smaller squares, ad infinitum. The accompanying illustration shows the first few steps in the construction process.



Origin Lauwerier(1987) mentions Bosman, A.E(1957) *Het wondere onderzoekingsveld van de vlakke meetkunde*, as source of Pythagoras Trees. The name comes from the Tree of order 1, which is used in the proof of the Pythagoras theorem $a^2 + b^2 = c^2$: the sum of the surfaces of the descendant squares equals the surface of the basic square, which in terms of the sides of the spurious rectangular triangle is generally formulated as: the square of the hypotenuse equals the sum of the squares of the legs.



← Lévy C curve

Julia set →

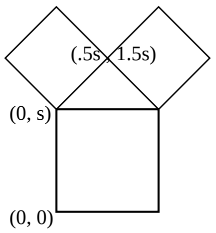
Properties The sum of the surfaces of the squares $\rightarrow \infty$ as does the circumference. The kind of infinity is characterized by the fractal dimension notion.[1] The circumference has fractal dimension 2, a local plane-filling curve.

If the largest square has a size of $L \times L$, the entire Pythagoras tree fits snugly inside a rectangle of size $6L \times 4L$. The finer details of the tree resemble the Lévy C curve. Lauwerier(1988) calls the blossom a Julia set.

The production rule for a Pythagoras Tree of order n reads

$$P_n = \Box_s \oplus [T_{(0,s)} R^{45} S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})} P_{n-1}] \oplus [T_{(\frac{s}{2},\frac{3s}{2})} R^{-45} S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})} P_{n-1}],$$

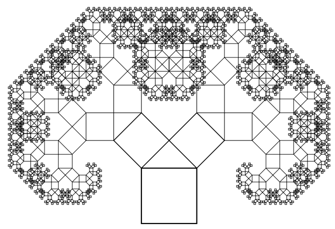
with P_n the Pythagoras Tree of order n, $\oplus$ denotes splice operator, means add properly, [means store graphics state and open a new one,] means close current graphics state and recall previous, R^{45} means rotate US 45° in the PS sense, $S_{a,b}$ means scale US by a and b, $T_{a,b}$ means translate US by a and b, $\Box_s$ means draw square with side s. In the Lindenmayer system [means start a new branch, remotely similar to a new graphics state or recursion level.



Why program the classical example in PS and share it? IMHO, it is an interesting non-trivial example of (recursive) programming, and … it demonstrates the power of PS's functionality to transform User Space. Moreover … EPSF is a (plain) TEXies graphics companion. PS abstracts from concrete printer devices, has useful programming features and is maintained by Adobe, which entails continuity.

Besides, I don't know how to include elegantly the results of BASIC programs in my publications.

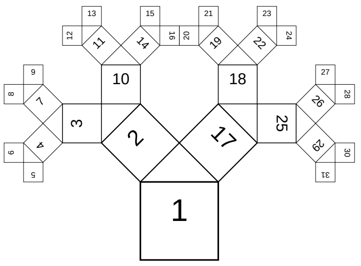
On the WWW I did not find Algol nor PASCAL nor … versions and only those by John Deubert in PS. From Lauwerier's books I picked up the Trees in BASIC. My Algol68 program — lost alas, after so many years — was definitely not so elegant as my current PS.



The PostScript program

The Tree is a collection of scaled and rotated squares placed such that each parent square and its descendants enclose a rectangular triangle. The program is my favourite, non-trivial example of translating and rotating user space in PS. All one has to program is drawing a square and place it scaled and rotated at the right place, repetitively. Backtracking and the bookkeeping of auxiliaries is implicit.

The above given production rule reads: the Tree of order n consists of the basic square and 2 Trees of 1 order lower, translated, scaled and rotated, adhering to self-similarity.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Pythagoras Tree of squares
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -125 -20 175 200
%%BeginSetup                                  %crops to the prescribed BB
%%EndSetup                                    %when processed by Acrobat Pro
%%BeginProlog                                 %defs
/drawsquare{0 0 s s rectstroke}def            %basic square
/pythagorastree{drawsquare                    %paint the square to the current page
1 sub  dup 0 gt                               %lower order on stack
{gsave 0 s translate 45 rotate .7071 dup scale%transform user space
 pythagorastree                               %play it again, Sam
 grestore
 .5 s mul 1.5 s mul translate
                    -45 rotate .7071 dup scale%transform user space
 pythagorastree                               %play it again, Sam
}if 1 add                                     %adjust order on stack
 } bind def
%%EndProlog
%
%   Program ---the script---
%
/s 50 def                                     %size of the side of the square
11 pythagorastree  pop                        %order 11
showpage
%%EOF
```

The order of traversal is given by the numbers in the squares in the lower Tree, and shows that the (recursive) algorithm is a backtracking process.

To run the program store the file with extension .eps (or .ps), right-mouse click the thumbnail of the file and choose the option convert to Adobe PDF in the pop-up menu. That is all when you have installed Acrobat Pro. I also used Adobe Illustrator and PSview. The latter just by double clicking the filename upon which the command window opened and a little later PSview.[2]

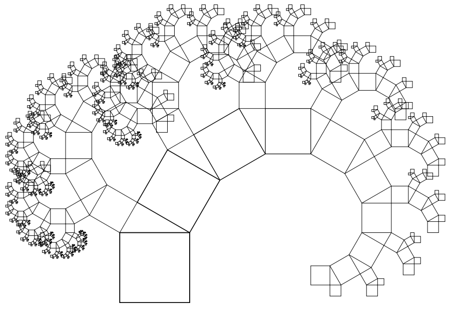
Variations

The CWI calendar contains also the oblique tree and the alternating (Christmas) tree as variations. Lauwerier(1987) mentions more realistic trees, as well as the simplified tree where the square is reduced to a line or a pair of parallel lines.

Realistic tree

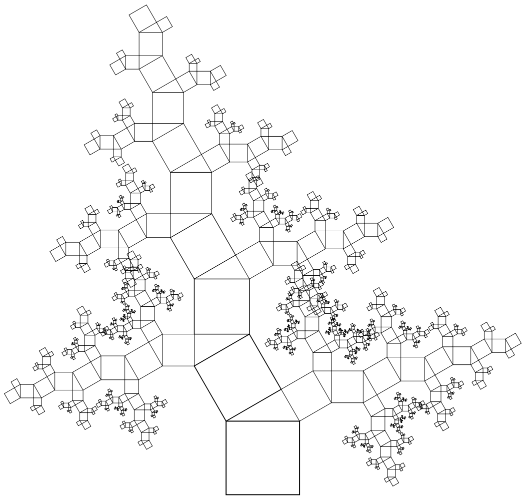
Oblique tree Let the skewness be denoted by ϕ (60°, say). We have to translate, rotate and scale before each recursive invoke. The left descendant is placed, as in the symmetrical case, at $(0, s)$ with the user space rotated by ϕ, and scaled by $(\cos\phi, \cos\phi)$. The right descendant is placed at $s(\cos^2\phi, \sin\phi\cos\phi{+}1)$ with the user space rotated by $\phi - 90^\circ$, and scaled by $(\sin\phi, \sin\phi)$. The tree is full of logarithmic spirals. The self-similarity of logarithmic spirals captivated Bernoulli. His epitaph reads: *eadem mutata resurgo* (hoewel veranderd zal ik als dezelfde herrijzen).

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Oblique Pythagoras Tree of squares
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -135 0 350 325
%%BeginSetup
%%EndSetup
%%BeginProlog
/drawsquare{0 0 s s rectstroke}def
/OPythagorastree{%on stack integer (order>=0) ==> Oblique Tree
                 %Global variables: s phi, and the def drawsquare
drawsquare
1 sub dup 0 gt
{gsave
 0 s translate     phi rotate phi cos dup scale OPythagorastree
 grestore
 s phi cos dup mul mul s phi sin phi cos mul mul s add translate
            phi 90 sub rotate phi sin dup scale OPythagorastree
}if 1 add } bind def
%%Endprolog
%
%   Program ---the script---
%
/phi 60 def /s 75 def %globals
10 OPythagorastree pop%order 10
showpage
%%EOF
```



Christmas tree Similar to the oblique tree, but at each level in the recursion the current angle ϕ is changed into its complement at the beginning and at the end. Let $\phi = 60^\circ$ at the start.

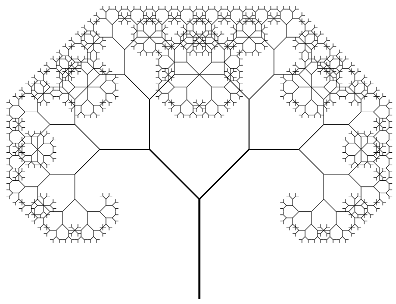
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: (Pythagoras) X-mas Tree of squares
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -235 -2 305 500
%%BeginSetup
%%EndSetup
%%BeginProlog
/drawsquare{0 0 s s rectstroke}def
/Xmastree{%on stack integer (order>=0) ==> XmasTree
          %Globals: s phi, and def drawsquare
drawsquare %draw the square
1 sub dup 0 gt
{gsave /phi 90 phi sub def
 0 s translate  phi rotate phi cos dup scale Xmastree
 grestore
 s phi cos dup mul mul  s phi sin phi cos mul mul s add translate
         phi 90 sub rotate phi sin dup scale Xmastree
 /phi 90 phi sub def
}if 1 add } bind def
%%Endprolog
%
%   Program ---the script---
%
/phi 60 def /s 75 def %globals
10 Xmastree pop       %order 10
showpage
%%EOF
```



Simplified Tree

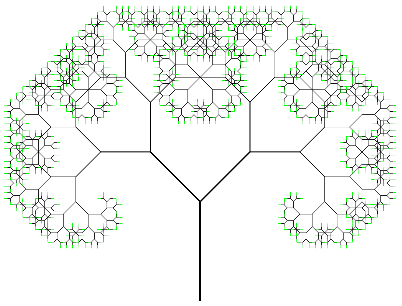
We may simplify the tree by drawing a line instead of a square. This implies that the `translate` — first adaptation of the User Space — is invoked at the beginning after drawing the line. The rotation and scaling adaptations of the user space remain at the same place.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Simplified Pythagoras Tree
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -125 -20 175 200
%%BeginSetup
%%EndSetup
%%BeginProlog
/line{0 0 moveto 0 s lineto stroke}def
/linetree{line 0 s translate     %draw the line and transform user space
1 sub  dup 0 gt                  %lower the order on the stack
{gsave 45 rotate .7071 dup scale %transform  user space
 linetree                        %play it again, Sam
 grestore
      -45 rotate .7071 dup scale %transform user space
 linetree                        %play it again, Sam
}if 1 add                        %adjust the order on the stack
} bind def
%%EndProlog
%
%   Program ---the script---
%
/s 50 def                        %size of side
11 linetree  pop                 %order 11
showpage
%%EOF
```



Simplified Tree with simplest green leaves (green lines) The end lines (leaves) can be coloured (by a shade of) green.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Simplified Pythagoras Tree with green leaves
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -125 -20 175 200
%%BeginSetup
%%EndSetup
%%BeginProlog
/line{0 0 moveto 0 s lineto stroke}def
/linetree{line 0 s translate     %draw the line and transform user space
1 sub  dup 0 gt                  %lower the order on the stack
 {gsave 45 rotate .7071 dup scale%transform user space
  linetree                       %play it again, Sam
  grestore
       -45 rotate .7071 dup scale%transform user space
  linetree                       %play it again, Sam
  }if 1 add                      %adjust the order on the stack
  gsave
  0 1 0 setrgbcolor line         %green leaves (simple)
grestore} bind def
%%EndProlog
%
%   Program ---the script---
%
/s 50 def                        %size of side
11 linetree  pop                 %order 11
showpage
%%EOF
```

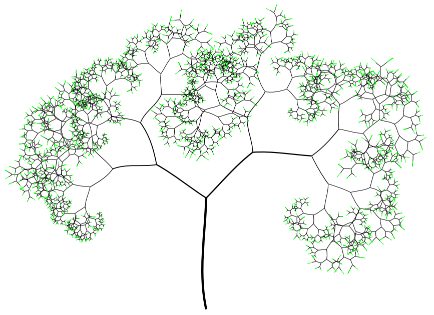


Simplified randomized Tree We might transform the user space irregularly, by some spread variation around 45°, and also with varying randomly the scale i.e. the length s of the line. Because of the diminishing scale the branches become thinner, automatically.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Randomized simplified Pythagoras Tree with green leaves
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -125 -20 175 200
%%BeginSetup
%%EndSetup
%%BeginProlog
/maxint 2147483647 def                 %2^31-1
/line{0 0 moveto 0 s lineto stroke}def
/spread{% value maxspread ==> new value
   rand //maxint div .5 sub 2 mul mul 1 add mul } bind def
/ranlinetree{line 0 s translate        %draw the line and transform user space
1 sub  dup 0 gt                        %lower the order on the stack
{gsave 45 .5 spread rotate .7071 .1 spread dup scale %transfor user space
 ranlinetree                           %play it again, Sam
 grestore
      -45 .5 spread rotate .7071 .1 spread dup scale     %transformed user space
 ranlinetree                           %play it again, Sam
}if 1 add                              %adjust the order on the stack
 gsave 0 1 0 setrgbcolor line grestore %green leaves
} bind def
%%EndProlog
%
%   Program ---the script---
%
/s 50 def                              %initial size of line
22121943 srand                         %initialize random number sequence
11 ranlinetree  pop                    %order 11
showpage
%%EOF
```

Splined simplified randomized green Tree We might further vary by drawing a spread of splines, see Appendix 1 about splines, instead of straight lines, and approximate a real tree more closely.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Spline randomized Pythagoras Tree with green leaves
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -110 -20 110 150
%%BeginSetup
%%EndSetup
%%BeginProlog
/maxint 2147483647 def%2^31-1
/spline{0 0 moveto 0 .1 s mul spread .33 s mul
                   0 .1 s mul spread .66 s mul
                   0 s curveto stroke} bind def
/spread{% value maxspread ==> new value
   rand //maxint div .5 sub 2 mul mul add } bind def
/splinetree{spline 0 s translate       %draw the line and transform user space
1 sub  dup 0 gt
{gsave 45 15 spread rotate .7071 .2 spread  dup scale %transform user space
 splinetree                            %play it again, Sam
 grestore
      -45 15 spread rotate .7071 .2 spread dup scale  %transform user space
 splinetree                            %play it again, Sam
}if 1 add
gsave 0 1 0 setrgbcolor spline grestore %green leaves
} bind def
%%EndProlog
%
%    Program ---the script---
%
/s 50 def                              %initial size of spline
22121943 srand                         %initialize random number sequence
12 splinetree  pop                     %order 12
showpage
%%EOF
```

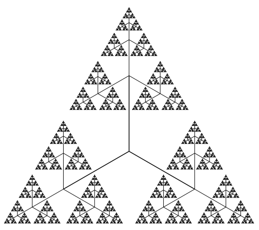


What if we don't vary randomly but according to certain rules of growth? Maybe the program can be adapted and simulate real trees, avoiding 3D and projection?

Trinary Trees

One might add another branch and create trinary trees, as John Deubert did (see later) and vary more. Below the classical symmetrical trinary tree.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Trinary Tree
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BeginProlog
/drawline{0 0 moveto 0 s rlineto currentpoint stroke translate}def
/trinarytree{%order on stack; s size (global) ==> Trinary Tree
1 sub  dup 0 gt
 {gsave  drawline           .475 dup scale %transform user space
  trinarytree                              %play it again, Sam
  grestore
  gsave 120 rotate drawline .475 dup scale %transform user space
  trinarytree                              %play it again, Sam
  grestore
  gsave 240 rotate drawline .475 dup scale %transform user space
  trinarytree                              %play it again, Sam
  grestore
 }if 1 add } def
%%Endprolog
```



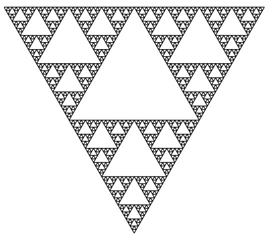
...

Lauwerier's trinary tree, `BOOM3`, contains an error: in the `LINE`s the index `k` is used instead of `m`.

Sierpiński's sieve is related to the trinary tree, and constructed by deleting the inner `half-triangles', which are obtained by connecting the midpoints of the sides, recursively ad infinitum.

The idea behind the code below is to identify each triangle with a trinary number. Possibly because it was a side-step Lauwerier did not provide a more efficient backtracking variant.

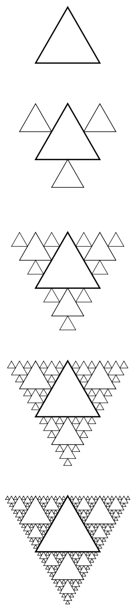
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski Sieve
%%Author: H.A. Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl
%%Date:  April 2011
%%BeginProlog
/sierpinski%p (order)==> Sierpinski triangle fractal
{/p exch def /t p 1 add array def /a 1.7320508 def
 1.415 setmiterlimit
 0 1 p{/m exch def
  0 1 3 m exp 1 sub{/n exch def
    /n1 n cvi def
    0 1 m 1 sub{/l exch def
      t l n1 3 mod put /n1 n1 3 idiv def
      }for%l
    /x 0 def /y 0 def
    0 1 m 1 sub{/k exch def
      /x x 4 t k get mul 1 add 30 mul cos  2 k exp div add def
      /y y 4 t k get mul 1 add 30 mul sin  2 k exp div add def
      }for%k
    /u1 x a 2 m 1 add exp div add def /u2 x a 2 m 1 add exp div sub def
    /v1 y 1 2 m 1 add exp div sub def /v2 y 1 2 m       exp div add def
    u1 s v1 s moveto x s v2 s lineto u2 s v1 s lineto u1 s v1 s lineto
    }for%n
}for%m
}bind def
%%EndProlog
%
%    Program ---the script---
%
/s{100 mul }def % scaling
5 sierpinski  stroke
a neg s 1 s moveto a s 1 s lineto 0 -2 s lineto closepath stroke
showpage
%%EOF
```



A recursive variant is similar to the coding and production rule of the Pythagoras Tree: draw a collection of isosceles triangles, properly placed and scaled. More concrete. The order 1 is an isosceles triangle with side s, of which I assumed the left corner in (0,0). The order 2 is a triangle of order 1 where scaled triangles at the sides are added. Repeat this process of placing scaled isosceles triangles for each new triangle.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski triangle recursive, Feb2012
%%Author: Kees van der Laan
%% Affiliation: kisa1@xs4all.nl
%%BoundingBox:-26 -425 76 46
%%BeginSetup
%%EndSetup
```

```
%%BeginProlog
/y{3 sqrt 4 div s mul}def
/SierTri{%on stack order >=1 ==> Sierpinski triangle fractal ( s global)
1 sub  dup 0 ge
{gsave 0 0 moveto s 0 lineto s 2 div .869 s mul lineto closepath stroke grestore%order1
 gsave s 4   div y neg translate .5 dup scale SierTri grestore
 gsave s .75 mul y     translate .5 dup scale SierTri grestore
 gsave s -4  div y     translate .5 dup scale SierTri grestore
 }if 1 add
} def
%%EndProlog
%
%Program ---the script---
%
/s 50 def %size of line segment
                       1 SierTri pop
0 -1.5 s mul translate 2 SierTri pop
0 -2 s   mul translate 3 SierTri pop
0 -2 s   mul translate 4 SierTri pop
0 -2.1 s mul translate 5 SierTri pop
showpage
%%EOF
```

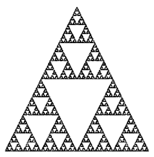


By iterated function system

A rather unusual way to generate the Sierpiński sieve is by the functions, L, R, T, given below, each applied with equal probability, Lauwerier(1990, p34).

$$\begin{pmatrix}x'\\y'\end{pmatrix}\overset{L}{=}.5\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+.5\begin{pmatrix}-1\\-1\end{pmatrix}\ ;\ \begin{pmatrix}x'\\y'\end{pmatrix}\overset{R}{=}.5\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+.5\begin{pmatrix}1\\-1\end{pmatrix}\ ;\ \begin{pmatrix}x'\\y'\end{pmatrix}\overset{T}{=}.5\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+.5\begin{pmatrix}0\\1\end{pmatrix}.$$

```
%!PS-Adobe-3.0 EPSF-3.0
%%Name: reduction of squares: SQAURE1, Sierpinsky sieve
%%Author: HA Lauwerier(1990) Een wereld van Fractals.
%%BoundingBox: -200 -200 200 200
%%BeginSetup
%%EndSetup
%%DocumentFonts: Helvetica
/scale{100 mul}def
/Helvetica 7 selectfont
/q1  715827882 def%2147483647/3 = maxint/3
/q2 1431655764 def
/x 0 def /y 0 def
22121943 srand
1 1 10000 {/i exch def /r rand def
 r q1 lt {/x x .5 mul -.5 add def
          /y y .5 mul -.5 add def}
         {r q2 lt {/x x .5 mul  .5 add def
                   /y y .5 mul -.5 add def}
                  {/x x  .5 mul def
                   /y y  .5 mul .5 add def}
          ifelse}

      ifelse
 i 16 gt{x scale y scale moveto (.) show
         x neg scale y scale moveto (.) show}if
}for
showpage
%%EOF
```



Randomized Sierpiński triangle

Peitgen c.s.(2004) mentions the addition of randomness to the Sierpiński triangle. Nice.

Pascal triangle and Sierpiński triangle

Pascal's triangle gives the binomial coefficients for the sum representation of $(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k, n \geqslant 0$. In PWT(1995) for low n Pascal's triangle was obtained by TEX alone simply by.

```
$$\displaylines{1\cr
 1\quad1\cr
 1\quad2\quad1\cr
 1\quad3\quad3\quad1\cr}$$
```

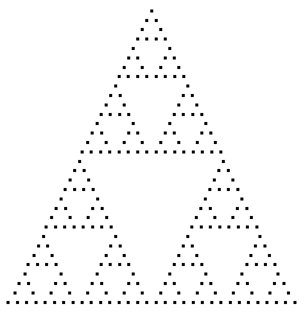
yields

```
      1
    1   1
  1   2   1
1   3   3   1
```

For general order n the macro for the Pascal triangle were the entries are calculated by the recursion $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}, n \geqslant k \geqslant 1$, given in PWT reads

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

Peitgen c.s.(2004) mentions the relationship between the Sierpiński triangle and the PASCAL triangle, when odd entries are blackened. Intriguing. Adaptation of the above code for the purpose is not that difficult.



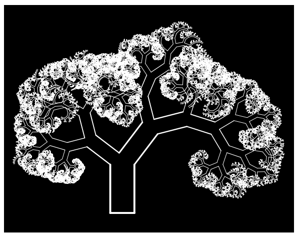
```
\newcount\n \newcount\rcnt \newcount\ccnt \newcount\tableentry \newcount\prev
\let\ea=\expandafter
%
\def\pascal#1{\n#1 \def\0{1} \ccnt1
\loop\ea\xdef\csname\the\ccnt\endcsname{0} %auxiliary sentinel
\ifnum\ccnt<\n \advance\ccnt1\repeat
\rcnt0 \ccnt0 \displaylines{\rows}}
%
\def\rows{\global\advance\rcnt1 \ifnum\rcnt>\n \swor\fi \nxtrow\rows}

\def\swor#1\rows{\fi}
```

```
%
\def\nxtrow{\black \ccnt1 \prev1
\loop\ifnum\ccnt<\rcnt \tableentry\prev \prev\csname\the\ccnt\endcsname
\advance\tableentry\prev %recursive addition
\ea\xdef\csname\the\ccnt\endcsname{\the\tableentry} %store the new entry
\quad\ifodd\tableentry\black\else\white\fi\advance\ccnt1 %black the entry
\repeat\cr}
\def\black{\vrule width1.1ex height1.1ex\relax}
\def\white{\hskip1ex}
$$\pascal{32}$$
\bye
```

Tree with stem

Lauwerier shows Trees with a stem, which in my terminology translate into drawing 2 lines instead of a square. Lauwerier creates the oblique tree by drawing 2 lines of unequal length, varied randomly, `PYTHBS`.

I did vary the angle randomly within a spread. The programming was more difficult because I had to store the randomized angles explicitly in an array.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Pythagorean Tree of lines
%%Author: Kees van der Laan, kisa1@xs4all.nl
%%BoundingBox: -110 -20 190 215
%%BeginSetup
%%EndSetup
%%BeginProlog
/maxint 2147483647 def%2^31-1
0 setgray -110 -20 300 235 rectfill %Mimics BoundingBox
/spread{% value maxspread ==> new value
  rand maxint div .5 sub 2 mul mul add }def
/d+1{/depth depth 1 add def} def
/d-1{/depth depth 1 sub def} def
/lines {0 0 moveto 0 l rlineto
        s 0 moveto 0 l rlineto stroke} def
/pythstem{%Global variables: s l phi
          %on stack integer (order>=0)
lines  %draw the lines
1 sub dup 0 gt
  {gsave /phi phi 10 spread def aphi depth phi put
  0 l translate    phi rotate        phi cos dup scale d+1 pythstem d-1
  grestore /phi aphi depth get def
  gsave
  s phi cos dup mul mul  s phi sin phi cos mul mul l add translate
  /phi aphi depth get def
                   phi 90 sub rotate phi sin dup scale d+1 pythstem d-1
  grestore
}if 1 add
}def
%%Endprolog
%
%    Program ---the script---
%
/phi 60 def /s 25 def /l 50 def         %globals
/depth 0 def
```

```
/aphi 25 array def
1 setgray 2 setlinewidth 1 setlinecap %oblique tree phi=60
22121943 srand
0 0 moveto s 0 rlineto stroke
15 pythstem pop
showpage
%%EOF
```

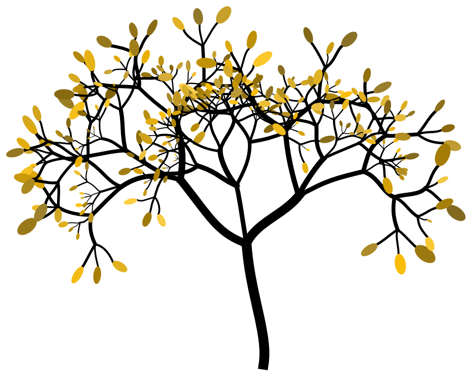
John Deubert's Tree

John begins by discussing what recursion is and how to do this in PS. As (classical) example of recursion he uses the calculation of n-factorial, also given in Adobe's BlueBook p.71. A tail recursion, with infix operators, which can easily be recasted into a loop as shown below at right. The point is that we may encounter recursions in the spirit of FIFO or in the spirit of LIFO. An example of this difference is the determination of the binary digits of a number via LIFO or via FIFO (see my LIFO and FIFO sing the Blues of old).

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Recursive calculation of N!
%%BeginProlog
/factorial{% n ===> n!
dup 1 gt {dup 1 sub factorial mul} if
}def
/Helvetica 12 selectfont
%%EndProlog
0 0 moveto 5 factorial (    ) cvs  show showpage
%%EOF
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Non-Recursive calculation of N!
%%BeginProlog
/factorial{% n ===> n!
1 2 1 4 -1 roll { mul } for
}def
/Helvetica 12 selectfont
%%EndProlog
 0 0 moveto 5 factorial (    ) cvs  show showpage
%%EOF
```

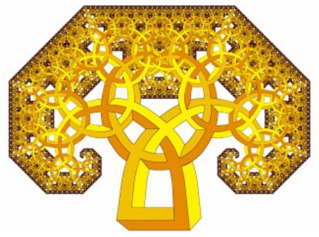
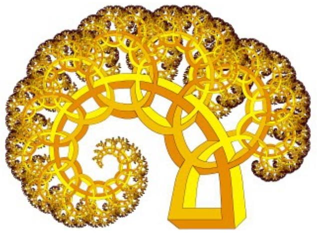
Next he discusses what might limit recursion: the sizes of the various stacks are finite, especially the graphics state stack — pushed by gsave and popped up by grestore — might cause problems by too deep recursion. Adobe implemented, since LanguageLevel 2, flexible stacks, where the (stack) limits are increased automatically, when the need arises. In Acrobat Pro as interpreter I expect no stack limit problems. I have modified John's Branch procedure by maintaining the recursion order on the stack and adjusted the (repeated) scaling to $1/\sqrt{2}$, set Maxdepth = 6, and selected yellow-brown.



Moreover, I stress that drawing a line in *transformed User Space*, is the essence what has to be done at each invoke of branch. Document structuring conventions have been included and the illustrations are cropped to the BoundingBox, when processed by Acrobat Pro. For the source of John's Tree, FractalTree4.ps (more than a page, mainly because his leaves have size and are worked out in detail) consult Acumen Journal 2003.

Art Trees from the WWW

Interesting and beautiful art variations are given on http://mathpaint.blogspot.com/2007/03/pythagoras-tree.html of which I borrowed the following. At right a sculpture by Koos Verhoeff(2007).

2.5D Pythagoras Trees

Lauwerier(1990) mentions the work of Masaki Aono and Tosiyasu Kunii of 1984. They simulated the trees Aucuba japonica and Gingko biloba. According to Lauwerier this comes down to a 2.5D bare Pythagoras Tree, i.e. prescribed in 3D and projected on 2D. Lauwerier(1990) contains `PYT3DBT` where a 3D Pythagoras Tree has been programmed via backtracking according to the Japanese rules of growth, complete in 3D with projection. A hard to read program.

It is easier for me to write anew recursive programs than to transcribe Lauwerier's backtracking programs. Moreover, limitations of BASIC are not inherited and more readable program without goto's will emerge.

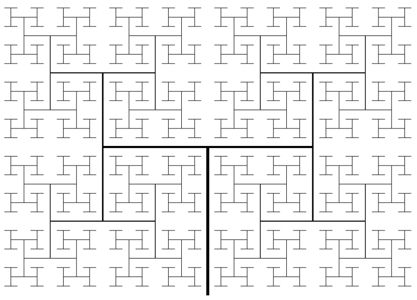
The growth rules are: from the stem branches have angle σ_k and each branch has 2 subbranches with angles α_{k_i}, β_{k_i} which lie in a plane perpendicular to the plane formed by the stem and the branch. Reduction factors are parameters.

For a 2.5D Tree I could modify the Line Tree program given earlier according to the growth rules and incorporate 3D data and projection with viewing angles as parameters. Maybe CAD software is to be preferred?

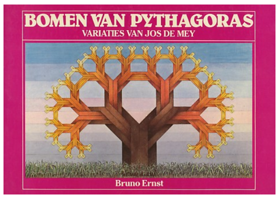
Annotated References

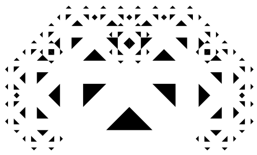
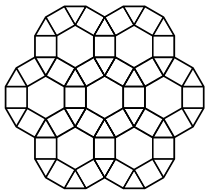
- An introductory survey: `http://en.wikipedia.org/wiki/Pythagoras_tree`.
- Adobe Red, Green and Blue Books. The musts for PS programmers. The simplified tree is a variant of `FractArrow` as given in the BLue Book p.74, which also embodies the H-fractal. One only has to vary the rotation angle, as can be witnessed from the enclosed example. Lauwerier(1987) gives `BOOMH1`, `BOOMH2`, `BOOM2` which I transcribed, but did not include because of my general binary tree and H-fractal backtracking recursive program in PS. Of `BOOM3`, trinary tree, the transcripted program and illustration have been given earlier.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: H-fractal, 2011
%%Author: Kees van der Laan, kisa1@xs4all.nl
%%BoundingBox: -75 -2 75 100
%%BeginSetup
%%EndSetup
%%BeginProlog
/line{0 0 moveto 0 s lineto stroke}def
/Hfractal{line 0 s translate      %draw the line
1 sub  dup 0 gt
 {gsave 90 rotate .7071 dup scale %transformed user space
   Hfractal                       %play it again, Sam
  grestore
      -90 rotate .7071 dup scale  %transformed user space
   Hfractal                       %play it again, Sam
  }if 1 add } def
%%EndProlog
/s 50 def 10 Hfractal pop showpage
%%EOF
```



- Biography of H.A. Lauwerier: http://bwnw.cwi-incubator.nl/cgi-bin/uncgi/alf.
- Deubert, J(2003, June): Acumen Journal.
  `http://www.planetpdf.com/planetpdf/pdfs/AcumenJournal_June2003.pdf`
  (Highly educative. This note is inspired by John Deubert's variations, although Lauwerier already touched upon them. I reconstructed his suggestions from scratch.)
- Ernst, B(1985): Bomen van Pythagoras. Met illustraties van Jos de Mey. Aramith. (Full of variations of Pythagoras Trees, such as the neomondriaan, second below, where the spurious triangles have been blackened.)



- Gleisk, J(1987): CHAOS — making a new science. Penguin.
  (An introduction to and survey of the world of nonlinearity, strange attractors and fractals.)
- Goossens, M(2007, sec ed) et. al.: LaTeX Graphics Companion. ISBN 978 0 321 50892 8.
- Helmstedt, J(2011): A New Method of Constructing Fractals and Other Graphics. The Mathematica Journal. (Nice examples of Lindenmayer systems, for which Lauwerier's KRONKEL can be used.)
- Jackowski, B, P. Strelczyk, P. Pianowski(1995-2008): PSView5.12. WWW. `bop@bop.com.pl`. (Extremely fast previewer for `.eps` among others, which allows `PSlib`(rary) inclusion via the `run` command).
- Knuth, D.E, T. Larrabee, P.M. Roberts(1989): Mathematical Writing. MAA notes 14. The Mathematical Association of America.
- Knuth, D.E(1990, 10th printing): The TeXbook. Addison-Wesley. ISBN 0-201-13447-0. (A must for plain TeXies.)
- Lauwerier, H.A(1987): FRACTALS — meetkundige figuren in eindeloze herhaling. Aramith. (Contains programs in BASIC. Lauwerier H.A (1991): Fractals: Endlessly Repeated Geometrical Figures, Translated by Sophia Gill-Hoffstadt, Princeton University Press, Princeton NJ1. ISBN 0-691-08551-X, cloth. ISBN 0-691-02445-6 paperback. "This book has been written for a wide audience ... " Includes sample BASIC programs in an appendix.)
- Lauwerier, H.A(1988): The Pythagoras Tree as Julia Set. CWI-Newsletter.
- Lauwerier, H.A(1989): Oneindigheid — een onbereikbaar ideaal. Aramith. ISBN 90 6834 055 7. (Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1990): Een wereld van FRACTALS. Aramith. ISBN 90 6834 076 X. (Contains a.o. PYT3DBT a BASIC backtracking program for 3D bare Pythagoras Trees.)
- Lauwerier, H.A(1994): Spelen met Graphics and Fractals. Academic Service. ISBN 90 395 0092 4. (An inspiring book with Math at the high school level for a wide audience; the BASIC programs I consider outdated for direct use.)
- Manning J.R(1972): Continuity conditions for spline curves. Computer Journal, 17,2, p181-186.
- Peitgen, H.O, H.Jürgens, D. Saupe(2004 sec.ed.): Chaos and Fractals. New frontiers of Science. (Images of the fourteen chapters of this book cover the central ideas and concepts of chaos and fractals as well as many related topics including: the Mandelbrot set, Julia sets, cellular automata, L-systems, percolation and strange attractors. This new edition has been thoroughly revised throughout.

The appendices of the original edition were taken out since more recent publications cover this material in more depth. Instead of the focused computer programs in BASIC, the authors provide 10 interactive JAVA-applets for this second edition via `http://www.cevis.uni-bremen.de/fractals`. An encyclopedic work. Audience: Accessible without mathematical sophistication and portrays the new fields: Chaos and fractals, in an authentic manner.)
- Swanson, E(1986, revised ed): Mathematics into Type. American Mathematical Society.
- Szabó, P(2009): PDF output size of TEX documents. Proceedings EuroTEX2009/ConTEXt, p57–74. (Various tools have been compared for the purpose.)
- Van der Laan, C.G(1992): LIFO and FIFO sing the Blues. MAPS 92.2.
- Van der Laan, C.G(1995): Publishing with TEX. Public Domain. (See TEX archives.)
- Van der Laan, C.G(1997): Tiling in PostScript and MetaFont — Escher's wink. MAPS 97.2.
- Van der Laan, C.G(unpublished, BachoTEX workshop): TEXing Paradigms. (A plea is made for standardized macro writing in TEX to enhance readability and correctness.)
- Van der Laan, C.G(2009): TEX Education: an overlooked approach. EuroTEX2009-3rdConTEXt proceedings. (Launched my `PSlib.eps` library.)
- Van der Laan, C.G(2011): Gabo's Torsion. MAPS 42. (Contains a summary of the PS language and its developments.)
- Van der Laan, C.G(submitted MAPS): Julia fractals in PostScript.
- Veith, U(2009): Experiences typesetting mathematical physics. Proceedings EuroTEX2009/ConTEXt, p31–43. (Practical examples where we need to adjust TEX's automatic typesetting.)
- Links to Pythagoras Trees in Art
  http://www.arsetmathesis.nl (A site about Art&Math, such as Pythagoras Trees, Escher, ...)
  `http://mathpaint.blogspot.com/2007/03/pythagoras-tree.html`
  `http://flickr.com/search/?w=32286042@N00&q=Pythagoras&m=text.`

Conclusions

The Pythagoras Tree family can be programmed elegantly in PS with its EPSF, with its feature to transform user space, and because after processing with Acrobat Pro the pictures are delivered in `.pdf` format and cropped to the prescribed BB, ready for inclusion in publications.

PS programs can be written as readable as literature.

While working on this note the Lindenmayer production rule was enriched by PS concepts!

The paradigm is to draw a square (line, spline) scaled and rotated, at the right place, repetitively.

My variant in Metafont with its path datastructure looks more efficient.

I have spotted a few BASIC Pythagoras Trees in Lauwerier(1987) and found John Deubert's versions of the Pythagoras Tree in PostScript on the WWW, where I also stumbled upon beautiful artistic variations on the theme.

It was a surprise that so few and minor adaptations yielded such a rich variety of results.

My placing of illustrations in TEX documents suffers from the same drawback as with Word: changing the source text might disturb the layout.

In TEXworks I finally found out how to keep program texts, especially comments, vertically aligned: I use the font `Terminal` in the input window.

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

Conversion into Word made me hands-on aware of differences between TEX and Word. If you are after utmost accurate user-controlled typeset Mathematics then plain TEX is to be preferred, for bread and butter Mathematics Word can do. The hyphenation by TEX seems better than Word. Inclusion of the `.jpg` figures and `.pdf` objects went smoothly. I did not succeed in handling the subtle spacing of Mathematics in Word. Neglecting superfluous spaces, which TEX does automatically, has been lost in the conversion. I don't know how to switch off, or change, pre-settings, such as: don't underline automatically WWW addresses, maybe by de-activating the option `WWW addresses as hyperlinks`?

Acknowledgments

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

Thank you Nico Temme for mentioning the work of Bruno Ernst, Hans Jansen for reminding me of the correct quote from the Casablanca movie, Jos Willink for proofing, Wim Wilhelm for drawing my attention to the class of scripting languages, MAPS editors for improving my use of English and Taco Hoekwater for procrusting my plain TEX note into MAPS format. Thank you Zinadia Nikolaevna Gulka for inviting me to submit a paper or two for the `Informatsionnie Texnologii i Matematicheskoe Modelirovanie´ journal, and her co-worker for stimulating me to convert the TEX marked up ASCII source of an early version of this note into the required Word. The invitation stimulated me to adapt and revise the material. Thank you GUST for publishing a previous version of this note in the EuroBachoTEX2012 proceedings. Thank you Svetlana Morozova for prompting me in the use of Word.

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

I was trapped by the use of the ∞-symbol in footnotes: explicit font switching has to be done, a nuisance.

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

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

The bad news with respect to `.eps` into `.pdf` conversion is that the newest Acrobat 10 Pro X does not allow for the `run` command for library inclusion.

Afterthoughts: Pondering about my languages and tools

Natural languages serve a lifetime: Dutch, English, Russian, ... My programming languages come and go. I learned programming in Algol60/68 at the time when performing (numerical) calculations was the main use of mainframe computers. FORTRAN was the language in use in the USA. Despite the numerical program library —NUMAL— in Algol60 by the CWI, to which I contributed several procedures, despite the NAG ALGOL60 library, despite the programs published in the handbook series of linear algebra and approximation in Nümerische Mathematik, despite … Algol60 is dead, as dead as a doornail. Algol68 I consider one big side effect, it never catched on, despite its NAG Algol68 library to which I contributed a FFT collection of operators, despite its interface to FORTRAN, which we commanded for at Groningen. FORTRAN is alive with its numerical program libraries (IMSL, NAG, …) with its mathlab, with its published programs in the Collected Algorithms of the ACM, with its published programs in TOMS (Transactions on Mathematical software), with its programs published in various books, with … and despite Dijkstra's famous and shocking `Goto's considered harmful' letter. FORTRAN is efficient, much more efficient than the Algol family ever was, because of Algol's rigorous checking on array bounds for example. FORTRAN is the language of choice on the so-called number crunchers. Over time FORTRAN has evolved, F77, F90, … and extended by facilities needed. PASCAL is alive as a general purpose educational programming language, I presume. Modula was interesting, with its modules concept. ADA elaborated on that but did not find a wide audience, despite the support of the EU. PL1 was an impressive, huge effort, while at the same time with UNIX&C a tendency towards simplicity, smaller scale and cooperating tools had started. BASIC (Beginners All Purpose Instruction Code, 1964!) is apparently alive (Wikipedia), and embraced by Lauwerier. The simplicity of BASIC is a plus. For graphics PS is more suited because the results can be easily included in publications typeset by pdfTEX, Word, … . Moreover, PS is portable in place and time, and maintained by Adobe. ConTEXt can include even more formats on the fly. I prefer to tune my illustrations separately. For interactive work Java is more appropriate, I guess. C, C++ (modular, object oriented) are heavily used programming languages. A newer approach forms the group of scripting languages such as AWK, Lex, Yacc, Perl, Python, …. Little experience with that group myself: read Larry Wall's interesting, amusing, beautiful Perl book, TIMTOWTDI, and played a little with Perl.

TEX&MetaFont/-Post serve a need in document production, although Metafont is outdated and overruled by the worldwide Open Type Fonts activity and TEX is becoming of age despite pdf(La)TEX, where the result is not a `.dvi` but a `.pdf` file.

The TEX collection DVD with

- pdf(La)TEX and its wealth of packages
- ConTEXt integrated with MetaPost, keeps TEX&MetaPost alive, and more
- its TEX&Co archive
- the TEXworks IDE, with scripting possibilities, dictionaries
- …

is the main reason that pdf(La)TEX, and descendants, are still being used mainly within the subculture of TEX User Groups. The discussion lists of UGs serve a need. LuaTEX is a promising new development as successor of TEX, though old Ben-LeeUser-TEXies have created their subset of use of TEX&Co and stick to it, fading away from the UG's.
MS Word is, I estimate, at least a million times more in use. For TEX&Co's survival the possibility to use Open Type fonts is a necessary condition —which activity is underway— next to the volunteer-biased production and distribution of the TEXLive DVD with the tools and languages.

Would I advise new users TEX&Co?

20 years ago I would.

The investment in learning TEX&Co will cost a novice still much time compared to the investment in learning to use other more intuitive interactive tools, such as MS Word, with its Cambria formula templates and OT (Math) fonts, despite its lack of automatic formula numbering, its … for the moment. For those already familiar with TEX&Co it is important that it 'll continue to cooperate smoothly with developments since the birth of TEX, such as `.pdf` and OT scalable fonts. LaTEX users don't have to invest much: they just use canned packages. After 2^5 years of TEX&Metafont

the TeX kernel can be considered time-proven bug free, after some adaptations casu quo corrections. The TeX world contributes with LaTeX packages, with TeXworks, an AllTeX IDE (integrated development environment), with newer versions of ConTeXt, and research towards the successors of TeX, such as LaTeX2ε, NTS, XeTeX, LuaTeX, ... For the graphics don't forget PS with its EPSF — the AllTeX time-proven, graphics companion — maintained (and adapted to developments) by Adobe, or if you can afford it the omnipotent Mathematica. I consider the biggest weakness of the preprocessor MetaPost that it shields you away from PS, it does not really interface with PS. Processing requires an extra step. The syntactic sugar, which MetaPost provides, is not necessary, despite its ability to solve linear equations on the fly. However, ... Hobby's `boxes` macros are nice and beautiful. The interfacing of TeX with MetaPost is nice, for inclusion of illustrations on the fly. Interfacing TeX with Metafont had its virtues (as was provided for in 4AllTeX, which is no longer maintained nor distributed) has become superfluous with the rise of the universal, scalable OT fonts. (A nice feature was to process a selected part of a document, separately on the fly.)

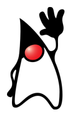
Adobe InDesign CS5.5 software lets you design and preflight engaging page layouts for print or digital distribution with built-in creative tools and precise control over typography. Integrate interactivity, video, and audio for playback on tablets, smart-phones, and computers. No hands-on experience. Impressive. HMTL (with its style sheets) and XML are the (markup) languages for the WWW, enriched with scripting facilities for interactivity and forms.

EPSF I use for my illustrations to be included in a plain TeX document. I call EPSF a TeXies graphics companion. Outside the TeXworld PS is the de-facto standard industrial page description cq printer language.

Photoshop is a tool, not a language, without rigorous BNF syntax notation. It is very alive, it is bundled in the so-called variants of the Creative Suite to facilitate exchange of files between other Adobe tools of the CS. I use Photoshop for editing photographs (illustrations) and for conversion of `.jpg` into `.eps`.

Acrobat is another useful tool with its `.pdf` format. It was developed to compensate for PS deficiencies in porting documents, especially the glyphs of the used fonts were not correctly rendered everywhere. I use `Acrobat Pro` as interpreter/viewer for my EPSF. `Acrobat Reader`, the `.pdf` viewer, is free to use. On Adobe's Acrobat activity MS reacted with XPS.

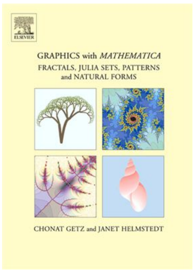
LINUX, a free to use UNIX-biased operating system, is a treasure, and comes with C and many other programs and tools. It is GNU-licensed, free to use software, enriched by the community at large, with the copyright and intellectual ownership protected.



Java Duke

Java, and the Cabri software, a must for (math) teachers, with its animation and interactive functionalities.
`http://en.wikipedia.org/wiki/Java_(programming_language)`
`http://download.oracle.com/javaee/6/tutorial/doc/bnadx.html`

Mathematica is rumored to be a very useful production and experimentation tool, for people like me.
It is too expensive for me privately, for the time being, but ... Mathematica notebooks can be read thanks to the free reader.


GRAPHICS with MATHEMATICA
FRACTALS, JULIA SETS, PATTERNS and NATURAL FORMS
CHONAT GETZ and JANET HELMSTEDT

On my Vista PC I have parallel Ubuntu LINUX, but hardly use it. The newest MEDION computers come with a fast boot option, meaning LINUX as dual and fast OS next to Windows System 7. Macintosh OS, X and beyond, has been built upon LINUX as documented open source kernel.

Quite something!

“Making the right choices has become more than ever important, because … you simply can’t familiarize with all.”

A question one must ask oneself over and over again Does it still make sense or is it a waste of time and energy, to create illustrations in PS to be included in AllTEX marked up documents, to maintain a library of PS `defs` etc, in the presence of the omnipotent commercial graphics- and Math-oriented tool Mathematica, with its ubiquitous free for use notebooks, in the presence of animated Java, in the presence of special fractal tools, such as for example the XaoS or Fractulus programs, with rich colouring and zooming functionalities?

Lauwerier(1994) adhered to PowerBASIC (compiler) because of the concise and fast programs and because of interactivity (he even has provided a zoom program in BASIC!?!). BASIC dates back to 1964(!). At the moment there is PowerBASIC version 10, which reflects that BASIC is maintained. In 2006 MS came with Visual BASIC. BASIC4GL is a free download, but … not upwards compatible; moreover, I can’t run Lauwerier’s programs as such, and … I don’t know for the moment how to reuse BASIC4GL illustrations in my publications.

Notes

1. Lauwerier(1989) narrates what mathematicians thought about the ∞-concept through the ages from the ancient Greeks onward.
2. The run command does include my library when processed by my version of PSView. PSView is very fast. BASIC is interactive and PS batch-oriented.
3. This is different from a polynomial $P_3(x)$, where 2 values and 2 derivatives determine the polynomial uniquely.

My case rests, have fun and all the best.
Chisinau, 14 febr 2012

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

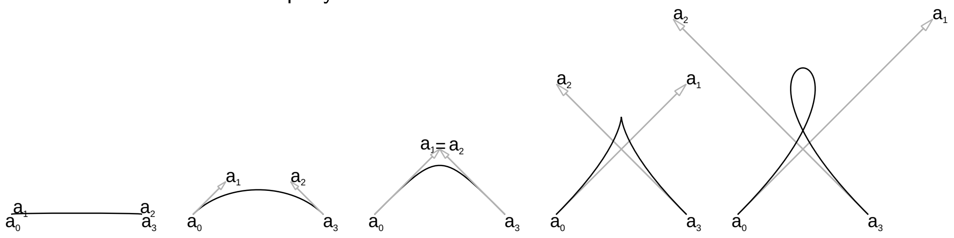
Appendix 0 Splines

In PS a spline, a Bézier cubic — a vector function of the time variable — is characterized by the begin point a_0, the control points a_1 and a_2, also called handles, and the end point a_3.

"Splines are the important 20th century's time-dependent functions comparable to the 19th century's Fourier series for approximations."

In Java one can drag the handles and watch the effects, interactively. Nice.

The control point a1 lies on the tangent to the spline in a0, and the control point a2 lies on the tangent to the spline in a3. The points `a0` and a3 and the angles of the tangents to the spline at these points are not enough to describe the spline uniquely: the size of the handles also matters, as explained in Manning(1972).[3] The control points stand for the angle of the tangents and the size of the handles and therefore determine the B-cubic uniquely.



With `a0` as `currentpoint` a B-cubic, characterized by `a0 , a1, a2 , a3`, is appended to PS's (internal) `path` by `a1 a2 a3 curveto`.

Mathematical formula of a spline Splines as used in PS, and in MetaPost, are Bézier cubics. These 3rd degree polynomials are a linear combination of 3rd degree Bernsten basis polynomials, which were discovered in 1912 by Bernsten.

A n^{th} degree Bernsten basis polynomial is $B_{\nu n}(t) = \binom{n}{\nu} t^\nu (1-t)^{n-\nu}$, $\nu = 0, 1, ..., n$.

A Bézier cubic — a linear combination of 3rd degree Bernsten basis polynomials — with begin point a_0, control points a_1, a_2, and end point a_3 reads

$$z(t) = (1-t)^3 a_0 + 3(1-t)^2 t\, a_1 + 3(1-t)t^2\, a_2 + t^3 a_3 \quad \text{with} \quad z, a_0, a_1, a_2, a_3 \in R^2,\ t \in [0,1].$$

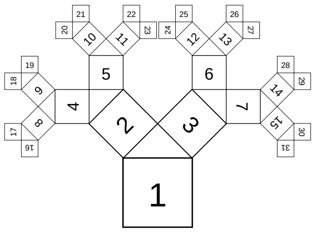
A PS spline path is created by `curveto` and painted to the current page by `stroke`. For the evaluation the de Casteljau algorithm is generally used

$$z(t) = (1-t)\Bigg((1-t)\Big((1-t)a_0 + t\, a_1\Big) + t\,\Big((1-t)a_1 + t\, a_2\Big)\Bigg) + t\Bigg((1-t)\Big((1-t)a_1 + t\, a_2\Big) + t\,\Big((1-t)a_2 + t\, a_3\Big)\Bigg).$$

For this note the above is enough to know about splines, because of the splines I only randomly varied the position of the control points.

Appendix 1: Lauwerier's BASIC versions

Lauwerier uses the binary number representation to identify the various nodes in a binary tree. A node with decimal number n has left descendant $2n$ and right descendant $2n+1$. For example node $5 = 101_2$, has left descendant node $10 = 1010_2$, and right descendant $11 = 1011_2$. If we identify 0 in the binary representation with a Left transformation and 1 in the binary representation with a Right transformation then the node $(1)01_2 = 5$ is arrived at by the transformation LR, where the first bit is neglected.



PYTHB1 is Lauwerier's computational intensive variant for the symmetric Pythagoras Tree: each node is calculated beginning from the root each time with a square drawn at the node layer for layer, or order after order.

```
10 REM ***Pythagoras Tree, program PYTHB1***
20 REM ***Computation intensive***
30 PI=3.141593
40 P=8 : DIM A(P) : REM ***KEUZE ORDER***
50 X=0 : Y=0 : U=1 : V=1 : C=1/sqr(2)
60 FOR M=0 TO P
70 FOR N=2^M to 2^(M+1)-1
80 L=N : H=1 : X=0 : Y=0 : F=0
90 FOR K=0 TO M-1
100 A(M-K)=L MOD 2 : L=INT(L/2) : NEXT K
110 X=0 : Y=0
120 FOR J=1 TO M
130 IF A(J)=0 THEN GOSUB 220 ELSE GOSUB 250
140 NEXT J
150 U=H*(cos(F)+sin(F))
160 V=H*(cos(F)-sin(F))
170 GOSUB 200
180 NEXT N : NEXT M : END
190 LINE  (X-V,Y-U)-(X+U,Y-V) : LINE -(X+V,Y+U)
200 LINE -(X-U,Y+V)           : LINE -(X-V,Y-U) : RETURN
210 X=X-H*(cos(F)+2*SIN(F))
220 Y=Y+H*(2*cos(F)-SIN(F))
230 F=F+PI/4 : H C*H : RETURN
240 X=X+H*(  cos(F)-2*SIN(F))
250 Y=Y+H*(2*cos(F)+  SIN(F))
260 F=F-PI/4 : H C*H : RETURN
270 END
```

Notes

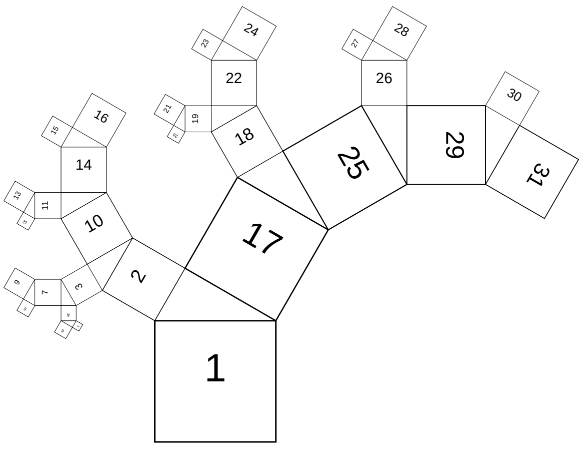
In the transcription use has been made of

- PS user Space
- **Left**, **Right** transformations
- **drawsquare** procedure
- in PS **mod** and **idiv** require integers
- the array has to be enlarged because it starts by 0

```
%!PS-Adobe-3.0 EPSF-3.0
%%Transcriptor: Kees van der Laan, April 2011, kisa1@xs4all.nl
%%BoundingBox: -300 -55 300 350 %6s X 4s
%%BeginSetup
%%EndSetup
%%BeginProlog
/drawsquare{-.5 s mul dup s s rectstroke}def
/Left {0 .5 s mul translate  45 rotate c c scale
0 s translate}def
/Right{0 .5 s mul translate -45 rotate c c scale
0 s translate}def
/PYTHB1{%Pythagoras Tree (computational intensive)
a la Lauwerier
/p exch def /A p 1 add array def
0 1 p{/m exch def
 2 m exp  1  2 m 1 add exp 1 sub{%2^m 1 2^(m+1)-1
   /n exch def
   gsave
   0  1  m 1 sub{/k exch def %store bits of n in
A
      A m k sub cvi  n cvi 2 mod  put  /n n cvi
2 idiv def
      }for%binary digits of n  into A
   %transform user space
   1 1 m{A exch get 0 eq {Left}{Right}ifelse}for
   drawsquare
   grestore
   }for%n
 }for%m
} bind def
%%EndProlog
%
%   Program --- the script ---
%
/s 100 def % s = side of initial square
10 PYTHB1  % order = 10
showpage
%%EOF
```

PYTHB3 is Lauwerier's backtracking variant of the (oblique) Pythagoras Tree.
The order of transversal of the Tree is given by the numbers in the squares which illustrates the printing sequence of the squares during the backtracking process. My (recursive, backtracking) variant is given in the beginning of this paper, which has also been used to generate the illustration below.

```
10 REM ***Skewed Pythagoras Tree, program PYTHB3***
20 SCREEN 3 L CLS : PI=3.141593
30 WINDOW (-5,-3)-(5,4.5)
40 P=12 : DIM X1(P),Y1(P),X2(P),Y2(P),U1(P),V1(P),U2(P),V2(P)
50 REM ***CHOICE FOR ANGLE F***
60 F=PI/5 : C=COS(F) : S=SIN(F)
70 A1=-C*S : A2=C^2 : B1=A1+A2 :B2=-A1+A2
80 C1=B2 : C2=1-B1 : D1=1-A1 : D2=1-A2
90 X1=0 : Y1=0 : U1=0 : V1=0
100 LINE (0,0)-(0,1) : LINE -(1,-1) : LINE -(1,0)
110 S=1 : GOSUB 170
120 FOR M=1 TO 2^(P-1)-1 : S=P : N=M
130 IF N MOD 2=0 THEN N=N\2 : S=S-1: GOTO 130
140 GOSUB 150 : NEXT M END
```



```
150 X1(S-1)=X2(S-1) : Y1(S-1)=Y2(S-1)
160 U1(S-1)=U2(S-1) : V1(S-1)=V2(S-1)
170 FOR J=S TO P
180 X=X1(J-1) : Y=Y1(J-1) : U=U1(J-1) : V=V1(J-1)
190 X3=U-X : Y3=V-Y
200 X1(J)=X+A1*X3-A2*Y3
210 Y1(J)=Y+A2*X3+A1*Y3
220 U1(J)=X+B1*X3-B2*Y3
230 V1(J)=Y+B2*X3+B1*Y3
240 X2(J)=X+C1*X3-C2*Y3
250 Y2(J)=Y+C2*X3+C1*Y3
260 U2(J)=X+D1*X3-D2*Y3
270 V2(J)=Y+D2*X3+D1*Y3
280 LINE (X,Y)-(X1(J),Y1(J)) : LINE -(U1(J),V1(J)) : LINE -(U,V)
290 LINE -(X,Y) : LINE -(X2(J),Y2(J)) : LINE -(U2(J),V2(J)) : LINE -(U,V)
300 NEXT J : RETURN : END
```

Appendix 2: My Metafont programs of old

For the MetaFont and MetaPost aficionados I have included my simplified MetaFont Tree, in 2 versions. Those fluent in MetaPost can adapt the codes for their purposes, I presume.

The first program is an example of straightforward recursive programming, as in classical languages with value variables, where the fractal property is used that at each level a complete tree of lower order has to be drawn with the right orientation and at the right scale.

```
%November 1995, CGL. Pythagoras (line) Tree. (Coding approach borrowed from my PWT guide, where
I did it in TeX!)
proofing:=1;screenstrokes; pickup pencircle scaled .005pt;
def pt(expr n,%order
            z,%drawing coordinate pair
            d,%(in-)angle  at z
            l %length of branch to be drawn
)=if n>1:draw z--z+l*dir(d+45); pt(n-1,z+l*dir(d+45),d+45,.7l);
         draw z--z+l*dir(d-45); pt(n-1,z+l*dir(d-45),d-45,.7l);
  fi
enddef;
l=100; draw (2l,0)--(2l,l); pt(8,(2l,l),90,.6l); showit; end
```

The non-recursive variant is impressive to watch when processed by Blue Sky's MetaFont on my old PowerMac of 1997. First the list of left nodes is built after which in the backtracking the left path is rotated and copied to yield the right part. Very fast this swapping.

```
%December 1995, CGL. Nonrecursive Pythagoras (line) Tree
pickup pencircle scaled 1.5;
pair node[];
n=15;                 %order
l=125;                %size of the trunk
node[0]=origin; d= 90;%position and orientation trunk
                      %(so rotating or shifting the tree is easy) %Create nodes of most leftbound
branche
for k=1 upto n: node[k]=node[k-1]+l*dir d; d:=d+45;l:=.7l;endfor; %/sqrt2
def openit=openwindow currentwindow from origin to (screen_rows, screen_cols) at (-250, 375) enddef;
for k=n-1 downto 1: draw node[k+1]--node[k];
    addto currentpicture also currentpicture rotatedaround (node[k],-90);%The swapping!
endfor
draw node1--node0; drawdot origin; showit; end
```

Comment anno 2012: This swapping should be formalized into a production rule; work to do.

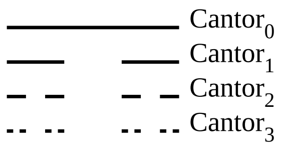
Classical Math Fractals in PostScript

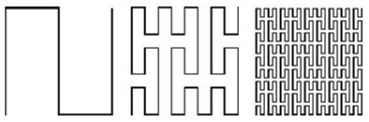
Fractal Geometry I

Abstract
Classical mathematical fractals in BASIC are explained and converted into mean-and-lean EPSF `defs`, of which the `.eps` pictures are delivered in `.pdf` format and cropped to the prescribed `BoundingBox` when processed by `Acrobat Pro`, to be included easily in pdf(La)TEX, Word, ... documents. The EPSF fractals are transcriptions of the Turtle Graphics BASIC codes or programmed anew, recursively, based on the production rules of oriented objects. The Lindenmayer production rules are enriched by PostScript concepts. Experience gained in converting a TEX script into WYSIWYG Word is communicated.

Keywords
Acrobat Pro, Adobe, art, attractor, backtracking, BASIC, Cantor Dust, C curve, dragon curve, EPSF, FIFO, fractal, fractal dimension, fractal geometry, Game of Life, Hilbert curve, IDE (Integrated development Environment), IFS (Iterated Function System), infinity, kronkel (twist), Lauwerier, Lévy, LIFO, Lindenmayer, minimal encapsulated PostScript, minimal plain TeX, Minkowski, Monte Carlo, Photoshop, production rule, PSlib, self-similarity, Sierpiński (island, carpet), Star fractals, TACP, TEXworks, Turtle Graphics, (adaptable) user space, von Koch (island), Word

Contents

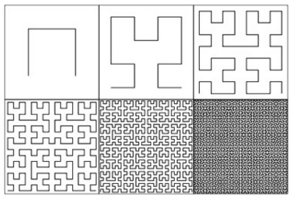



Peano curves: order 1, 2, 3

Introduction

My late professor Hans Lauwerier published nice, inspiring booklets about fractals with programs in BASIC. However, I don't know how to include elegantly the pictures, obtained by running the BASIC codes, in my documents. Moreover, I consider PostScript (PS, for short) more portable in place and time, can include EPSF results in my TEX documents[1] easily, and ... do realize that PS is the de-facto standard industrial printer language.

This note is about conversion of some of Lauwerier's BASIC Turtle Graphics codes for the simplest fractals into EPSF, ànd about the programming of new recursive EPSF `defs` biased by Lindenmayer production rules for oriented objects, enriched with PS concepts.



Now and then I have explained Lauwerier's algorithms, especially when he associates binary and quaternary number representations with fractals.

Fractals have widened the dimension concept into fractal-valued dimensions. Although the fractal dimension concept is not necessary in order to understand the codes, I have added the appendix Fractal Dimension, because fractal dimensions contribute to characterizing fractals. Moreover, fractal dimension gives meaning to the 19th century 'monstruous' plane-filling curves.

Fractals were invented in the 20th century, and became the geometry of this century due to the development of computers, because computers are the tools for viewing and researching fractals.

The ancestor of fractals is the 1D Cantor Dust. 2D predecessors of fractals are the plane-filling curves named after Peano, Hilbert, Sierpiński, ... , which captivated mathematicians in the late 19th and the early 20th century.

Sierpiński curves have found their niche in the solution of the travelling salesman problem.

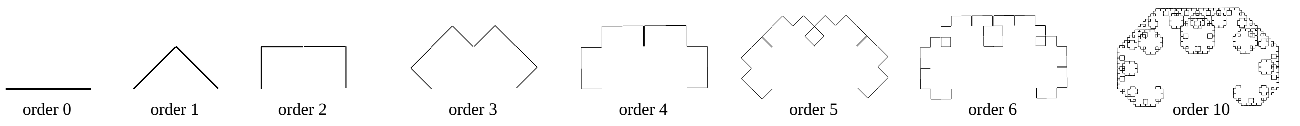
In the sequel Lévy, von Koch, Kronkel (Dutch, means twist), Minkowski, Dragon curve, star fractals, and a variant of the Game of Life are discussed. There are 4 appendices: the first about Fractal Dimension, the second about the historical Cantor Dust, the third about the classical Hilbert curve, and the last about Sierpiński islands.

In the footsteps of Lauwerier, the reader is invited to experiment with the PS programs, of which `defs` are supplied in my `PSlib.eps` library, which I'll send on request. MetaPost aficionados may translate the included Metafont codes into MetaPost, I presume.

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 $2^{-n/2}\times C_0$.

Fractals have various infinite lengths. The question arose: Can these blends of ∞ be used to characterize fractals?[2] Below C_0 ... C_6 and C_{10} have been constructed from the definition.



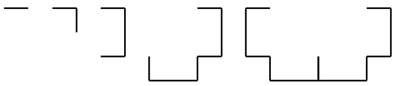
Properties

1a. The above sequence of curves loosely obey

$$C_i = C_{i-1}^{45} \oplus C_{i-1}^{-45}, \quad i = 1, 2, \dots \quad C_0 = \text{ segment}$$

$$\oplus \text{ means spliced} \quad C_{i-1}^{45} \text{ means rotated over } 45°.$$

In fractal terminology such a recursion, or production rule, characterizes what is called the self-similarity of fractals, because C_i consists of 2 spliced copies of C_{i-1}, which are not scalars but 2D, oriented objects. (Positive rotation is counter-clockwise à la PS). Lindenmayer (Dutch theoretical biologist) invented production rules in order to describe plants; production rules are also used in program development.



1b. The C curves at right have a different orientation.
The formula, which reflects the self-similarity in this orientation, reads

$$C_i = C_{i-1} \oplus C_{i-1}^{-90}, \quad i = 1, 2, \dots$$

$$\oplus \text{ means spliced} \quad C_{i-1}^{-90} \text{ means rotated over } -90°.$$

Self-similarity as construction method can be suitably programmed in MetaPost/-font, with their path data structure, as follows. Create the row of paths $p_0, p_1, p_2, \dots$

$$p_0 = C_0, \qquad p_i^{45} \oplus p_i^{-45} \to p_{i+1}, \quad \text{for } i = 0, 1, 2, \dots \quad \text{with } \oplus \text{ the splice operator.}$$

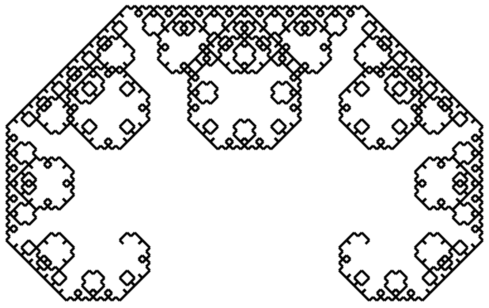
2. In the pen-plotter days the natural question arose: What is the direction of a segment? Lauwerier(1987) gives the intriguing relationship between the angle ϕ_k of a segment and its index k (according to the orientation as given under 1b).

$$\phi_k = (s_k \,\mathbf{mod}\, 4)\frac{\pi}{2} \quad \text{with } s_k = \sum_{j=0}^{p-1} b_j \quad \text{sum of bimals of k}$$
$$\text{and } k = \sum_{j=0}^{p-1} b_j 2^j \text{ binary representation of k.}$$

3. The Lévy fractal has fractal dimension 2, a local plane-filling curve, Lauwerier(1990). The C curves intersects themselves from order 4 onward.

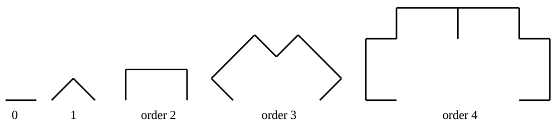
The PostScript program

One might create an efficient recursive backtracking program based on property 1a, as a `levyC def` with the `def` given below. Scaling is commented out; just remove the two `%` signs if scaling is wanted.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Author: Kees van der Laan
%%Date: april 2011
%%Affiliation: kisa1@xs4all.nl
%%BoundingBox: -1 -1 346 61
%%BeginSetup %crops to BoundingBox
%%EndSetup   %by Acrobat Pro
%%BeginProlog%collection of defs
/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 %/s s 1.4142 div def %lower order on stack; s scaled variant
  45 rotate levyC%-45 rotate%combine -45 twice into -90
 -90 rotate levyC  45 rotate
 1 add %/s s 1.4142 div def %adjust order on stack, and s(cale)
}ifelse
}def
%%EndProlog
%
% Program --- the script ---
%
/s 20 def              0 levyC pop
s 2 div   0 translate 1 levyC pop
s         0 translate 2 levyC pop
1.5 s mul 0 translate 3 levyC pop
2.5 s mul 0 translate 4 levyC pop
showpage
%%EOF
```



The above mean-and-lean PS `def` is the result of programming in the spirit of The Art of Computer Programming, TACP for short. I'll come back on a more systematic approach of programming based on production rules, a little further on.

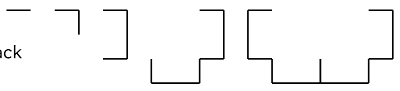
If the `levyC def` is included in the `PSlib.eps` library, then the above `def` can be replaced by

```
(C:\\PSlib\\PSlib.eps) run
```

This feature of the run command is not generally known, so it seems.

Because programming in PostScript is subtle, I have included below the PS `def` based on the production rule as stated under property 1b, which is highly similar to the above backtracking process, but the result differs in orientation. This `levyCvar def` is also included in `PSlib.eps`.

```
/levyCvar{%order on stack ==> C line
          %s = size of segment (global))
dup 0 eq
{0 0 moveto s 0 lineto
   currentpoint stroke translate}%draw line
{1 sub                          %lower order on stack
          levyCvar              %C line
 -90 rotate levyCvar 90 rotate   %rotated C line
  1 add                         %adjust order on stack
}ifelse }def
```



To run the program store the file with extension `.eps` (or `.ps`), right-mouse click the thumbnail of the file and choose the option convert to Adobe PDF in the pop-up menu. That is all when you have installed Acrobat Pro 7. (Other versions of Creative Suite ask for `open in Acrobat`.) I also used Adobe Illustrator and PSView. The latter just by double-clicking the filename upon which the command window opened and a little later PSview.[3]

The Turtle Graphics algorithm is based on property 2. In order to understand the formula mentioned, a table for the direction (with orientation as mentioned under property 1b) of each segment is included. Such a table forms the basis for discovering the regularity.

order \ k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	→																
1	→	↓															
2	→	↓	↓	←													
3	→	↓	↓	←	↓	←	←	↑									
4	→	↓	↓	←	↓	←	←	↑	↓	←	←	↑	←	↑	↑	→	
s_k **mod** 4	0	1	1	2	1	2	2	3	1	2	2	3	2	3	3	0	...

Below I have included Lauwerier's program and my conversion in PS, which is interesting because of the transformation of the user space by ϕ_k, $k = 0, 1, 2, \ldots$.

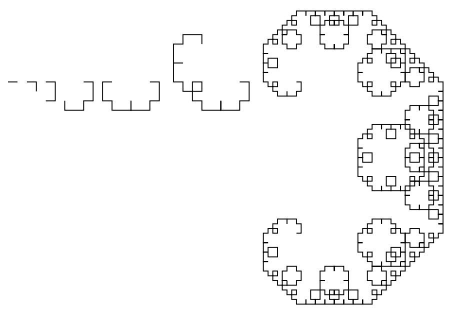
```
10 REM ***Levy fractal***
10 P=12 : REM***order***
20 H=2^(-(P/2)) : A=H*COS(P*PI/4) : B=H*SIN(P*PI/4)
30 LINE (0,0)-(A,-B) : LINE -(A+B,A-B)
40 X=1 : Y=1
50 FOR N=2 TO 2^P-1
60 M=N : S=1
70 IF M MOD 2 = 1 THEN S=S=1
80    M\2
90 IF M>1 THEN GOTO 70
100 IF S MOD 4 = 0 THEN X=X+1
110 IF S MOD 4 = 1 THEN Y=Y+1
120 IF S MOD 4 = 2 THEN X=X-1
130 IF S MOD 4 = 3 THEN Y=Y-1
140 LINE -(A*X+B*Y, A*Y-B*X)
150 NEXT N
160 END
```

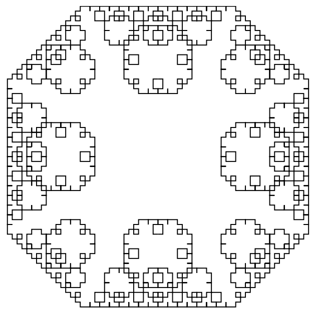
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Levy fractal a la Lauwerier
%%Transcriptor: Kees van der Laan, April 2011, kisa1@xs4all.nl
/LevyLauwerier{/p exch def%order =>C curve (l is global)
0 0 moveto l 0 lineto    %order 0
p 1 eq {0 l neg rlineto} %order 1, segments 0, 1
{0 %previous phi on stack
1 1 2 p exp 1 sub {%for%n, the number of the segments 1,2,...
0 exch %s n :s sum of bimals and n the segment number on stack
   p{dup 2 mod 3 -1 roll add exch 2 idiv}repeat pop%discard n
   4 mod 90 mul dup 3 1 roll sub rotate l 0 rlineto
}for%n
pop%discard phi
}ifelse stroke}def
/l 5 def 10 LevyLauwerier stroke showpage
%%EOF
```

Size of line-piece is wired-in (In PS parametrized)

Subtle stack programming with only 1 rotate for each segment instead of a rotate and a back rotate. The length l of each segment is a global parameter.

Below at left C_0 ...C_5, and C_{10}; at right C_{10} spliced with its `-1 1 scaled` copy, a Lévy carpet.





In my PWT guide of 1995, I did program the above Lévy fractal in TEX (orientation 1b) by the Turtle graphics method, in the footsteps of Knuth. Nowadays, I much prefer the much more powerful and useful PostScript for programming my graphics. Sorry to say so, but Knuth put me on the wrong track by his graphics in the TEXbook.

Lindenmayer enriched by PostScript concepts for the Lévy fractal

What we miss in the 1a property specification is the scaling to smaller size of the segments when the order increases, as well as a more precise meaning of what spliced entails. A more accurate and improved production rule à la 1a, can be obtained when we use PS concepts in the production rule at the expense of simplicity.

$$C_n = [R_{45}S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})}C_{n-1}] \oplus T_{\frac{s}{2}\frac{s}{2}}[R_{-45}S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})}C_{n-1}]$$

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.

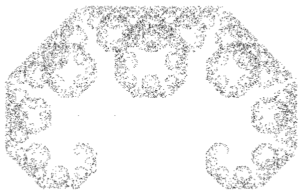
The above PS production rule transcribes systematically into the following PS `def`, which has become more verbose.

```
!PS-Adobe-3.0 EPSF-3.0
%%Author: Kees van der Laan
%%Date: feb 2012
...
/levyC{%on stack: the order => C line
      %s = size of initial segment C_0 (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub
  gsave  45 rotate .7071 dup scale levyC grestore
 .5 s mul dup translate
  gsave -45 rotate .7071 dup scale levyC grestore
 1 add%reset order
}ifelse
}def
```

Systematic programming versus TACP at the expense of verbosity.

Lévy fractal as Iterated Function System

Lauwerier(1994) in one of his exercises created a Lévy fractal by the IFS (Iterated Function Systems) method, which consists of 2 contracted, affine transformations, L and R, (both rotations characterize Lévy) applied with equal chance.



$$\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{L}{=} \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a-1 \\ b \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{R}{=} \begin{pmatrix} c & -d \\ d & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1-c \\ -d \end{pmatrix}, \quad a = .5,\ b = a = c = -d.$$

or after substituting the parameters

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

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

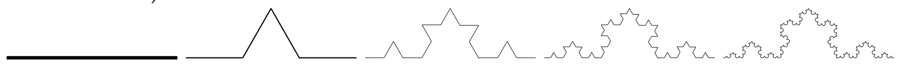
Associated with the Lévy fractal are 2 rotations with rotation centres for L: (-1,0) and for R: (1,0) and contraction $\sqrt{.5} \approx .7$. Amazing, isn't it! Laurier's BASIC program `FRACMC2` and my conversion are given below.

```
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
%%Title: fracmc2
%%Author: H.A. Lauwerier(1994): Spelen met Graphics en Fractals
%%Transcriptor: Kees van der Laan, febr 2012
%%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
```

Helge von Koch

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 $K_0 \ldots K_4$, scaled with increasing order (line thickness is scaled as well).



Properties

1. Each von Koch curve contains 4 copies of the von Koch curve of an order lower, meaning self-similarity, which entails the production rule

$$K_i = K_{i-1} \oplus K_{i-1}^{60} \oplus K_{i-1}^{-60} \oplus K_{i-1},$$

with $K_0 =$ initial segment, $\oplus$ means spliced, K^{60} rotated over 60°.

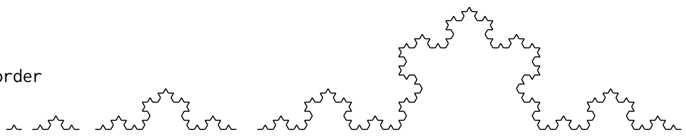
2. The von Koch fractal is a historical example of a curve without a tangent. The curve never intersects itself.
3. The length of the broken line for order n is $(4/3)^n \times K_0$, which with increasing order n goes to ∞. The von Koch curves gave rise to the awareness that the length of the coast of England is infinite. Imagine that the yardstick has length K_0, then all the lines above of the scaled von Koch curves have length 1! So, the length of a fractal depends on the size of your yardstick! Awareness of grades of infinity stirred up the concept of the fractal dimension D, a jolt to the minds of those with an iron cast idea about the 1-2-3-dimensional geometrical world. The fractal dimension is: D = log4 / log3 ≈ 1.26.
4. Lauwerier(1987) mentions the intriguing relationship between the angle ϕ_k of a segment and its index k

$\phi_k = ((s_k + 1)\,\mathbf{mod}\,3 - p)\frac{\pi}{3}$ with $s_k = \sum_{j=0}^{p-1} q_j$ sum of quatermals of k
and $k = \sum_{j=0}^{p-1} q_j 4^j$ quaternary representation of k.

5. The von Koch island remains within the circumscribed circle of the initial triangle (see later).

The PostScript def is an efficient and concise implementation of the above specified rewrite under property 1, neglecting scaling.

```
/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 and do von Koch
   60 rotate vonKoch
 -120 rotate vonKoch
   60 rotate vonKoch
 1 add        %reset order
}ifelse}def
```



Turtle Graphics algorithm is based on the knowledge of each angle ϕ_k. In order to understand, or get a feeling for, the formula mentioned, I have included a small table for the orders 0 and 1. Order 2 yields a too long table, and has been suppressed.

order	p	k	s_k	$((s_k+1)\,\mathbf{mod}\,3) - p$	ϕ_k
0	1	0	0	0	0
1	1	0	0	0	0
		1	1	1	$\pi/3$
		2	2	-1	$-\pi/3$
		3	3	0	0

Lauwerier coded a BASIC program based on the knowledge of the direction of each segment via the Turtle Graphics method. Below I have included Lauwerier's tiny program next to my conversion in PS.

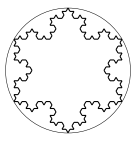
```
10 REM ***Fractal of Helge von Koch***
10 P=4 : DIM T(P) : PI= 3.141593 : REM***order***
20 H=3^(-P) : PSET (0,0)
30 FOR N=0 TO 4^P-1
40 M=N : FOR L=0 TO P-1
50       T(L)=M MOD 4 : M=M\4 : NEXT L
60 S=0 : FOR K=0 TO P-1
70       S=S+(T(K)+1) MOD 3 - 1: NEXT K
80 X=X+H*COS(S*PI/3)
90 Y=Y+H*SIN(S*PI/3)
100 LINE -(X, Y)
110 NEXT N
120 END
```

Length of line-piece wired-in.

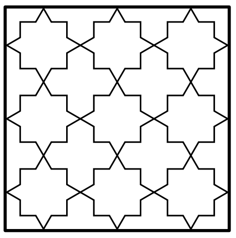
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: von Koch fractal a la Lauwerier
%%Transcriptor: Kees van der Laan, April 2011, kisa1@xs4all.nl
/vonKochLauwerier{/p exch def %order => von Koch fractal (l is global)
/t p array def /h l 3 p exp div def 0 0 moveto
0 1 4 p exp 1 sub{%for%n
/m exch def
0 1 p 1 sub{t exch m cvi 4 mod put
            /m m cvi 4 idiv def
           }for
/s 0 def
0 1 p 1 sub{t exch get 1 add cvi 3 mod s add /s exch def}for
 /s s p sub def
 60 s mul cos h mul 60 s mul sin h mul  rlineto
}for }def
%
/l 100 def 4 vonKochLauwerier stroke
showpage
%%EOF
```

Note that in PS we have to convert the subscript expression for the index of an array explicitly into integer. Another difference is that the arguments of the trigonometric functions are in degrees in PS and in radians in BASIC.

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
```



Lindemayer system enriched with PostScript concepts for the von Koch fractal

What we miss in the program is the scaling to smaller size of the segments when the order increases, as well as a more precise meaning of what spliced entails. A more precise production rule enriched with PS concepts reads

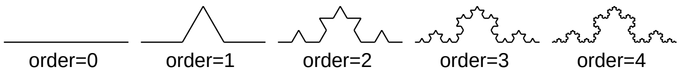
$$\begin{aligned}K_n = [S_{\frac{1}{3}\frac{1}{3}}K_{n-1}] \oplus T_{\frac{s}{3}0}\,[S_{\frac{1}{3}\frac{1}{3}}R_{60}K_{n-1}] \oplus T_{\frac{s}{6}\frac{s\sqrt{3}}{6}}\\ [S_{\frac{1}{3}\frac{1}{3}}R_{-60}K_{n-1}] \oplus T_{\frac{s}{6}\frac{-s\sqrt{3}}{6}}\,[S_{\frac{1}{3}\frac{1}{3}}K_{n-1}]\end{aligned}$$

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.

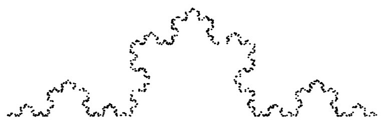
The above PS production rule transcribes systematically into the following PS `def`.

```
!PS-Adobe-3.0 EPSF-3.0
%%Author: Kees van der Laan
%%Date: feb 2012
...
/vonKoch{%on stack: the order => von Koch curve
         %s = size of initial line segment C_0 (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub %adjust order on the stack
  gsave           .3333 dup scale vonKoch grestore
  .3333 s mul          0 translate
  gsave  60 rotate .3333 dup scale vonKoch grestore
  .1666 s mul .285 s mul translate
  gsave -60 rotate .3333 dup scale vonKoch grestore
  .1666 s mul -.285 s mul translate
  gsave           .3333 dup scale vonKoch grestore
 1 add %reset order on the stack
}ifelse
}def
```



Von Koch-like fractal as Iterated Function System

Lauwerier(1994) in one of his exercises created a von Koch-like fractal by (linear) IFS (Iterated Function Systems), which consists of 2 contracted, affine transformations, L and R, which both for the von Koch fractal do contraction and mirroring. For the picture at right I just took 1000 points in order to expose the dot structure, in contrast with the line structure of the earlier approximations of the fractal.



M.F. Barnsley(1988) Fractals Everywhere, exploited contracted IFS.[4]

Most important property: with each contracted IFS is associated a limit figure, the fractal attractor.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{L}{=} \begin{pmatrix} a & b \\ b & -a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a-1 \\ b \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{R}{=} \begin{pmatrix} c & d \\ d & -c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1-c \\ -d \end{pmatrix}, \ a = .5,\ b = .289,\ c = a,\ d = -b.$$

or, after substitution of the parameters

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

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

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, 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
```

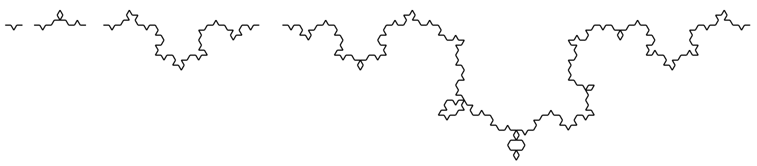
```
Other values of the parameters
a=.5 b=.5   c=.6667 d=0     %bebladerde tak
a=.5 b=.289 c=.5    d=-.289 %von Koch
a=.5 b=.5   c=.5    d=0     %kale tak
a=.5 b=.5   c=.6    d=-.2
a=0  b=.64  c=0     d=-.64  %tegelpatroon
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: fracmc4
%%Author: H.A. Lauwerier(1994): Spelen met Graphics en Fractals
%%Transcriptor: Kees van der Laan, febr 2012
%%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
```

Deterministic von Koch and randomness

Peitgen c.s.(2004) mentions the deterministic von Koch fractal combined with randomness, and states that a better model for coastlines is obtained.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: von Koch Random fractal, 2012
%%Author: Kees van der Laan, kisa1@xs4all.nl
%%BoundingBox: -5 -125 650 15
%%BeginSetup
%%EndSetup
%%BeginProlog
/vonKoch{dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub       vonKoch
 pm{  60 rotate vonKoch
    -120 rotate vonKoch
      60 rotate vonKoch}
   { -60 rotate vonKoch
     120 rotate vonKoch
     -60 rotate vonKoch}ifelse
1 add
}ifelse
}def
22121943 srand
/pm{rand 1073741823 gt{true}{false}ifelse}def
%%EndProlog
%
%Program ---the script---
/s 5 def        % s = initial size of line piece
                    1 vonKoch  pop % order 1
2 s mul 0 translate 2 vonKoch  pop
3 s mul 0 translate 3 vonKoch  pop
showpage
```



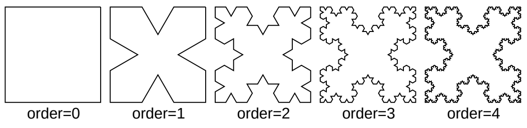
KRONKEL is Lauwerier's universal program to construct fractal islands based on similarity transformations.

```
10 REM ***KRONKEL: Fractal polygonal Island and
model***
10 DIM x(4096), Y(4096)
20 U=4 : DIM A(U), B(U) : REM ***Number of sides of Island***
30 V=4 : DIM C(V), D(V) : REM ***Number of pieces of model***
40 DATA 1,1,-1,1,-1,1,-1,-1,1,-1,1,1 : REM ***Corners island***
50 DATA .3333,0,.5,.2887,.667,0      : REM ***Data model***
60 INPUT P : REM ***Choice order***
70 FOR I=0 TO U   : READ A(I), B(I) : NEXT I
80 FOR I=1 TO V-1 : READ C(I), D(I) : NEXT I
90 REM ***Calculation coordinates Kronkel line***
100 C(0)=0 : D(0)=0 : x(0)=0 : Y(0)=0 : X(v^P)=1 : Y(V^P)=0
110 FOR I=0 TO P-1
120    FOR J=0 TO V^P-1 STEP V^(P-I)
130       M1=J+V^(P-I) : DX=x(M1)-X(J) : DY=Y(M1)-Y(J)
140       FOR K=1 TO V-1
150           M2=J+K*V^(P-I-1)
160           x(M2)=DX*C(K)-DY*D(K)+X(J)
170           Y(M2)=DY*C(K)+DX*D(K)+Y(J)
180       NEXT K
190    NEXT J
200 NEXT I
210 REM ***DRAW ISLAND***
220 PSET(A(0),B(0))
230 FOR M=0 TO U-1
240     DA=A(M+1)-A(M)
250     DB=B(M+1)-B(M)
260     FOR N=0 TO V^P
270         LINE -(DA*X(N)-DB*Y(N)+A(M), DB*X(N)+DA*Y(N)+B(M))
280     NEXT N
290 NEXT M
300 END
```

Length of size of line-piece is wired-in.

In PS
`s` scaling
`u a b` size and corners of island
`v c d` size and corners of model line
have been used as globals, and could have been initialized within the dictionary
No fixed bounds on `x` and `y`

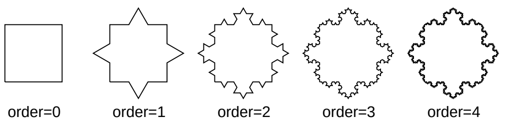
```
%!PS-Adobe-3.0 EPSF-3.0
%%Titel: Kronkel ---Koch Island--- H.A. Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl, April 2011
/kronkel%order p ==> fractal island (globals s, u, a, b, v , c, d)
{kronkeldict begin%push kronkeldict on the d-stack
 /p exch def
/x u v p exp mul cvi array def /y u v p exp mul cvi array
def%auxiliaries
%calculate coordinates corners `kronkel'
 x 0 0  put            y 0 0 put
 x v p exp cvi 1   put y v p exp cvi 0 put
 0 1 p 1 sub{/i exch def %for i
  0 v p i sub exp  v p exp 1 sub{/j exch def %for j
   /m1 j v p i sub exp add def
   /dx x m1 cvi get x j cvi get sub def
   /dy y m1 cvi get y j cvi get sub def
   1 1 v 1 sub{/k exch def %for k
    /m2 j k v p i sub 1 sub exp mul add def
x m2 cvi dx c k get mul dy d k get mul sub x j cvi get add puty m2
cvi dy c k get mul dx d k
get mul add y j cvi get add put }for%k
  }for%j
 }for%i
%create path for each side m of base line
 a 0 get b 0 get moveto
 0 1 u 1 sub{/m exch def%for m
  /da a m 1 add cvi get a m cvi get sub def
  /db b m 1 add cvi get b m cvi get sub def
  0 1 v p exp{/n exch def%for n
   da x n cvi get mul db y n cvi get mul sub a m cvi get add
db x n cvi get mul da y n cvi get mul add b m cvi get add lineto
  }for%n
 }for%m
end}def %end pops kronkeldict off the d-stack
/kronkeldict 8 dict def
%
/s {50 mul} def %scale
/u 4 def %number of corners of the island
/a [ 1 s -1 s -1 s 1 s 1 s ] def %x coordinates of corner points
/b [ 1 s 1 s -1 s -1 s 1 s ] def %y coordinates of corner points
/v 4 def %number of line pieces of the model line
  /c [ 0 .3333  .5    .6667  ] def%x coordinates of corners
  /d [ 0 0      .2887 0      ] def%y coordinates of corners
2 kronkel stroke showpage
%%EOF
```



inward islands

His islands are based on similarity transformations, not on the calculation of the direction of the next line piece as in the line fractals. The `Kronkel` program can also be used for degenerated islands, i.e. for line fractals, such as Lévy, von Koch, Minkowski, ...

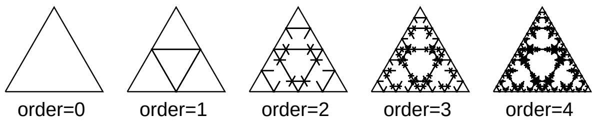
The order of specifying the corners of the island determines whether the fractal is drawn inside (anti-clockwise specification) or outside (clockwise specification)



outward islands

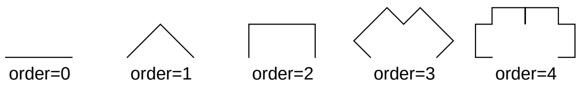
```
...
(C:\\PSlib\\PSlib.eps) run
%globals s, u, a, b, v, c, d
/s {30 mul} def %scale
/u 4 def
/a [-1 s -1 s 1 s 1 s -1 s] def %abcissae corners island (clockwise=>inside, scaled)
/b [-1 s 1 s 1 s -1 s -1 s] def %ordinates corners island (clockwise=>inside, scaled)
/v 4 def
/c [0 .3333 .5    .6667 ] def %abcissae corners broken model line
/d [0 0     .2887 0 ]     def %ordinates corners broken model line
0 kronkel stroke gsave
110 0 translate 1 kronkel stroke grestore
showpage
```

A triangular island (0, 0), (1, 0) (.5, .866) (0, 0) can equally-well be specified, with the broken model line (0, 0), (.5, 0), (.375, .2165), (.5, 0), (.625, .2165), (.5, 0).



The degenerate Lévy island can be specified by the line (-1, 0), (1, 0), with the (broken) model line (0, 0), (.5, 0)

```
%...
(C:\\PSlib\\PSlib.eps) run
%globals s, u, a, b, v, c, d
/s {30 mul} def
/u 1 def /a [ -1 s 1 s ] def%abcissae corners line (scaled)
         /b [  0   0   ] def%ordinates corners line (scaled)
/v 2 def /c [ 0  .5] def    %abcissae corners broken model line
         /d [ 0  0 ] def    %ordinates corners broken model line
                0 kronkel stroke
110 0 translate 1 kronkel stroke
%...
showpage
%%EOF
```



An interesting program to experiment with. The PS transcription is also included in my `PSlib`. Lauwerier provides moreover variants: `KRONKEL`**T**, biased by the number system (Dutch **t**alstelsel) approach and `KRONKELB`, where **b**acktracking has been used.

Minkowski fractal

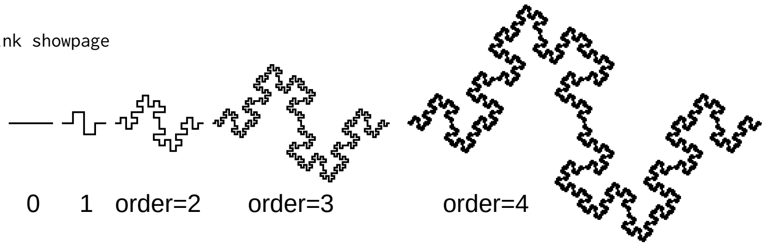
Much similar to the von Koch fractal is the Minkowski fractal, called sausage by Mandelbrot. The replacement scheme can be distilled from the illustration below, especially $M_0 \rightarrow M_1$.

```
10 REM ***Sausage of Minkowski***
10 DIM A(7) : A(0)=0 : A(1)=1 : A(2)=0 : A(3)=3
20           A(4)=3 : A(5)=0 : A(6)=1 : A(7)=0
30 P=3 : DIM T(P) : REM***order***
20 H=4^(-P) : X=0 : y=0: PSET (0,0)
30 FOR N=0 TO 8^P-1
40 M=N : FOR L=0 TO P-1
50       T(L)=M MOD 8 : M=M\8 : NEXT L
```

```
60 S=0 : FOR K=0 TO P-1
70       S=(S+A(T(K))) MOD 4: NEXT K
80 IF S=0 THEN X=X+H
81 IF S=1 THEN Y=Y+H
82 IF S=2 THEN X=X-H
83 IF S=3 THEN y=y-H
90 LINE -(X, Y)
91 NEXT N
92 END
```

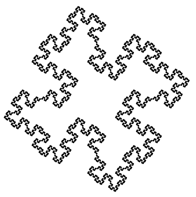
Length of line-piece is wired-in.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Minkowski sausage by H.A Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl,m April 2011
/mink{/p exch def%order, global l length of initial line
/a [ 0 1 0 3 3 0 1 0] def%model specification by direction numbers
/t p array def 0 0 moveto
/h l 4 p exp div def
0 1 8 p exp 1 sub{/m exch def
 0 1 p 1 sub{t exch m cvi 8 mod put /m m cvi 8 idiv def}for
 /s 0 def
 0 1 p 1 sub{/s a t 4 -1 roll get get s add cvi 4 mod def}for
 s 0 eq { h     0     rlineto}if
 s 1 eq { 0     h     rlineto}if
 s 2 eq { h neg 0     rlineto}if
 s 3 eq { 0     h neg rlineto}if
}for stroke} def
%
/l 100 def 3 mink showpage
%%EOF
```



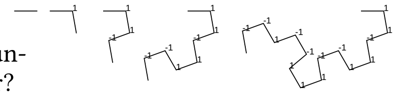
The fractal dimension of the Minkowski fractal $D = \frac{\log 8}{\log(1/4^{-1})} = 1.5$. The array a contains the direction numbers: 0, 1, 3, meaning direction 0°, 90°, -90°, respectively.

Minkowski island The essentials of the island program in PS are given below.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Minkowski island
%%...
/l 200 def                          4 mink
gsave l 0     translate  -90 rotate 4 mink grestore
gsave l l neg translate -180 rotate 4 mink grestore
      0 l neg translate -270 rotate 4 mink
showpage
%%EOF
```

Dragon figures



Folding a strip of paper repeatedly and after unfolding one may ask: How to draw the meander?

The curve with rounded 90° corners is named Dragon curve by Heighway. The curve does not intersect itself. A nice example for developing the mathematical problem solving attitude in discovering the intriguing pattern. (Be aware of folding consistently in the right direction.)

Let us set up a table, where for each line piece the continuation angle is given: r means rotate $-90°$, and l means rotate $90°$, and unearth the regularity in the directions $d(n)$, for $n = 1, 2, 3, \ldots$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
r	r	l	r	r	l	l	r	r	r	l	l	r	l	l	r	r	r	l	r	r	l	l	l	r	r	l	l	r	l	l	r

$d(n) = r(ight)$ for $n = 1, 5, 9, \ldots$ $d(n) = l(eft)$ for $n = 3, 7, 11, \ldots$ $d(n) = d(n/2)$ for n is even.

Express n in the form $k \times 2m$ where k is an odd number. The direction of the nth turn:

if $k \,\mathbf{mod}\, 4 = 1$ then the nth turn is r;

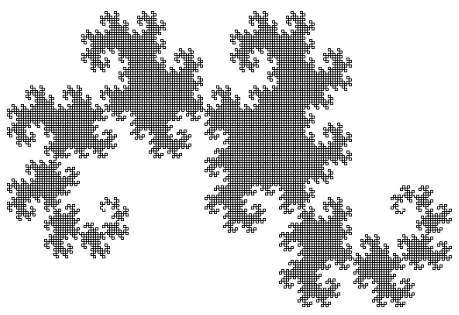
if $k \,\mathbf{mod}\, 4 = 3$ then the nth turn is l.

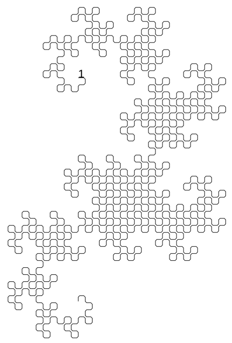
The direction of turn 76376: $76376 = 9547 \times 8$ & $9547 \,\mathbf{mod}\, 4 = 3 \rightarrow d(76376) = l$.

```
10 REM ***Draak***
10 P=3 : REM***order***
20 H=2^(-P/2) : A=1.7453 : REM ***Corner***
30 B=PI-A : X=H : Y=0 : LINE (0,0)-(H,0) : S=0
40 FOR N=1 TO 2^P-1 : M=N
50 IF M MOD 2 = 0 THEN M=M/2: GOTO 50
60 IF M MOD 4 =1  THEN D=1 ELSE D=-1
70 S=S+D
80 X=X+H*COS(S*B)
90 Y=Y+H*SIN(S*B) : LINE -(X,Y)
91 NEXT N
92 END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Draak by H.A Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl, April 2011
/draak{%draak curve with angle A and order p, global scaling s
       %angle p ==> dragon curve
{/p exch def /b 180 3 -1 roll sub def%order p and angle b
/h 2 p -2 div exp s def %size piece scaled
0 0 moveto h 0 lineto
1 1 2 p exp 1 sub{%for n
 /m exch def
 {m cvi 2 mod 0 eq {/m m 2 div def}{exit}ifelse}loop
  m cvi 4 mod 1 eq {/d b neg def}{/d b def}ifelse
  d rotate h 0 rlineto%no currentpoint translate necessary
}for %n
}bind def %in library for local variables with draakdict
%
/s {30 mul} def 100 3 draak stroke showpage
%%EOF
```

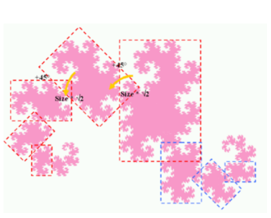
For the order $p = 14$ and angle $90°$ I reproduced Lauwerier's result in PS, see below at left.



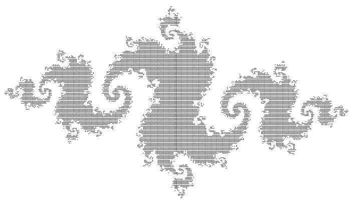


The number of line pieces is 2^p. The curve of order 10 with rounded corners is at right. The curves don't intersect themselves, which is seen in the figure with rounded corners. (In Lauwerier's program the direction `D` is not in agreement with the folded paper and the dragon figure. This is adapted in the PS code.)

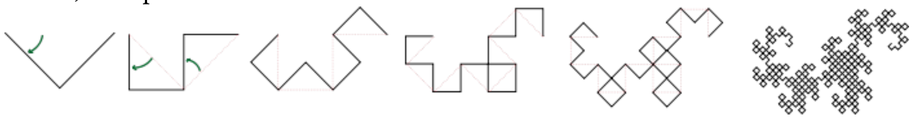
Knuth in the TEXbook Appendix D p390 also mentions the dragon curve in relation to Turtle Graphics, and draws dragon figures in TEX. When I tried the order 12 in TEX, in 1995, TEX gave the error message 'TEX capacity exceeded.'



Dimensions The bounding box obeys the proportion 3 : 2. The fractal dimension of the curve equals 2, a local plane filling curve (Courtesy http://en.wikipedia.org/wiki/Dragon_curve, which mentions more properties of the Dragon curve, such as its self-similarity and the spiral shape.)
At right a filled dragon-like Julia set.(More on Julia fractals: JULIA fractals in PostScript, submitted for MAPS.)

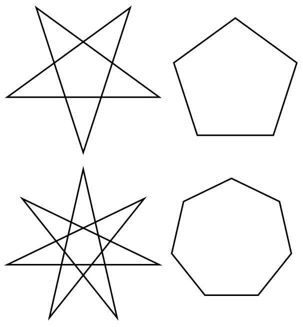


The Dragon curve can be generated similarly to the rewriting scheme of the Lévy fractal, with parts rewritten mirrored.



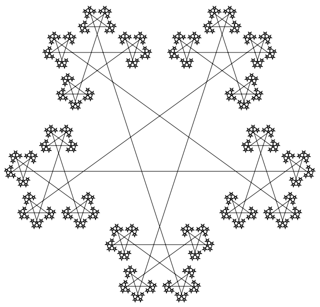
Star fractals

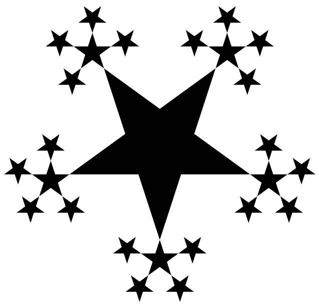
As introduction a generalization of the program star of the Blue Book p51. The program is more general because it allows to draw the pentagram or the 5-star depending on the value of the angle parameter. Moreover, the number of vertices can be varied, to obtain for example a heptagon casu quo 7-star (heptagram).

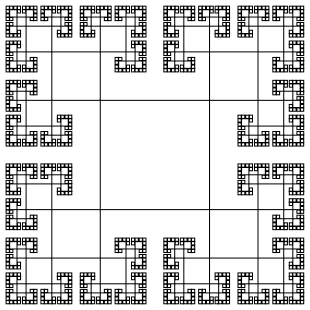


```
/gonstar%p (order) v ==> star
{gonstardict begin /v exch def /angle exch def
0 0 moveto
v{angle rotate l 0 rlineto}repeat closepath
end} bind def
/gonstardict 2 dict def
%%EndProlog
%
%Program ---the script---
%
/l 100 def   144  5 gonstar stroke
gsave 75 -25 translate
/l 50 def 1.415 setmiterlimit
72 5 gonstar stroke grestore
gsave 0 -110 translate
/l 100 def   1080 7 div  7 gonstar stroke
grestore
gsave 65 -130 translate
/l 35 def 1.415 setmiterlimit
360 7 div 7 gonstar stroke
showpage
```

Lauwerier's ingenious, concisely programmed star fractal illustrations, left and right below, consist also of 1 (broken) line.







```
10 REM ***Star***
10 P=5 : REM***order***
20 V=4: A=.8*3.141593 : R=.35
30 PSET (0,0) : X=0 : Y=0
40 FOR N=0 TO (V+1)+V^(P-1)-1
50    M=N : B= N*A : F=0
60    IF M MOD V <> 0 OR F>=P-1 THEN GOTO 80
70      F=F+1 : M=M\V : GOTO 60
80    X=X+R^(P-F-1)*COS(B)
90    Y=Y+R^(P-F-1)*SIN(B)
100   LINE -(X,Y)
110 NEXT N
120 END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Starfractal by H.A Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl, May 2011
/starfractal%reduction angle p (order) v ==> starfractal
{starfractaldict begin
 /v exch cvi def /p exch def /a exch def /r exch def
 0 0 moveto
 0 1 v 1 add v p 1 sub exp mul 1 sub
   {/n exch cvi def
    /m n def /f 0 def
    {m v mod 0 ne   f p 1 sub ge   or
     {exit}
     {/f f 1 add def /m m v idiv def}
     ifelse
    }loop
    r p f sub 1 sub exp s 0 rlineto
    a rotate
   }for %n
 end} bind def
/starfractaldict 8 dict def
%
starfractaldict begin /s {300 mul} def end % scaling
.3 144 3 4 starfractal fill % r=.3 a=144 order=3 vertiges-1=4
showpage
%%EOF
```

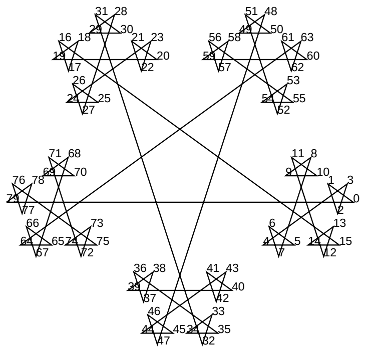
Remarks

scaling factor has been added in PS
user space is rotated by the angle after each line
parameter driven

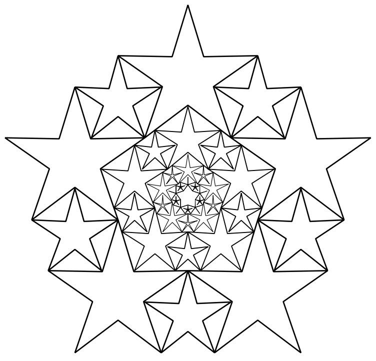
The algorithm is based on that consecutive line pieces make a constant angle, only the line size varies. For the order 5 we have 5 different lengths of the line pieces

1	n=0, 256, 512, 768, 1024, …
r	n=64, 128, 192, **320**, 384, 448, …
r^2	n=16, 32, 48, **80**, 96, 112, …
r^3	n=4, 8, 12, **20**, 24, 28, **36**, 40, 44, **52**, …
r^4	n=1, 2, 3, **5**, 6, 7, **9**, 10, 11, **13**, …

In order to follow the way the drawing has been made sequential numbers have been added in the accompanying illustration.



Another 5-star composition has been published in my Tiling in PS and Metafont in MAPS 97.2. I copied the program from the article, adapted it to EPSF, et voilà. In the middle a composition borrowed from Helmstedt created by a Lindenmayer production rule in Mathematica. At right a nice illustration from Lauwerier(1990), which reminds me of Escher's limit cycles.

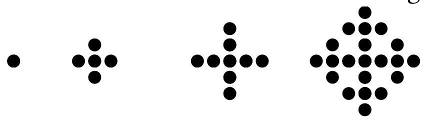


Game of Life

Lauwerier(1990) mentions a fractal which he obtained from the Pickover variant of the Game of Life, made popular by Martin Gardner in a Scientific American in 1970. The game is played on a grid. Each node can be alive or dead. Once alive it stays alive. If dead it comes to life if only one neighbour, N, E, S or W is alive. On each heartbeat the whole grid is inspected in parallel. Lauwerier's BASIC program is given below.

```
***naam: PICK1***
40 DEFINT I, J, K, N, T, X, Y
70 IF SCR=9 THEN XM=320 : YM=175
80 IF SCR=12 THEN XM=320 : YM=240
100 INPUT "NUMBER OF ROWS=", N
120 DIM X(N,N), Y(N,N)
130 X(0,0)=1
140 FOR K=1 TO N-1
150 FOR I=0 TO K : FOR J=0 TO K-I
160 IF X(I,J)=0 THEN GOSUB 220 ELSE GOSUB 270
170 NEXT J : NEXT I
180 FOR I=0 TO K : FOR J=0 TO K-I
190 X(I,J)=Y(I,J)
200 NEXT J : NEXT I : NEXT K
210 A$=INPUT$(1) :END
220 IF I>=1 AND J>=1  THEN T=X(I+1,J)+X(I-1,J)+
                           X(I,J+1)+X(I,J-1)
230 IF I=0 AND J>=1  THEN T=2*X(1,J)+
                           X(0,J+1)+X(0,J-1)
240 IF I>=1 AND J=0  THEN T=2*X(I,1)+
                           X(X+1,0)+X(I-1,0)
250 IF T=1 THEN Y(I,J)=1
260 RETURN
270 PSET (XM+2*I, YTM-2*J),14 : PSET (XM-2*I, YM-2*J),14
280 PSET (XM+2*I, YTM+2*J),14 : PSET (XM-2*I, YM+-2*J),14
290 RETURN : END
```

If we start with 1 alive node then the generations 1, 2, 3, 4 look as follows

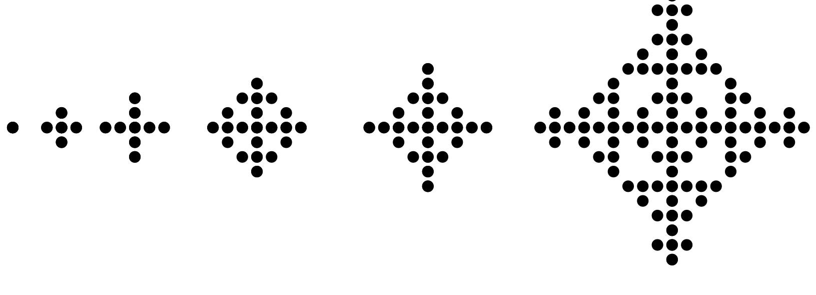


Lauwerier's program is computational intensive.

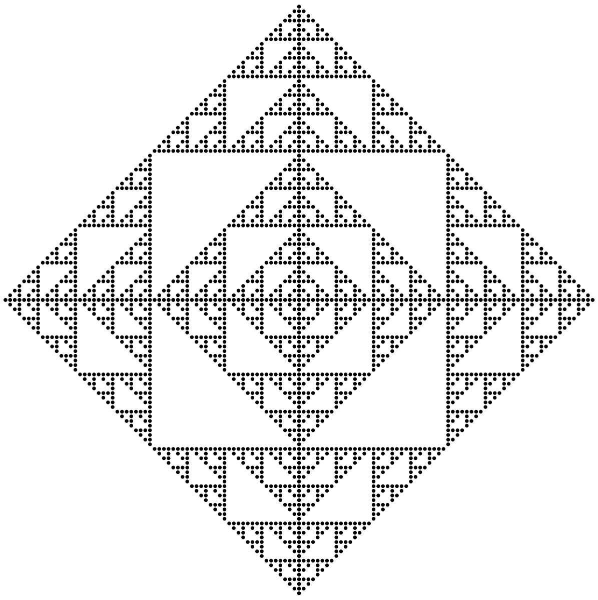
My translated version could not reproduce Lauwerier's result in reasonable time.

I simplified the program by inspecting on each heartbeat only the new contra diagonals ($i + j$ =constant) in the first quadrant.

The 1 …6 generations look



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth Cell model a la Pickover, simplified
%%Author:  Kees van der Laan
%%Date: March 2012
%%BoundingBox: -195 -195 195 195
%%BeginSetup
%%EndSetup
%%BeginPrologue
%%DocumentFonts: Times-Roman
/Times-Roman 20 selectfont
/printdot{3 i mul -3 j mul moveto (.) show
          3 i mul +3 j mul moveto (.) show
         -3 i mul -3 j mul moveto (.) show
         -3 i mul +3 j mul moveto (.) show
         }def
/alive?{%check whether cell has become alive
  i 1 ge j 1 ge and
    {ax i 1 sub n mul j       add get
     ax i       n mul j 1 sub add get add}if
  i 0 eq j 1 ge and
    {2 ax  n  j   add get mul
       ax     j 1 sub get add}if
  i 1 ge j 0 eq and{ax i 1 sub n mul get}if
  1 eq{ax i n mul j add 1 put printdot}if
}def%alive?
/pickover{% stack integer>=0 the order==> cell pattern
/n exch def
/ax n 1 add dup mul array def                    %array
1 1 n n mul{/k exch def ax k 0 put}for ax 0 1 put%initialize
/i 0 def /j 0 def printdot
1 1 n 1 sub {/k exch def
  0 1 k{/i exch def /j k i sub def%contradiagonal
        alive?
       }for%i
  }for%k
}def
%%EndProlog
%
% Program
%
64 pickover showpage
%%EOF
```



The point Lauwerier wanted to make — the game yields fractal patterns — is also obtained by this simplified game.

Annotated References

- An introductory survey: `http://en.wikipedia.org/wiki/Dragon_curve`.
- Adobe Red, Green and Blue Books. The musts for PS programmers.
- Biography of H.A. Lauwerier: `http://bwnw.cwi-incubator.nl/cgi-bin/uncgi/alf`.
- Gleisk, J(1987): CHAOS — making a new science. Penguin.
  (An introduction to and survey of the world of non-linearity, strange attractors and fractals.)
- Goossens, M(2007, sec ed) et. al.: LaTeX Graphics Companion. ISBN 978 0 321 50892 8.
- Helmstedt, J(2011): A New Method of Constructing Fractals and Other Graphics. The Mathematica Journal. (Nice examples of Lindenmayer systems, for which Lauwerier's KRONKEL can be used.)

- Jackowski, B, P. Strelczyk, P. Pianowski(1995-2008): PSView5.12. WWW. `bop@bop.com.pl`. (Extremely fast previewer for `.eps` among others, which allows `PSlib`(rary) inclusion via the `run` command).
- Knuth, D.E, T. Larrabee, P.M. Roberts(1989): Mathematical Writing. MAA notes 14. The Mathematical Association of America.
- Knuth, D.E(1990, 10th printing): The TEXbook. Addison-Wesley. ISBN 0-201-13447-0. (A must for plain TEXies.)
- Lauwerier, H.A(1987): FRACTALS — meetkundige figuren in eindeloze herhaling. Aramith. (Contains programs in BASIC. Lauwerier, H.A (1991): Fractals: Endlessly Repeated Geometrical Figures, Translated by Sophia Gill-Hoffstadt, Princeton University Press, Princeton NJ1. ISBN 0-691-08551-X, cloth. ISBN 0-691-02445-6 paperback. "This book has been written for a wide audience ... " Includes sample BASIC programs in an appendix. Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1988): The Pythagoras Tree as Julia Set. CWI-Newsletter.
- Lauwerier, H.A(1989): Oneindigheid — een onbereikbaar ideaal. Aramith. ISBN 90 6834 055 7. (Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1990): Een wereld van FRACTALS. Aramith. ISBN 90 6834 076 X. (Sequel and updated version of Lauwerier(1987). Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1994): Spelen met Graphics and Fractals. Academic Service. ISBN 90 395 0092 4. (An inspiring book with Math at the high school level for a wide audience; the BASIC programs I consider outdated for direct use. Audience: Instructors, (high-school) students, and the educated layman.)
- Peitgen, H.O, H.Jürgens, D. Saupe(2004 sec. ed.): Chaos and Fractals. New frontiers of Science. (Images of the fourteen chapters of this book cover the central ideas and concepts of chaos and fractals as well as many related topics including: the Mandelbrot set, Julia sets, cellular automata, L-systems, percolation and strange attractors. This new edition has been thoroughly revised throughout. The appendices of the original edition were taken out since more recent publications cover this material in more depth. Instead of the focused computer programs in BASIC, the authors provide 10 interactive JAVA-applets for this second edition via `http://www.cevis.uni-bremen.de/fractals`. An encyclopedic work. Audience: Accessible without mathematical sophistication and portrays the new fields: Chaos and fractals, in an authentic manner.)
- Swanson, E(1986, revised ed): Mathematics into Type. American Mathematical Society.
- Szabó, P(2009): PDF output size of TEX documents. Proceedings EuroTEX2009/ConTEXt, p57–74. (Various tools have been compared for the purpose.)
- Van der Laan, C.G(1992): LIFO and FIFO sing the Blues. MAPS 92.2.
- Van der Laan, C.G(1995): Publishing with TEX. Public Domain. (See TEX archives. `BLUe.tex` comes with `pic.dat` the database of my pictures in TEX-alone.)
- Van der Laan, C.G(1997): Tiling in PostScript and MetaFont — Escher's wink. MAPS 97.2.
- Van der Laan, C.G(unpublished, BachoTEX workshop): TEXing Paradigms. (A plea is made for standardized macro writing in TEX to enhance readability and correctness.)
- Van der Laan, C.G(submitted EuroTEX2012): Julia fractals in PostScript.
- Veith, U(2009): Experiences typesetting mathematical physics. Proceedings EuroTEX2009/ConTEXt, p31–43. (Practical examples where we need to adjust TEX's automatic typesetting.)

Conclusions

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

I don't know how to include the results of the BASIC programs elegantly in publications. The results of the PS programs can be easily included in pdf(La)TeX, Word, ... documents.

PS' variable user space and recursion alleviated programming, with concise `defs` and programs as readable as literature, but ... be aware of its subtleness. PostScript's variable user space was the key to my adaptation of production rules. Because of PS' subtleness not many people program in PS, I presume, or ... do they consider it of too low-level?

In programming self-similarity the awareness of orientation is paramount. I did not find classical Math fractals in PS on the WWW, only one Sierpiński curve in Java.

Lauwerier's analysis — associating binary, ternary, ... tree structures with binary, ternary, ... numbers, is an eye-opener. In his, and my, programs all the self-similar sub-curves are draw anew. In Metafont/-Post we could just build the paths and splice them suitably into paths of higher order, as I did in the past with the Pythagoras Tree in Metafont.

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

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

TEX mark up For the symbols of the number systems I, N, Q, R, C, which curiously are not provided for in plain TEX, I use the the AMS (blackboard) font `msbm10`.

The $\overset{\rm L}{=}$ and $\overset{\rm R}{=}$ composed relational operators are marked up by `\mathrel {\mathop =^{\rm L}}` and not by `$\buildrel\rm L \over=$`, TEXbook p437; the latter is OK for the stacked composed symbol as such.

For typesetting tables `\halign` and the tabbing mechanism have been used (TEXbook ch22). The 11-element of one of the tables needs an oblique line. I provided for this in PS, which is simpler and not restricted by obliqueness. In 1995, in my PWT guide, I used the GKP macros for this, which suffer from the same inconvenience as LATEX's picture environment: restricted obliqueness.

A blank line before display math yields too much white space! This blank line is important, though, in order to avoid widows.

Locally I have used for parallel listings of program texts `vbox-s` next to each other, which inhibits proper page breaks. I don't know how to provide macros for local elegantly marked up multi-column texts, which allow page breaks. (I also tried `\valign`, alas in vain.) My inserted pictures suffer from the same inconvenience as in Word: changing the text might disturb the layout, such that the pictures will become ill-placed.

As known, I could not use footnotes from within a `vbox`; kludged around.

In TEXworks I used the Terminal font in the edit window with the pleasing effect that comments remain vertically aligned in the `.pdf` window.

Conversion of my TEX script into Word made me (hands-on) aware of the differences between TEX and Word. If you are after utmost accurate, user-controlled typeset Mathematics then TEX is to be preferred. For bread and butter Mathematics Word can do, especially with Cambria, I presume. I did not find in Word (MS equation 3.0) the possibility to discern between displayed Math and in-line Math. Tables in Word are tricky, the WYSIWYG approach does not always yield the table layout you are after. As in TEX I can't make an appropriate 11-element. I could not handle the inclusion of a PS made 11-element in Word. Program texts, as columns in a table, don't suffer from difficulties in allowing page breaks. A pre-index, as is usual with hyper-geometric functions, I could not nicely typeset with MS equation 3.0.

Inclusion of the `.jpg` figures and `.pdf` objects went smoothly. I had to convert `.png` objects. The `inclusion of EPSF object` option did not work on my PC, though the option is available. It invoked Adobe Illustrator CS2 12.0.0 and fell silent. The same EPSF invoked by AI directly worked. Maybe incompatible versions? Neglecting superfluous spaces, which TEX does automatically, has been lost in the conversion. A local change in Word might change the document more than local, beyond user control. I don't know how to switch off, or change, pre-settings, such as: don't underline automatically WWW addresses, maybe by de-activating the option `WWW addresses as hyperlinks`?

Conversion also entailed splitting up the original (concept) paper and rewriting the parts into 2 new papers. Converting back into TEX, after changes were made, was more difficult than converting into Word.

After I had finished I became aware of Acrobat Pro X which also converts `.pdf` into a Word document.

Acknowledgements

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

Thank you Jos Willink for proofing, Wim Wilhelm for drawing my attention to cellulaire automata, and the LATEX graphics environment `asymptote` which he integrated in TEXnicCenter; I don't have experience with them, but they may be of interest for LATEX users. MAPS editors for improving my use of English and Taco Hoekwater for procrusting my plain TEX note into MAPS format.

Thank you Zinadia Nikolaevna Gulka for inviting me to submit a paper or two for the 'Informatsionnie Texnologii i Matematicheskoe Modelirovanie' journal', and her co-worker for stimulating me to convert the TEX marked up ASCII source of an early version of this note into the required Word. The invitation stimulated me to adapt and revise the material. Thank you GUST for publishing a previous version of this note in the BachoTEX2012 proceedings. Thank you Svetlana Morozova for prompting me in the use of Word.

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

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

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

The bad news with respect to `.eps` into `.pdf` conversion is that the newest Acrobat 10 Pro X does not allow for the `run` command for library inclusion.

Notes

1. Alas, the `\psfig` has been lost in pdfTEX. Happily, ConTEXt and LuaTEX allow direct EPSF inclusion.
2. Lauwerier(1989) narrates what mathematicians thought about the ∞-concept through the ages from the ancient Greeks onward.
3. Acrobat Pro X does not allow for the library inclusion via `run`, alas :-(. BASIC is interactive, PS is batch-oriented.
4. Barnsley is famous for his fern fractal. In his later works he is more ambitious and constructs fractals given a picture, on demand.
5. The 'fixed-point' of the production rule is the fractal. At right my old TEX code is displayed with its result.
6. The 'fixed-point' of the production rule is the real fractal.

My case rests, have fun and all the best.

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

Appendix: Fractal Dimension

The need arose to associate various fractal curves with numbers, to characterize them. Mathematicians came up with definitions which were generalizations and compatible extensions of the classical, topological dimension notion. The fractal dimension à la Kolmogorov(1958) is based upon covering the fractal with a grid and counting the cells of the grid which contain points of the fractal, the so-called box-counting dimension. The definition reads

$$D = \lim_{s\to 0} \mathbf{log} N/\mathbf{log}(1/s),$$

with N the number of cells which contain points of the fractal and s the size of the side of a cell. Let us calculate the (fractal) dimension of a square in order to verify the compatibility of the definition with the classical fact. Cover the (unit)square with a grid which has s as side of a cell, then we have $1/s^2$ cells which cover the unit square, and therefore $D = \mathbf{log}(1/s^2)/\mathbf{log}(1/s) = 2$, QED.

All plane-filling fractal curves have fractal dimension $D = 2$.

For the calculation of the fractal dimension of fractals defined by contracted mappings, with contraction factors f_1, f_2, f_3, ... Lauwerier(1990) mentions the Hausdorff formula

$$\sum_i f_i^D = 1.$$

For the Lévy fractal, which can be defined by 2 contractions each with contraction factor $1/\sqrt{2}$, we arrive at

$$2 * (1/\sqrt{2})^D = 1 \rightarrow D = 2.$$

For the von Koch fractal, which can be defined by 4 contractions each with contraction factor 1/3, we arrive at

$$4 * (1/3)^D = 1 \rightarrow D = \mathbf{log}4/\mathbf{log}3 \approx 1.26.$$

For the Sierpiski sieve, which can be defined by 3 contractions with contraction factors 1/2, we arrive at

$$3 * (1/2)^D = 1 \rightarrow D = \mathbf{log}3/\mathbf{log}2 \approx 1.58.$$

For the Menger sponge, which can be defined by 8 contractions each with contraction factor 1/3, we arrive at

$$8 * (1/3)^D = 1 \rightarrow D = \mathbf{log}8/\mathbf{log}3 \approx 1.89.$$

The details of the definition of the fractal dimension D by Hausdorff (1919) are cumbersome, and serve a theoretical need. The calculation of other fractal dimensions: self-similarity dimension, and compass dimension (for coast lines) goes beyond the scope of this paper, see Peitgen c.s.(2004). My purpose is that one knows about the concept as a grey box with nodding knowledge, that one knows some fractal dimensions for simple cases, and ... that one does not become frightened or confused on encounter.

For a survey of various fractals and their dimensions see Peitgen c.s.(2004) and/or `http://en.wikipedia.org/wiki/List_of_fractals_by_Haussdorf_dimension`.

Finally, do you know that the path of Brownian movement is a fractal with fractal dimension 2, and ... do you know that the Cantor Dust has Hausdorf dimension $D_H = \log 2/\log 3 \approx .63$, which differs from the fractal dimension $D \approx .69$?

Appendix: Cantor Dusts

The Cantor Dust has been shown at the beginning of this note. The idea is that each line is replaced by its left and right third parts. This pattern can be repeated yielding what is called a fractal nowadays. (It is included in `pic.dat`, the library of (TEX-alone) pictures which come with `BLUe.tex`.)

In Publishing with TEX(1995) the Cantor Dust example by TEX alone was included.

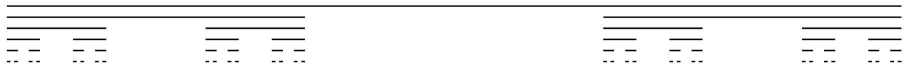
The PS code for Cantor Dust of order n has a similar production rule as for the von Koch fractal:

$$CD_n = [S_{\frac{1}{3}\frac{1}{3}} CD_{n-1}] \oplus [S_{\frac{1}{3}\frac{1}{3}} T_{\frac{2s}{3}0} CD_{n-1}] \quad \text{with}$$

CD_0 the initial line,
CD_n the Cantor Dust of order n,
$\oplus$ splice operator, meaning add properly the second piece to the set,
[open a new GS on the GS stack,
] remove current graphics state from the GS stack and recall previous,
$S_{a,b}$ 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.
The above PS production rule transcribes systematically into the following PS def.[5]

```
%!PS-Adobe-3.0 EPSF-3.0
/cd%integer n>=0 ==> Cantor Dust of order n
{1 sub dup -1 eq
 {0 0 moveto s 0 lineto stroke}
 {gsave .3333 1 scale                      dup cd grestore
  gsave .3333 1 scale  2 s mul 0 translate dup cd grestore
 }ifelse 1 add
}def
%%EndProlog
%
% Program
%
%...
/s 300 def 4 cd pop
   305 -2  moveto TR10 setfont (Cantor) show
     0 -3 rmoveto TR7  setfont (4) show 0 -10 translate
%...
showpage
%%EOF
```

```
%Borrowed from BLUe's pic.dat
\newcount\x\newcount\y\newcount\width
\newdimen\unitlength
\def\E#1{\hbox to 0pt{\kern\x\unitlength
   \vbox to 0pt{\vss\hrule width#1\unitlength
             \kern\y\unitlength
            }\hss
   }\advance\x#1 }
\def\cf{\ifnum0=\width \fc\fi
  {\E{\the\width}}\divide\width3
   \advance\y-3
   {\cf}\advance\x\width
        \advance\x\width
   {\cf}\relax}%
\def\fc#1\relax{\fi}%
$$\width243 \unitlength1pt
  \x-\width \divide\x2
  \cf$$
```



The ‘fixed point’ of the production rule is the Cantor Dust fractal. For just showing a few approximations of the Cantor Dust the TEX code will do. Lauwerier(1987) provides a tiny BASIC program KAM. He also associates the Cantor dust with the trinary number system because the interval is divided in 3 pieces, repeatedly. The Cantor dust of [0, 1] consists of the trinary fractions where only the digits 0 and 2 occur, Lauwerier(1987, p26). An eye-opener!

Cantor Dust as IFS Lauwerier(1989, ch8) constructs the Cantor Dust of high order by the IFS

$$x_{n+1} \overset{L}{=} x_n/3 \qquad \text{and} \qquad x_{n+1} \overset{R}{=} x_n/3+2/3 \qquad n=0,1,2,\ldots \quad x_0=1/3.$$

Again an eye-opener! His tiny program and my conversion read

```
X=.3        'start
FOR K=1 TO 100
   IF RND<.5 THEN X=X/3 ELSE X=(X+2)/3
   PSET(X, 0)'scale X if wanted
NEXT K
END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Cantor Dust via IFS
%%Author: H.A. Lauwerier(1989): Oneindigheid
%%Transcriptor: C.G. van der Laan, may 2012
%%BoundingBox: -1 39 1010 70
%%BeginSetup
%%EndSetup
%
% Program
%
/Courier 10 selectfont
22121943 srand /x .3 def%start
100{rand 1073741823 gt{/x x 2 add 3 div def}
                      {/x x 3 div def}ifelse
    x 1000 mul 50 moveto (.) show
   }repeat
showpage
%%EOF
```

If you, kind reader, shrug shoulders about paying so much attention to such a tiny problem, I can only say that if you don't analyse tiny problems deeply, your solutions of bigger problems will lack ingenuity.

Generalization to 2D yield the so-called Sierpiński carpets.

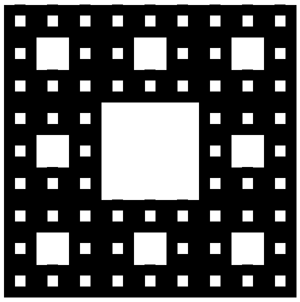
The algorithm reads: divide a square in 3^2 equal sub-squares and delete the middle, or the middle cross, and do this repeatedly for the remaining 8 squares.

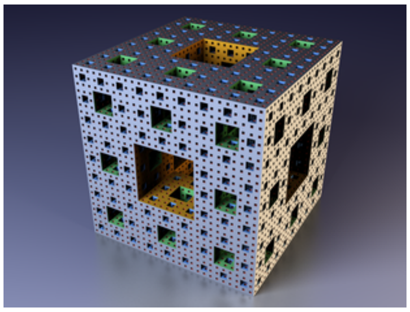
At the EuroTEX95 Bogusłav Jackowski showed his variant, probably in connection with his mftoeps program, which transforms Metafont code into PS. (I did not use mftoeps, because it was PC-biased, and I had a Mac Powerbook 150.) This was at the time that MetaPost, the preprocessor for PS with Metafont-biased user-language, was not yet released in the public domain.

In my opinion he was on the right track: PS is mandatory for graphics to be included in documents. It is just a pity he did not pursue that PS alone, or better EPSF, is suitable for constructing graphics to be included in documents. His collaborators, Pjotr and Pjotr, pursued PS in PSview.

I programmed the Sierpiński carpet, it is not the gasket, in TEX and in Metafont after the conference. For historical reasons I have included the programs below. The black-and-white figure is created by TEX-alone on-the-fly. (I just copied it from the revised 1996 FIFO-article, and see,... it still works. The MF program also still works on my museum Mac Powerbook 150 of 1995!)

The picture at right has been borrowed from the WWW. It illustrates a generalization of the good old Cantor Dust into 2.5D, also called Menger sponge. Both pictures are more interesting and more beautiful than the original 1D Cantor Dust.





```
tracingstats:=1;proofing:=1;screenstrokes; %
pickup pencircle scaled 1;
def sierpinskisquare (expr s, p)=
if s>5:unfill unitsquare scaled .333s
      shifted (p+.333s*(1,1));
   sierpinskisquare(.333s, p);
   sierpinskisquare(.333s, p+(.333s,0));
   sierpinskisquare(.333s, p+(.667s,0));
   sierpinskisquare(.333s, p+(0,.333s));
   sierpinskisquare(.333s, p+(0,.6673s));
   sierpinskisquare(.333s, p+(.667s,.333s));
   sierpinskisquare(.333s, p+(.667s,.667s));
   sierpinskisquare(.333s, p+(.333s,.667s));
fi enddef;
%
s=100; fill unitsquare scaled s;
sierpinskisquare(s,origin);
showit;
end
```

```
\newdimen\x\newdimen\y\newdimen\size\newdimen\lsize
\def\sier{\ifdim\size<35pt \reis\fi
   \divide\size3
  {\sier}{\advance\x\size\sier}{\advance\x2\size\sier}%
  {\advance\y\size{\sier}{\advance\x2\size\sier}}%
  \advance\y2\size{\sier}{\advance\x\size\sier}%
                      \advance\x2\size\sier}
\def\reis#1\sier{\fi\putatxy\draw}
%with auxiliaries
\def\putatxy#1{\vbox to0pt{\vss
   \hbox to0pt{\kern\x#1\hss}\kern\y}}
\def\draw{{\lsize\size\divide\lsize3
   \rlap{\vrule height\size width\lsize
   \vbox to\size{\hrule width\lsize height\lsize\vss
                 \hrule width\lsize height\lsize}%
      \vrule height\size width\lsize}}}
```

```
$$\vbox to120pt{\vss
  \offinterlineskip\size120pt\x=-.5\size\y0pt
  \sier}$$
```

A PS program for a 2.5D Cantor Dust is cumbersome, because it has to deal with projection and has to handle hidden lines.

Appendix: Hilbert Curve

Hilbert curves H_1 … H_6 have been shown in the introduction. Wirth(1975) I consider a starting point for programming a Hilbert curve. Wirth could not know about the rotation of US facility, nor did PS exist. The self-similarity property of the H-curve, i.e. a curve is composed of (rotated) curves of one order lower, he programmed by four rotated instances in (recursive) procedures A, B, C and D, which entailed a more complex recursion scheme. For those who don't own the 1975 book I have included the program and the procedure A. (Procedures B, C and D are similar.)

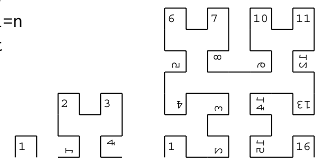
The left PS-program below is based on the production rule[6]

$$H_i = {}_mH^{90}_{i-1} \oplus \uparrow \oplus H_{i-1} \oplus \rightarrow \oplus H_{i-1} \oplus \downarrow \oplus {}_mH^{-90}_{i-1}, \quad \text{for } 1 = 1, 2, \ldots$$

where $\oplus$ means spliced, the superscript denotes the rotation angle, the arrows $\uparrow \rightarrow \downarrow$ mean draw a segment in the direction North, East, and South. In order to splice correctly, the rotated copies have also to be mirrored, which is indicated by the pre-index m.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Hibert curve, Feb 2012
%%Author: Kees van der Laan, kisa1@xs4all.nl
%%BoundingBox: -1 -1 321 151
%%BeginSetup
%%EndSetup
%%BeginProlog
/-s  s neg def
/lineN{0 0 moveto 0 s  lineto currentpoint stroke translate}def
/lineS{0 0 moveto 0 -s lineto currentpoint stroke translate}def
/lineE{0 0 moveto s 0  lineto currentpoint stroke translate}def
%
/Hilbert{%on stack order >=1 ==> Hilbert curve
        %s size of line segment (global),
1 sub dup 0 gt
{90 rotate  1 -1 scale Hilbert  1 -1 scale -90 rotate
  lineN                Hilbert
  lineE                Hilbert
  lineS
 -90 rotate 1 -1 scale Hilbert 1 -1 scale  90 rotate
 }if 1 add%reset order
} def
%%EndProlog
%
%Program ---the script---
%
/s 10 def %size of segment
                  1 Hilbert pop
   s    0 translate 2 Hilbert pop
2 s mul 0 translate 3 Hilbert pop
3 s mul 0 translate 4 Hilbert pop
showpage
%%EOF
```

```
program Hilbert(pf,output);
{plot Hilbert curves of orders 1 to n. Wirth(1975).}
const n= 4; h0=512;
var i,h,x,y,x0,y0: integer;
   pf: file of integer; {plotfile}
procedure A(i: integer);
begin if i>0 then
  begin D(i-1); x:=x-h; plot;
        A(i-1); y:=y-h; plot;
        A(i-1); x:=x+h; plot;
        B(i-1);
  end
end;
{B, C and D similar for rotated instances}
begin startplot;
   i:=0; h:= h0; x0:=hdiv2; y0:=x0;
   repeat {plot Hilbert curce of order i}
      i:=i+1; h:= h div 2;
      x:=x0 + h div 2;
      y:=y0 + h div 2;
      setplot;
      A(i)
   until i=n
   endplot
end.
```



The US facility of PS facilitates programming of these kinds of curves. Moreover, we don't need a plotfile in PS. After executing the PASCAL code we still have to create a `.pdf` file suitable for inclusion in documents.

Lauwerier(1990) provides a BASIC program, `Peanol`, for the Hilbert curves, restricted to orders $\leqslant 5$, via the Turtle Graphics method, i.e. drawing forward, meaning the broken line is obtained by drawing the segments in increasing order 1, 2, 3, ... The above program draws also forward by backtracking, a short of LIFO, Last-In-First-Out. In `Peanol` the direction numbers are stored in an array of size 3411. Large enough for practical purposes. (The program above does not need the explicit direction numbers.)

The Peano curve, as shown at the beginning of this note, can be programmed along the same lines as for the Hilbert broken line above or via the Turtle Graphics casu quo Lindenmayer approach à la Lauwerier, which I leave as an exercise to the reader.

Application of plane-filling curves occurs in discretization of images where instead of a Cartesian grid the plane filling curve is used.

Hilbert curve in Metafont and Joseph Romanovski's (1995) PS code

```
%cgl, 1995
tracingstats:=1; proofing:=1;screenstrokes;autorounding:=0;
pickup pencircle scaled 0.2pt;
def openit = openwindow currentwindow
    from origin to (screen_rows,screen_cols) at (-40s,15s)enddef;
path p; s=10;
sz=0;p:=origin;%H_0 size and path
n=5; %Order of H-curve
for k=1 upto n:%H_1,...H_n consecutively
p:= p transformed (identity rotated 90
    reflectedabout (origin,up))--
p shifted ((-sz-1)*s,0)--
p shifted ((-sz-1)*s,(-sz-1)*s)--
p transformed (identity rotated -90
   reflectedabout (origin,up) shifted (-sz*s,(-2sz-1)*s));
sz:=2sz+1;
clearit;draw p; showit;
endfor
end
```

```
...
/S{0 R rlineto currentpoint stroke moveto}def
/T{90 rotate}def
/TM{T 1 -1 scale}def
/H{TM dup    0 gt
   {1 sub
          H S
       TM H S
          H T S
-1 1 scale H 180 rotate
    1 add
   }if    TM
}def
/R 8 def 100 100 moveto 6 H pop
showpage
```

Joseph's code is a bit cryptic (but is similar to mine). He uses short names, which does not make sense, makes the code difficult to read, which is considered a bad practice. But, ... he was on the right track, also with the use of colours via PS.

Conclusion. Such a simple(?) problem yielded already a few variants, of which most suffer from bad readability, because they are not biased by a production rule, IMHO.

Appendix: Sierpiński islands

Sierpiński islands S_1... S_3 have been shown in the introduction, in overlay.

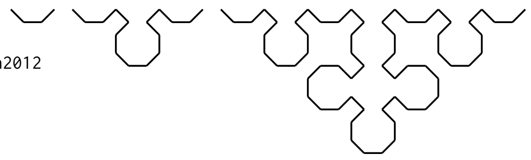
Wirth(1975), I consider a starting point for programming a Sierpiński island. Wirth did not have the rotation of US facility, nor did PS exist.

Algorithm: The curve is closed and is composed of 4 (90° rotated) broken lines connected by lines in the direction SE, SW, NW and NE. The program consists of 2 parts: first the generation of a side and second the appropriate splicing of rotated copies.

The PS program is based on the following production rule for a side

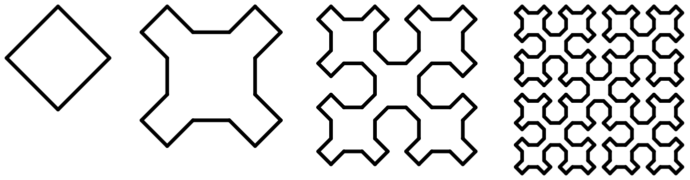
$$S_i = S_{i-1} \oplus SE \oplus S_{i-1}^{-90} \oplus E \oplus S_{i-1}^{90} \oplus NE \oplus S_{i-1}, \quad \text{for } i = 1, 2, ...$$

where $\oplus$ means spliced, the superscript denotes the rotation angle. and SE, E, NE mean draw a line in the direction South-East, East, and North-East.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski island Side, March2012
%%Author: Kees van der Laan
%% Affiliation: kisa1@xs4all.nl
%%BoundingBox:-1 -82 287 2
%%BeginSetup
%%EndSetup
%%BeginProlog
/s' s 1.4142 div def
/NE{0 0 moveto s' dup     lineto currentpoint stroke translate}def
/SE{0 0 moveto s' dup neg lineto currentpoint stroke translate}def
/E {0 0 moveto s 0        lineto currentpoint stroke translate}def
/SideS{%on stack order >=1  ==> Sierpinski side
      %s size of line segment (global)
1 sub dup 0 ge
{ dup           SideS
  SE
  dup -90 rotate SideS  90 rotate
  E
  dup  90 rotate SideS -90 rotate
  NE
  dup            SideS
}if 1 add} def
0 setlinejoin 1.415 setmiterlimit 1 setlinecap
%%EndProlog
%
%Program ---the script---
%
/s 10 def % size of segment
             1 SideS pop
s 0 translate 2 SideS pop
s 0 translate 3 SideS pop
showpage
```

Wirth programmed the four rotated instances in (recursive) procedures A, B, C and D, which entailed a more complex recursion scheme.


Sierpiński islands 0 ... 3

For those who don't own the 1975 book I have included Wirth's program, translated into Metafont in 1995.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski island, March 2012
%%Author: Kees van der Laan
%%Affiliation: kisa1@xs4all.nl
%%BoundingBox:-10 -185 307 2
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %loads /SideS and auxiliaries from PSlib
%%EndProlog
%
%Program ---the script---
%
/s 20 def % size of segment
4{                        SE -90 rotate}repeat%S_0
1.5  s mul 0 translate /s s 2 div def
4{             1 SideS pop SE -90 rotate}repeat%S_1
4.5  s mul 0 translate /s s 2 div def
4{             2 SideS pop SE -90 rotate}repeat%S_2
10.5 s mul 0 translate /s s 2 div def
4{             3 SideS pop SE -90 rotate}repeat%S_3
showpage
%%EOF
```

The S_0...S_3 pictures obtained by the given program are shown above.

```
%Translation of Wirth's code into Metafont
tracingstats:=1;proofing:=1;screenstrokes;
pickup pencircle scaled 0.01pt;
s=10; path p;
def openit = openwindow currentwindow from origin
    to (screen_rows,screen_cols) at (-5s,5s)enddef;
def A expr k=if k>0:
   A(k-1);draw z--hide(x:=x+h;y:=y-h)z;
   B(k-1);draw z--hide(x:=x+2h)z;
   D(k-1);draw z--hide(x:=x+h;y:=y+h)z;
   A(k-1); fi enddef;
def B expr k=if k>0:
   B(k-1);draw z--hide(x:=x-h;y:=y-h)z;
   C(k-1);draw z--hide(y:=y-2h)z;
   A(k-1);draw z--hide(x:=x+h;y:=y-h)z;
   B(k-1); fi enddef;
def C expr k=if k>0:
   C(k-1);draw z--hide(x:=x-h;y:=y+h)z;
   D(k-1);draw z--hide(x:=x-2h)z;
   B(k-1);draw z--hide(x:=x-h;y:=y-h)z;
   C(k-1); fi enddef;
def D expr k=if k>0:
   D(k-1);draw z--hide(x:=x+h;y:=y+h)z;
   A(k-1);draw z--hide(y:=y+2h)z;
   C(k-1);draw z--hide(x:=x-h;y:=y+h)z;
   D(k-1); fi enddef;
def sierpinski expr k=%k is order of curve
  if k>0: A(k);draw z--hide(x:=x+h;y:=y-h)z;
          B(k);draw z--hide(x:=x-h;y:=y-h)z;
          C(k);draw z--hide(x:=x-h;y:=y+h)z;
          D(k);draw z--hide(x:=x+h;y:=y+h)z;
  fi enddef;
z=origin; h=16;
sierpinski2; showit; end
```

Exam Papers Revisited
Posing Questions To Students

Abstract
Described is a module for the consistent production and maintenance of student examinations. It can typeset questions with long or short answers, yes/no questions and multiple choice. The questions are formulated as XML documents and access ConTEXt through a special interface with HTML-like syntax.

Keywords
exam, examination, collection, problem, question, multiple choice, ConTeXt, XML, HTML

Introduction
The ConTEXt `hvdm-xam` module aims at easy and consistent typesetting of student exams and maintaining collections of exam questions. It especially facilitates questions with short answers and multiple choice. Many aspects of the typesetting are configurable. The module depends on two other modules `hvdm-xml` and `hvdm-lua`. The former provides the bridge between HTML-style formatting expressions and the ConTEXt-world, the latter contains support functions that are better programmed in the programming language Lua than in TEX.

This module is a followup of the previous `hvdm-exm` module, which in turn was an upgrade to ConTEXt of an older LATEX package. After a period of relative inactivity in using my own module, it turned out that working in ConTEXt-MKII did not automatically meant working in ConTEXt-MKIV. Nothing to be surprised of, because after all ConTEXt is a fast moving target and incompatibilities are the price one has to pay for using it. Moreover, it was one of the incentives that made me move forward on the path of separating data and program code as much as possible.

In this latest overhaul of the exam producing module, the questions in the exam are no longer in TEX or ConTEXt, but in XML. A change that to a great extent decouples the problem description from the typesetting machinery. TEX code might sometimes be indispensible to achieve high quality math formulae and can be conveniently inserted. Using MathML is another option. In a sense, this article can be seen as an updated version of the earlier one published in the NTG-MAPS number 36, spring 2008.

Exam structure
Exams need to be typeset in different formats. First of course is the format in which the questions will be presented to the students taking an examination. Secondly, one can produce the same with the answers added to the questions. In my experience this is useful for scoring student results, especially when the questions are divided in several parts each meriting a reward for a correct answer. Finally one can produce a catalogue of all questions including the answers, annotations and scores. In between there is a lot of flexibility. The product can be customized through attributes on the nodes of the XML data descriptions. A small built in and simple to adapt and extend vocabulary of common language dependent terms eases localization of the product. Defining this vocabulary for a specific language allows one to typeset them automatically in ConTEXt's current active language.

The typesetting of exams is structured in three layers. The top layer is a CONTEXT file driving the typesetting. Do font setup and other such things here. A short example is given below.

```
% Load exam module
% includes [mathml][hvdm-lua][hvdm-xml]
\usemodule[hvdm-xam]
% papersize, layout, font, language, etc.
\def\prologue{..}% execute before exam
\def\epilogue{..}% execute after exam
\starttext
% Call root node <exam> in examfile.xml
\xmlprocessfile{exam}{examfile.xml}{}
\stoptext
```

The above ConTEXt program processes `examfile.xml`, which is the file harbouring the `<exam>` root. Within that root are nodes that call files for each problem. The example below clarifies the structure of this file. Nodes `<title>`, `<date>`, `<time>` and `<location>` define macros `\thetitle`, `\thedate`, `\thetime` and `\thelocation`, that can be used in the `\prologue` and `\epilogue` code figuring in the example above. How to have these called will be explained shortly. But, as can be seen in the example, it is also possible to use these nodes directly in composing a header to the exam and even have them preceded by a language localized prefix. The bonus for having date, etc. in nodes like `<date>` is the possibility to process a bunch of exams within another XML structure with direct access to these data.

```
<?xml version="1.0" encoding="UTF-8"?>
<exam attributes>
  <p align="middle">
    <title>So and So Exam</title><br/>
    <date>2099-09-09</date>
    - <time>13:00-16:00</time><br/>
    <location Pre=": ">
      Main building lounge
    </location>
  </p>
  <file name="dirAAA/question-01.xml"/>
  ...
  <file name="dirZZZ/question-99.xml"/>
</exam>
```

So and So Exam
2099-09-09 – 13:00–16:00
Location: Main building lounge
"EXAM"

There are a few commands useful in the introductory text to the exam. These are:

1. `<currentdate/>` inserts the current date in the format yyyy-mm-dd, as in 12-07-2013

2. `<date>content</date>` for setting *content* as date and at the same time remembering it. The date may either be given as yyyymmdd in which case it will be formatted as above, or in completely free format. Use the attribute `show="no"` to set the stored value without displaying.

3. `<currenttime/>` inserts the current time in the format hh:mm in a 24 hour clock, as in 11:26.

4. `<time>` the same as the date equivalent.

5. `<title>` place and setup a title for the exam.

6. `<location/>` place and setup a location for the exam.

The nodes `<date>`, `<time>`, `<title>`, `<location>` can carry an attribute `pre=""` or `Pre=""`. If present the language equivalent of the node name is typeset before the contents. The capitalization of the attribute determines the capitalization of the keyword. The value of the attribute may be empty, but if there is content then the value of the attribute will be placed before it. An example can be seen above in the location node.

Problem structure

Each problem description should get its own file, possibly in different directories. The `name` attribute of the `<file>` node locates the file, taking its origin in the directory that contains the caller. Alternatively, one may place the data in a ConTEXt buffer and call it up with a `<buffer>` node instead of a `<file>` node, as is done in the source of this article. See the heavily shortened example below.

```
\startbuffer[example]
  <?xml version="1.0" encoding="UTF-8"?>
  <problem>
    .. content of problem ..
  </problem>
\stopbuffer
\startbuffer[caller]
  <?xml version="1.0" encoding="UTF-8"?>
  <exam>
    <buffer name="example"/>
  </exam>
\stopbuffer
\xmlprocessbuffer{exam}{caller}{}
```

The problems themselves have the structure shown in the next example code. The root is a `<problem>` node and within it are nodes for specific information and one or more `<question>` nodes. If there are several questions in the problem, they can either be alternatives or they can comprise a series of questions all to be answered. Most nodes can be placed in any order. But of course the ordering matters for a series of successive questions.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem category="bachelor">
  <subject>Literature</subject>
  <description>About Shakespeare.</description>
  <history date="Earlier">First version</history>
  <history date="20120322">Was changed</history>
   <text>Text of question here.</text>
   <question score="1">
     <note>An example only.</note>
     ... <answer> nodes here ...
  </question>
  <note>Final note.</note>
</problem>
```

Buffer: example-0
Category: bachelor
Subject: Literature
Description: About Shakespeare
Maxscore: 1 point

—— *History* ——
Earlier: First version
22-03-2012: Was changed

—— *Notes* ——
1. An example only.
2. Final note.

Problem-1 Text of question here.

The purpose of the `<subject>` `<description>` and `<history>` nodes needs no further explanation. They are expected at the top of the node tree, i.e. directly inside the `<problem>` node. As can be seen, dates may be given either in free form or as `yyyymmdd`. Annotations given as `<note>` may occur everywhere, but for printing they are collected and output in one place. The `<question>` node has an attribute for the score that goes with the correct answer. Scores can either be shown or suppressed on the printout. In this article the printing of scores is suppressed, because they do not cope well with a multicolumn layout.

All information like `<subject>` etc. regarding the problem is placed inside a framed area having three sections.

Characteristic data, like the name of the data file, a short description, the subject matter, the maximum number of points allotted, etc.
The annotation part when there is at least one `<note>` present.
The version history of the problem.

Options are provided to determine what parts of this information should appear, if at all.

Question structure

There are five types of questions programmed:

1. `<shortanswer>`
2. `<altanswer>`
3. `<listanswer>`
4. `<blockanswer>`
5. `<choiceanswer>`

All will subsequently be described.

Short Answer

The first example shows the programming of a `<shortanswer>` question. Note that in this specific example the `<problem>` node carries the optional category attribute by which one can differentiate between various target groups. Specify on exam production a corresponding filter – for this example `<exam filter="bachelor">` – to filter out problems of this category and suppress all others.

The `<shortanswer>` example is presented here three times. First how it is given to students at the examination, then the same with answers filled in and finally how it will appear in a catalogue of problems. This example also demonstrates the use of consecutive questions preceded by an introductory `<text>` node. As already has been mentioned, score value are not shown but this problem will have 3 score points reported.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem category="bachelor">
  <text>Written by Shakespeare.</text>
  <question score="2">
    <text>
      What is the most famous question?
    </text>
    <shortanswer>
      To be or not to be.
    </shortanswer>
  </question>
  <question score="1">
    <text>Who said that?</text>
    <shortanswer>Hamlet.</shortanswer>
  </question>
  <note>Just an example.</note>
</problem>
```

Problem-1 Written by Shakespeare.
What is the most famous question?

Answer: ....

Who said that?

Answer: ....

Problem-1 Written by Shakespeare.
What is the most famous question?

Answer: To be or not to be.

Who said that?

Answer: Hamlet.

Buffer: example-1
Category: bachelor
Maxscore: 3 points
Alternatives: no

Note
Just an example.

Problem-1 Written by Shakespeare.
What is the most famous question?

Answer : To be or not to be.

Who said that?
Answer : Hamlet.

Alternate answers
The <altanswer> type of question is meant for simple alternatives, like yes/no questions. The example also demonstrates the use of different colors for good and the wrong answers. Understandably they show up in the answered version only.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
  <text>
    Was something rotten in the state of Denmark?
  </text>
  <question score="1">
    <altanswer>
      <item value="true">yes</item>
      <item value="false">no</item>
    </altanswer>
  </question>
</problem>
```

Problem-1 Was something rotten in the state of Denmark?
Markgood: yes / no

Problem-1 Was something rotten in the state of Denmark?
Markgood: yes / ~~no~~

List of answers
The <listanswer> is meant for a series of short questions. Each question has some small answer that should be filled in by the student.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
  <text>
    Finish the following three statements on
    quotation "To be or not to be":
  </text>
  <question score="1">
    <listanswer>
      <item>
        <text>this quote is from</text>
        <value>Hamlet</value>
      </item>
      <item> .... </item>
      <item> .... </item>
    </listanswer>
  </question>
</problem>
```

Problem-1 Finish these statements on "To be or not to be"

1. this quote is from ....
2. of the English writer ....
3. supposedly born at ....

Problem-1 Finish these statements on "To be or not to be"

1. this quote is from Hamlet
2. of the English writer Shakespeare
3. supposedly born at Stratford-on-Avon

Block answer
The <blockanswer> is meant for questions needing more space for the answer. Space can be reserved below the question or a separate answer sheet can be given. For the latter the answer block will be completely suppressed when answer and prompt are off both. That case is illustrated first, thereafter the prompt is kept and a given amount of space reserved for the answer.

Problem-1 Elaborate on the question "To be or not to be?"

Problem-1 Elaborate on the question "To be or not to be?"

Answer:

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
  <text>
    Elaborate on the question:
    "To be or not to be"?
  </text>
  <question score="1">
    <blockanswer frame="on" height="2cm">
      This famous quote is from <i>Hamlet</i>,
      a play of the English writer William
      Shakespeare. ....
    </blockanswer>
  </question>
</problem>
```

It is even possible to fill the block with a series of dotted lines. To accomplish this the height attribute on the `<blockanswer>` is set to zero and the `lines` attribute to the number of lines.

Problem-1 Elaborate on the question "To be or not to be?"

Answer: ....
....
....

Finally here follows the output for the case where `answer="on"`.

Problem-1 Elaborate on the question "To be or not to be?"

Answer: This famous quote is from *Hamlet*, a play of the English writer William Shakespeare. One really has to suspect that nowadays not many students have seen even one of Shakespeare's plays, let alone having read one. Most might not even know when or where William Shakespeare is supposed to have been born.

Multiple choice answers

Multiple choice questions are formulated in a `<choiceanswer>`. The items can be presented to the students in the given or in random order. In the first example below the order has been randomized; see the difference with the second one. The `random` attribute on the `<exam>` node governs this behaviour. Attribute `randomseed` on `<exam>` enables one to set the start of the random generator.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
  <text>
    Which person is found in <i>Hamlet</i>?
  </text>
  <question score="1">
    <choiceanswer>
      <item value="true">Ophelia</item>
      <item value="false">Rosalind</item>
      <item value="false">Desdemona</item>
      <item value="false">Juliet</item>
    </choiceanswer>
  </question>
</problem>
```

Problem-1 Which person is found in *Hamlet*?

☐ Juliet
☐ Desdemona
☐ Ophelia
☐ Rosalind

Problem-1 Which person is found in *Hamlet*?

☐ Rosalind
☐ Juliet
☐ Desdemona
☒ Ophelia

Other examples

Here follow some examples showing various possibilities of the package. A more systematic description of the nodes involved is presented after this section.

Define and use of mathml

This example shows how a MathML coded formula can be defined at the `<problem>` level, then used twice in the subsequent code.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
```

```
  <define name="parabola">
  <math xmlns='http://www.w3c.org/mathml'
      version='2.0'>
    <apply><eq/>
      <ci>y</ci>
      <apply><plus/>
        <apply><power/>
          <ci>ax</ci><cn>2</cn>
        </apply>
        <ci>bx</ci><ci>c</ci>
     </apply></apply></math>
  </define>
  <question score="1">
    <text>
      How is this function called?<br/>
      <define name="parabola"/>
    </text>
    <shortanswer>
      <define name="parabola"/> is a parabola
    </shortanswer>
  </question>
</problem>
```

Problem-1 How is this function called?
$y = ax^2 + bx + c$

Answer: $y = ax^2 + bx + c$ is a parabola

Messages

Messages are typeset as XML nodes and can serve as reminders. To name two examples: (1) as a note for the corrector that special attention is needed here, (2) as a reminder that the problem is still under construction. Messages are not shown in case answers are suppressed, because the answer flag is checked and must be true in order to get the message through. Thus by default messages are absent from exams to be handed out, but visible on the correction sheet. Another flag can takeover the role of guardian, for example: `<message flag="series"..>` will typeset the message when the series flag has been set true. Messages can be colored and given another tag, as can be seen in the example.

Problem-1 Which person is found in *Hamlet*?

☒ Ophelia

TODO: 3 answers missing

The following example shows how an error is presented. Here caused by the absence of `<item>` nodes in the multiple choice.

Problem-1 Which person is found in *Hamlet*?
ERROR <choiceanswer> no items (input source = example-9)

Random selection

Multiple choice questions have their items shuffled at random by setting the `random` attribute on the `<exam>` to `on`. Other decisions can be randomized too. For example, a question can have two variants where a random choice is appropriate. The `<random>` node does just that: choosing at random from the nodes directly inside. But what if a second, related question in the same problem must be synchronized with that choice? In that case the last random choice can be stored and called up later on. The next example, albeit a little contrived, illustrates the principle. A first `<random save="yes">` generates the random selection and saves its value, the following `<random reuse="yes">` uses this value again. In more intricate cases, one can even remember several random choices by replacing the `yes` with a name.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
  <random save="yes">
  <text>First text</text>
  <text>Second text</text>
  <text>Third text</text>
  </random>
  <blockanswer>
    <random reuse="yes">
    <text>First text again</text>
    <text>Second text again</text>
    <text>Third text again</text>
    </random>
  </blockanswer>
</problem>
```

Problem-1 Third text

Answer: Third text again

Hide and show

The following example illustrates how parts can be selectively shown and hidden. The `<show>` and `<hide>` nodes take the name of a flag (default: `answer`) as an attribute triggering appearance and disappearance, respectively. In the example the flag is the otherwise unused `user` flag. A `<show>` node will present its content when the `flag=on` and suppresses it for `flag=off`. In contrast `<hide>` hides for `flag=on` and shows for `flag=off`.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
  <blockanswer user="on">
    <hide flag="user">
      <text>&lt;hide&gt;: flag=on</text>
    </hide>
    <show flag="user">
      <text>&lt;show&gt;: flag=on</text>
    </show>
  </blockanswer>
  <blockanswer user="off">
    <hide flag="user">
      <text>&lt;hide&gt;: flag=off</text>
    </hide>
    <show flag="user">
      <text>&lt;show&gt;: flag=off</text>
    </show>
  </blockanswer>
</problem>
```

Problem-1

Answer: <show>: flag=on

Answer: <hide>: flag=off

Nodes at the <exam> level

This and the next section describe the attributes that various nodes can have. The presentation takes the form `<node attributes>`, where `attributes` is a list of one or more attribute-value pairs.

`<exam attributes>`

The `<exam>` node has quite a number of attributes. Many of these provide an opportunity to customize the type, specific content and formatting style of the product. If the attribute has a default, that one is presented with the name of the attribute.

1. `init="exam"`
   This attribute determines the product type. Also switches on by default some flags. Attribute values are:
   - `exam` examination paper without answers
   - `answer` examination paper with answers
   - `series` catalogue of questions in all variants, with answers and full data

2. `base="./"`
   This attribute states the directory from where the search for included problem and other data files begins. By default the home of the calling tex-file is taken as point of departure. Macro `\currentdirectory` will be defined to the value of the attribute. Note that the value must end with a directory separator `/`. Besides affecting file reading and include processing, it is also doing a setup for the figures directory with
   ```
   \setupexternalfigures[
     location=global,
     directory={\currentdirectory}]
   ```

3. `prologue="" epilogue=""`
   Code executed before and after processing of the problem files. The value of these attributes should be the name of a macro. For example, to call macro `\startup` on the prologue one has to use `<exam prologue="startup">`. Empty by default.
   - `prologue=""` If present executes its value before processing the content.
   - `epilogue=""` If present executes its value after processing the content.

4. `randomseed="0"`
   Seeding the random generator. Give it a value different from 0 (the default) as the start of the random series generator.

5. `filter="all"`
   When this attribute is not empty, `<problem>`'s having the same `category` are allowed to pass while others are rejected. The `<problem>` of the first example in this paper has a `category="bachelor"`, which means that `<exam filter="bachelor">` will select that one.

6. `<flag="off">`
   Here the attribute `flag` stands for one of the flag names defined below. These flags are provided in order to influence type and content of the exams produced. They assume the value `on` or `off`. For example, set `random="on"` to turn on the random ordering of multiple choice questions. For convenience `true`/`false` and `yes`/`no` act as synonyms for `on`/`off`. Flags are initialized to `off`. However, depending on the `init` attribute some are turned on by default.
   - `init="exam"` on:
     `prompt`, `random`, `score`
   - `init="answer"` on:
     `prompt`, `random`, `answer`, `subscore`
   - `init="series"` on:
     `prompt`, `answer`, `subscore`, `info`, `history`, `note`

   This is the complete list of flags:
   - `answer`

Show answers.
- `break`
  Generate linebreak after prompt.
- `frame`
  By default frame answer blocks.
- `history`
  Show history data.
- `info`
  Show information block as a whole.
- `newpage`
  Start each problem on a new page.
- `note`
  Show notes.
- `series`
  Show all problems in full.
- `source`
  Show problem source.
- `prompt`
  Precede answer space with a prompt.
- `random`
  Activate random generator and by default randomize multiple choice item order.
- `score`
  Show total score for problems; can be set to `left` for putting the score on the left or `right` for right placement (default).
- `source`
  Typeset the sourcecode of the problem.
- `subscore`
  Show (sub)score for all questions within problem; values are as for `score`.
- `user`
  Flag to be freely used.
- `verbose`
  Report activities in the logfile.

In order to provide as much flexibility as possible, the presence of flag attributes is checked and processed on entrance of the following nodes `<exam>` `<file>` `<buffer>` `<problem>` `<question>` and all the `<...answer>` nodes. Thus, if there is a reason to permanently inhibit random item order for a specific multiple choice question, one can do so by giving `<problem random="off">` on its definition. To inhibite randomization in just this specific exam, put `random="off"` on the `<file>` node calling this problem.

A flag whose value was changed on a `<file>` or anywhere within, will revert back to the value that was set on the `<exam>` node before the next file is processed. A word of advice might be in order. Use flags sparingly below the `<exam>` level, because it is all too easy to create bewildering behaviour by indiscriminately sprinkling flag changes all over the place.

7. `infostyle=""`
   Font setting for the text in the information block, empty by default. A nonempty value triggers the execution of `\switchtobodyfont[infostyle]` at the start of the information block.

8. `sourcestyle=""`
   Font setting for source code. Source is typeset verbatim, the default therefore uses the current monotype font. A nonempty value triggers the execution of `\switchtobodyfont[sourcestyle]` at the start of typesetting the source.

9. Color settings.
   - `errorcolor="black"`
     Color of error messages.
   - `messagecolor="errorcolor"`
     Color of `<message>`, default is `errorcolor`.
   - `infocolor="black"`
     Color of items in information block.
   - `goodcolor="black"`
     Color of good answers.
   - `wrongcolor="black"`
     Color of wrong answers.
   - `backgroundcolor="white"`
     Color of background in framed blocks.
   - `infobackgroundcolor="white"`
     Color of background in information block.
   - `colors="yes"`
     Use set of colors built in, black and white otherwise.

`<file attributes>`
During the processing of a file or buffer the `\currentsource` macro gives access to its name. This node has the following set of attributes.

1. `name=""`
   This attribute gives the path to the XML file containing the problem code. The path must originate in `\currentdirectory` and lead to the location of the file. Note that the origin is dependent on the `base` attribute on the `<exam>` node. The alternative `src` is also accepted.

2. `selection="1"`
   A `<select="on">` attribute on a `<problem>` node means the problem has alternative questions. For problems having alternatives this attribute provides a means to select a specific one. The attribute value 1...n determines which one will be chosen. That value must be a number greater than zero and less than or equal to the number of `<question>` nodes in the problem. By default the first of the alternatives is taken.

If this attribute is something other than a number, then the list of `<question>` nodes is searched for a matching `selection` attribute. If found, that question is chosen otherwise the first question is taken by default, An attribute of this type has precedence over a numeric one.

3. `theselector="value"`
   The attribute `theselector` denotes a selector defined inside the problem. See the description of the `<content>` node below for its usage.

`<buffer attributes>` Same as the `<file>` node but pertaining to a ConTEXt buffer. The `name` attribute or its alternative `src` gives the name of the buffer. Of course `\currentdirectory` has no meaning here.

`<page option="yes"/>` Generates a ConTEXt `\page` statement at this point. The default value is `yes`.

`<vocabulary file="" buffer=""/>` The terms appearing in the information block or for example in the answer prompt, are all localized through the value of ConTEXt's `\currentlanguage` macro. This is set by macro call `\language[..]`. In the program these terms are referred to in English and internally translated when an equivalent one is available in the current language. Currently built in are translations for English, Dutch and German. With `<vocabulary>` one can add other languages or supersede existing translations. The structure of a `<vocabulary>` can be seen in the following example. Here the standard term *file* is replaced by another Dutch (nl) equivalent: the word *bestand*. The English word must be present too, because all translations originate from that language.

```
<vocabulary>
  <word>
    <language name="en">file</language>
    <language name="nl">bestand</language>
  </word>
</vocabulary>
```

The vocabulary to add can be placed in a file or in a buffer whose name must be given as the appropriate attribute on the `<vocabulary>`. In case one needs to add or replace a few terms only, the `<vocabulary>` node can conveniently be placed in its entirety inside the `<exam>` node.

Counter values

Nodes that can come in handy, are those giving access to the counters for the number of problems and the scores.

`<counter value="problem">` sequence number of current problem.

`<counter value="problemscore">` score for the current problem.

`<counter value="totalscore">` total value of scores sofar.

`<message attributes>`
The `<message>` node without attributes has its content typeset surrounded by `<MESSAGE>` and has the same color as error messages. This node can be used to draw attention to special situations as for example an unfinished part in the problem. Its appearance can be made conditional on a specific flag. A message ends the preceding paragraph.

1. `color="messagecolor"`
   The color used for the message.

2. `text="MESSAGE"`
   The text inside the brackets appearing before and after the message content.

3. `flag=""`
   If the name of an existing flag is given here, then that flag must be true in order to typeset the message; an unknown flag suppresses the message.

Nodes at the <problem> level

`<problem attributes>`
A `<problem>` node can contain a `<subject>` and `<description>` node and any number of `<note>`, `<history>`, `<include>` and `<define>` nodes.

1. `category="all"`
   Specifies a category for this problem on which it can be filtered.

2. `select="off"`
   Having `select` set to `off` signifies a problem consisting of several parts, all to be used. Setting this flag to `on` makes this a problem with alternative questions. Then the one selected on an examination is determined by the value of the `selection` attribute on the `<file>` node. To produce a catalogue of all alternatives choose the `series` option on `<exam>`.

`<question score="0">`
This node carries the `score` attribute. It specifies

the number of points to be earned by answering the question correctly. In the information block one will see the maximum score for the total of all independent questions contained in the problem. An absent `score` attribute defaults to zero.

A problem with a number of questions all to be answered, can be typeset as an itemized list. See the example below.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem select="off">
  <text>Itemized questions:</text>
  <ol columns="2">
    <li><question>Question-1</question></li>
    <li><question>Question-2</question></li>
    <li><question>Question-3</question></li>
    <li><question>Question-4</question></li>
  </ol>
</problem>
```

Problem-1 Itemized questions:

1. Question-1
2. Question-2
3. Question-3
4. Question-4

Answer nodes

`<shortanswer attributes>`

1. `fill="dots"`
   Type of line fill after the answer prompt. Other possibilities are `none` and `line`. After definition of macro `\myfill` the use of `fill="myfill"` is allowed.
2. `lines="1"`
   The number of filled lines typeset after the answer prompt. The default is one line. It is allowed to set the number of lines to 0.

`<listanswer attributes>`

1. `break="off"`
   Set this attribute to `on` in order to break the line after the descriptive text.
2. `sym="n"`
   Set the symbol used for the items as defined by ConTEXt's `\startitemize`. Numbers are the default.
3. `fill` and `lines`: See `<shortanswer>`

The attributes `fill` and `line` are valid also on the individual items of the list, covering the case where one needs to vary them individually.

`<altanswer break="off">`

Set the attribute `break` to `on` in order to break the line after the descriptive text.

`<blockanswer attributes>`

1. In case both the `answer` and `prompt` flags are off, the block is completely suppressed thus enabling questions to be answered elsewhere.
2. `frame="off"`
   Draws a frame around the block when set to `on`.
3. `height="0pt"`
   Set the height of the block. Setting `height` to a value greater than zero forces the answerblock to take that height. Otherwise the `lines` setting is honored.
4. `width="0pt"`
   Set the width of the block.
5. `alignblock="middle"`
   Force the position of the answer block to either `left`, `middle`, `right` or `none`.
6. `align="right"`
   Force the alignment of the content of the block. Values are one of the ConTEXt options `left`, `middle`, `right`.
7. `break="off"`
   Set this attribute to value `on` in order to insert a linebreak directly after the answer prompt, whenever the `prompt` option is selected. Useful when one needs the full textwidth, for example for a multicolumn list to follow.
8. `fill` and `lines`: See `<shortanswer>`

`<choiceanswer attributes>`

1. `distance=""`
   Offset between the item marker and the text.
2. `skip="none"`
   Extra blank space between items; either a dimension or one of small, medium, etc.

Other nodes

`<text attributes>`

The `<text>` node is meant for regular text and usually typesets at least one paragraph. By default a bare `<text>` node ends with a `\par`. This seems the most natural behaviour, because the objective of this node is the placement of a block of text. However, sometimes it might be better to suppress the trailing `\par`, most notably when the content is a small fragment of text. In that case, give the `par` attribute the value `no`.

1. `par="yes"`
   A `no` will suppress the `\par` at the end of the node.

2. `align=""`
   Presence of an attribute value `left`, `middle`, `right` will place the node content inside respectively a `\leftaligned`, `\midaligned`, `\rightaligned` macro.

3. `color="black"`
   Color the whole text node.

Be aware that in XML the ampersand & and similar reserved characters must be typed as `&amp;` `&lt;` `&gt;` etc. TEX's nonbreaking space ~ must to be given as `&nbsp;`

`<content attributes>`
This node enables one to select from alternatives. These alternatives are defined by the user with a (`selector`,`value`)-pair of freely chosen names. Within each group a member should be designated as the default for that group. On a `<file>` node the attribute pair `selector="value"` will select that specific member.

1. `selector="selection"`
   The `selector` attribute designates a group of alternatives. Through this designator one chooses a member from that group. Its value becomes an attribute on the `<file>` node having as its legal values those of its member `value` attributes.

2. `value=""`
   This `value` distinguishes this alternative from the other members of its `selector` group. On the `<file>` node a specific alternative is chosen by specifying its value as the attribute value of the group selector.

3. `default="off"`
   Setting the `default` attribute to `on` makes this one the default member within its group, to be chosen whenever there is no selection made on the `<file>` node. If none of the group members has been designated as the default, then none of them will be typeset.

Thus in the following example the first `<file>` will typeset the default and the second will typeset the content explicitely chosen.

```
<question>
  <content
    selector="mysel"
    value="first"
    default="yes">
    ..
  </content>
  <content
    selector="mysel"
    value="second">
    ..
  </content>
</question>
```

```
<file name=".."/>
<file name=".." mysel="second"/>
```

Inclusion and exclusion of code

`<include file=""/>`
Instead of repeatedly copying the same code into several problems, one can define these common parts separately (`includes.xml` in the example below) in a file. This file must contain a `<root>` node in which the definitions of the common code parts are put. The particular name of this node is not significant, it is merely a container for the `<define>` nodes inside. Each of the definitions contained should be given a name whereby they can be called up for insertion in the problem. This inclusion mechanism greatly enhances the maintainability of the problem database, because common code corrections and alterations can be concentrated in one place. Loading the file with definitions is done with the `<include>` node carrying the path of the file to read.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <define name="mypart">
    .. code to include ..
  </define>
</root>
<?xml version="1.0" encoding="UTF-8"?>
<exam>
  <include file="includes.xml"/>
  .. other nodes ..
</exam>
```

It is imperative that `<include>` nodes are placed directly inside the `<exam>` node, otherwise they are ignored. In case the use of a specific definition is local to a single problem, one has the option of putting the definition directly inside the `<problem>` itself.

`<define name="">`
A `<define>` node carrying content is a definition node. These nodes may be placed either in a separate file to be included in the `<exam>` node, in the `<exam>` node itself, or put directly inside the `<problem>` node. Definitions in the `<problem>` have priority above those in the `<exam>`. Those in `<include>` files come last, later inclusions having priority above those loaded earlier. The first match of a definition breaks off the search.

A `<define name="name"/>` node not carrying content implies retrieval of the definition. Thus the presence or absence of the node's content determines its role. In the following example code the definition `mypart` is put inside the `<problem>` and afterwards substituted for `<define name="mypart"/>`.

```
<problem>
  <define name="mypart">
    .. code of mypart ..
  </define>
  ..
  <text><define name="mypart"/></text>
</problem>
```

The `<define>` can have a `type` attribute when it is a definition node. In that case when called some special action depending on its value is taken. The preprogrammed actions are:

1. `type="image"` When the `<define>` resolves to the location of a file, this is put into the document with an `\externalfigure` call.

2. `type="mpgraphic"` The `<define>` must resolve to code for a graphic. For example a definition can be

   ```
   <define type="mpgraphic" name="example"
     parameters="color=black,variant=0">
   \startuseMPgraphic{example}{color,variant}
   if \MPvar{variant} = 0:
     draw (0,0) -- (10,10)
       withcolor \MPcolor{color};
   else:
     draw (0,0) .. (10,10);
   fi
   </define>
   ```

   then called with `<define name="example" parameters="color=orange"/>`. Note that the name on the `<define>` and MPgraphic definition must be the same. Other attributes on these nodes are `height`, `width`, `scale` and `rotation`, their usage speaks for itself.

For more details and parameter usage see the paper on the `hvdm-xml` module.

Beware-1. The inclusion of `<question>` nodes inside a `<define>` is explicitely forbidden and raises an error, because it interferes with the intended processing of questions within `<problem>`. The radical solution chosen here is to delete the offending `<define>` from the nodetree. As a consequence its usage later in the input will produce another error message, enabling one to pinpoint the cause.

Beware-2. The inclusion of a `<define>` is not the same as macro substitution in a programming language as for instance C. As a result its replacement is not always bringing what is expected and its usage is somewhat limited. However, for low level replacements in `<text>` and `<value>` nodes or inclusion as list items it works reasonably well. Complicated formulas in MathML are good candidates too. On higher level nodes its usage turned out somewhat erratic and is therefore not supported everywhere. Feel free to experiment and don't be too disappointed if it doesn't work as you hoped for. After all, it is nothing more than a convenient extra.

Beware-3. The inclusion through XML can have consequences for special characters. Be especially on guard for trouble with `&`, a character that may need replacement by `&amp;` more proper for XML data.

`<hide flag="answer">`
The content of the `<hide>` node will appear depending on the value of its `flag` attribute, by default the `answer` flag. For value `on` of the targeted flag the content of the `<hide>` node will be suppressed. Otherwise the content will appear. This node is especially useful in case one wants to put something inside a `<blockanswer>` having its `<force>` flag set. Then for example `<hide flag="prompt">` allows suppression of content based on the value of the `prompt` flag.

`<show flag="answer">`
The alter ego of `<hide>`. When the flag targeted by `<show>` is on then the content of the `<show>` will be shown, otherwise it will be suppressed. Using a `<show>` and `<hide>` pair one can have alternating content depending on the attribute value of the flag. It is therefore possible to have one text for `answer=off` and a different one for `answer=on`.

`<random save="" reuse="">`
Chooses at random one of the nodes directly under it. Does nothing if there aren't nodes inside and always takes the first node if random selection is off. Intervening `<!-- comment -->`'s are ignored.

Attribute `save="yes"` enables the user to reuse the last value acquired by `<random>`. That last value is internally stored and can be reused (nondestructively) with `<random reuse="yes">`.

For more elaborate constructions it is possible to tie the random value to a chosen identification. For example, the value of the random generator produced with the call `<random save="mysave">` is reused with `<random reuse="mysave">`. Mistyping the identification to a nonexistent save generates an error, as is to be expected. Furthermore the strings `yes`, `no`, `on`, `off`, `true`, `false` cannot be used as identifier.

I wish to thank Martin Althoff for stimulating discussions contributing greatly to the rapid finalization of the main features of both the module code and this article.

Hans van der Meer
H.vanderMeer@uva.nl

A bit of HTML and a bit of ConTeXt

HTML constructs translated to ConTEXt

Abstract
Described is a module for the typesetting of a subset of HTML operators. These can be used to build data sets in XML with HTML as formatting elements and have them typeset in ConTEXt. Other features are the inclusion of predefined content and provision for language localized words and expressions.

Keywords
ConTeXt, HTML, XML, include, vocabulary

Introduction

Let us start with a tiny example, just to get the idea. For the sake of exposition the content of this example is enclosed as if it were HTML source. That is not necessary though, any root node can contain these elements. For it to work one has to load the module with `\usemodule[hvdm-xml]` first. It in turn depends on modules `[hvdm-ctx]` (a few general macros) and `[hvdm-lua]` (some Lua coded functions). By the way, note the mandatory XML-notation for the characters < and > and of course the same will apply to the ampersand `&amp;` as these are XML-related (call it) features. Also note that the example code below can be rendered in a browser unchanged.

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
  <body bgcolor="bbccbb"/>
  <title>Ignored head</title>
</head>
<body>
  Example: &lt;body&gt;
  <hr/> <!-- comment: rule -->
  <rm>roman</rm>
  <i>italic</i>
  <b>bold</b>
</body>
</html>
```

Example: <body>

---

roman *italic* **bold**

Below follows a description of the API of this module, here and there illustrated with an example. Many of the nodes are standard to HTML but there are some extras, primarily because the author needed them for his work. The idea behind the development of the module was to stay close to HTML and use its vocabulary where possible.

Document Structure

These are nodes for structuring your XML document.

- ☐ `<html>`
  The well known enclosure of all HTML.
- ☐ `<head>`
  The header part of the document. Its content is ignored because it is not supposed to play a role in the typesetting process.
- ☐ `<body>`
  The body part of the document. The content of this node can contain all the typeset material.
- ☐ `<h1 align="" color="" style=""`…`<h1>`
  Typeset headers in diminishing size according to the digit behind the ‘h’. The alignment can be `left`, `center`, `middle` or `right`, the color of the heading as well as its style can be specified. The possibilities in size are `<h1>` …`<h6>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <h2 align="center" color="blue"
      style="bold">
    heading text
  </h2>
</root>
```

heading text

Following are nodes for paragraphs and linebreaks.

- ☐ `<br/>`
  Break the line.
- ☐ `<p height="" color="" align="" style="">`
  Encloses a paragraph, although `<p/>` does equally well as a simple paragraph break. Extra white can be placed above the paragraph with the `height` attribute. The content can be colored , aligned and given a specific style, all with attributes.
- ☐ `<div>`
  Just a division as in HTML.

TEX directly

These are nodes for those parts of the document where one has to resort to using TEX directly. One such area is the typesetting of math, allthough it is possible to use MathML here. However, I find that a tedious business and for quick results am using TEX's math directly. MathML translation can come later, when things have to be tidied up.

- ☐ `<tex>`
  Contains pure TEX or ConTEXt.
- ☐ `<m>`
  Encloses the node content in pair of $'s.
- ☐ `<M>`
  Encloses the node content in pair of $$'s.
- ☐ `<nohyphens>`
  Stop hyphenation and put the content inside a `\begingroup`-`\endgroup` pair, as might be expected.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  Formula inline: <m>y=a^2+bx+c</m>
<div/>
  Formula displayed: <M>y=a^2+bx+c</M>
</root>
```

Formula inline: $y = a^2 + bx + c$
Formula displayed:
$$y = a^2 + bx + c$$

Appearance

These are nodes for making font, style and color changes.

- ☐ `<font family="" size="" style="">`
  Does font setup `\setupbodyfont[family]`, `\switchtobodyfont[size]` and/or a simple `style` change, where appropriate. The presence of a `family` attribute triggers a `\setupbodyfont` command, a `size` is effected with a `\switchtobodyfont`. The attribute for the size can be `big`, `small`, etc. A style can be something like `normal` which is translated into `\tf`; similarly for other variants, always under the assumption that the chosen variant is available.

  The list of style changes is:
  — normal = `\tf`
  — mono = `\tt`
  — bold = `\bf`
  — italic = `\it`
  — bolditalic, italicbold = `\bi`
  — slant, slanted = `\sl`
  — boldslanted, slantedbold = `\bs`
  — smallcaps = `\sc`
  — mediaeval = `\os`

- ☐ `<small step="" min="">` or `<smaller>`
  Typeset content smaller by the step size (usually in points) but enforce a minimum size given in the `min` attribute.
- ☐ `<big step="">` or `<bigger>`
  Typeset content bigger by the step size (usually in points).
- ☐ `<color value="">`
  Sets content in the color given in the attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
Before
<font family="euler" size="big">
  Fontswitch
</font>
 After
</root>
```

Before – Fontswitch – After

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<smaller step="2pt">
  smaller
</smaller>
normal
<big step="2pt">
  bigger
</big>
</root>
```

smaller normal bigger

The common HTML style nodes are present.

□ `<r>`
Typeset content in `\tf`.
□ `<u space="yes/no">`
Typeset content underlined, the space attributed governs the breaking of the underlining at word boundaries.
□ `<i>`
Typeset content in `\it`.
□ `<b>`
Typeset content in `\bf`.
□ `<em>`
Typeset content emphasized as with `\em`.
□ `<sl>`
Typeset content in `\sl`.
□ `<tt>`
Typeset content in `\tt`.
□ `<code>`
Typeset as in the corresponding HTML.
□ `<pre>`
Typeset as in the corresponding HTML.

Special operations on text.

□ `<sub>`
Typeset content in subscript.
□ `<sup>`
Typeset content in superscript.
□ `<lohi>`
Typeset content both in subscript and in superscript. The node should contain a `<sub>` and a `<sup>` node, in any order.
□ `<quote>` and `<q>`
These enclose their content in single and double quotes, respectively.

Space and Alignment
These are nodes for structuring the typesetting with horizontal and vertical space.

□ `<spacer size="" type="">`
By default the kind of spacing depends on whether the typesetting is currently in horizontal or in vertical mode. But its kind can be explicitely specified by setting the attribute `type` to either `horizontal` or `vertical`. A third type `fill` is evaluated to `\hfill` or to `\vfill` respectively. By default the `size` values are `1em` and `\normalbaselineskip` for the horizontal and vertical displacements. The size may also be given as a percentage, which is applied to `\textwidth` or `\textheight`.
□ `<center>`
Typeset centered.
□ `<left>`
Typeset aligned to the left.
□ `<right>`
Typeset aligned to the right.
□ `<narrow left="" right="" middle="" option="">`
Narrowed paragraph, the `option` attribute determines the narrowing as in the command `\startnarrower`, middle is default.
The other attributes determine the value of the narrowing and evaluate to calling command `\setupnarrower`.
□ `<blockquote>`
The HTML blockquote behaviour.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<narrow middle="1cm">
  abc
  <spacer type="fill"/>
  abc
  <spacer type="fill"/>
  abc
  <p/>
  <spacer type="vertical" size="1cm"/>
  abc
</narrow>
</root>
```

abc abc abc

abc

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<center>
  Centered
</center>
</root>
```

Centered

Lists
These are nodes for enumerations, mimicking those in HTML.

□ `<li>`
List element.
□ `<ol sym="" start="" columns="" option="">`
Ordered list with attribute `sym` specifying the kind of numbering: `n`, `a`, `A`, `r`, `R`, `g`, `G` for numbers,

letters, roman numerals or Greek letters.
The `start` attribute determines the number of the first item, by default 1 as expected. The `columns` attribute specifies the number of columns (1 by default).
Attribute `option` functions as sort of catch all, because its content is transferred to the first `[]`-argument of the typesetting `\startitemize` macro.

□ `<ul sym="" columns="" option="">`
Unordered list with the symbol having a great number of variants, switched by a number from 1 to 14 and ranging from bullets to triangles and lozenges (provided the font has them).

□ `<dl compact="yes/no">`
As in HTML, encloses `<dt>` and `<dd>` elements.

□ `<dt>`
As in HTML.

□ `<dd>`
As in HTML.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<ul sym="3" columns="2">
  <li>first item</li>
  <li>second item</li>
  <li>third item</li>
  <li>fourth item</li>
</ul>
</root>
```

first item	third item
second item	fourth item

Tables

These are nodes for tables, mimicking those in HTML and translated to XML-table macros of ConTEXt.

□ `<table foregroundstyle="" offset="" columndistance="" spaceinbetween="" location="" option="">`
The setup attributes implemented here are `foregroundstyle` corresponding to HTML's style, `offset` corresponding to cellpadding, `columndistance` to cellspacing, `spaceinbetween` to rowspacing and `location` for placement `top`, `left` or `middle`. The catch all `option` attribute can be used to set `frameoffset`, `backgroundoffset`, `textwidth`, `textheight`, `leftmargindistance`, `rightmargindistance`.

□ `<thead>`
As in HTML.

□ `<tfoot>`
As in HTML.

□ `<tbody>`
As in HTML.

□ `<tr>`
As in HTML.

□ `<td>`
As in HTML.

□ `<th>`
As in HTML.

Images and graphics

These are nodes for typesetting images and pictures.

□ `<framed many-attributes>`
Frame the contents of the node. There are a great many of attributes here, corresponding with the most important ones from `\setupframed`. To name them without further explanation: `height`, `width`, `offset`, `color`, `bgcolor`, `frame`, `framecolor`, `framecorner`, `frameradius`, `rulethickness`, `strut`, `align`, `valign`, `corner`, `radius`, `bgcorner`, `bgradius`, `option`. The last one `option` can be anything and its value is passed on to the underlying setup call.
Frame parts may be specified with HTML-like attributes for the `frame` attribute and has values

— `on,t,above,vsides` is `topframe=on`
— `on,b,below,vsides` is `bottomframe=on`
— `on,l,lhs,hsides` is `leftframe=on`
— `on,r,rhs,hsides` is `rightframe=on`

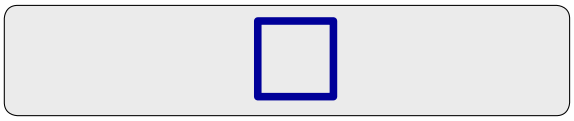
□ `<img>`
Place an image file through `\externalfigure`. Separate attributes are `height`, `width`, `scale`, `frame`, `corner`, `radius`, `rotation`, `option`. They are passed on in the second `[]`-argument to `\externalfigure`.

□ `<mpgraphic name="" [file="" buffer=""] parameters="">`
Execute METAPOST on an MPgraphic definition and place this in the document. The node must contain a `\startuseMPgraphic` `\stopuseMPgraphic` pair.
Other sources for the definition of the graphic may be specified with the `file` and `buffer` attribute.
The attribute `parameters` is used to transfer variable values via the `\MPvar` macro to the METAPOST code. Note that the name on the `\startuseMPgraphic` must be the same as the one on the `<mpgraphic>`.

Example where the graphic is defined in a separate ConTEXt buffer.

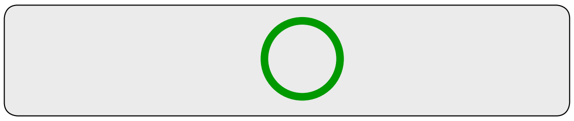
```
\startbuffer[graphic]
\startuseMPgraphic{graphic1}{color}
  pickup pencircle scaled 1mm;
  draw unitsquare scaled 1cm
      withcolor \MPvar{color};
\stopuseMPgraphic
\stopbuffer
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<center>
  <mpgraphic buffer="graphic" name="graphic1"
        parameters="color=darkblue"/>
</center>
</root>
```



Example where the graphic is defined in the <mpgraphic> node itself.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<center>
  <mpgraphic name="graphic2"
        parameters="color=darkgreen">
    \startuseMPgraphic{graphic2}{color}
      pickup pencircle scaled 1mm;
      draw unitcircle scaled 1cm
          withcolor \MPvar{color};
    \stopuseMPgraphic
  </mpgraphic>
</center>
</root>
```



Include and definition nodes

Instead of repeatedly copying the same code over and over, one can define these common parts separately (includes.xml in the example below) in a file. This file must contain a root node in which the definitions of the common code parts are put. The particular name of this node is not significant, it is merely a container for the <define> nodes inside. In the example below <includes> was chosen.

Each of the definitions contained in the file should be given a name whereby it can be called up for insertion. Loading the file with definitions is done with the <include> node carrying the path of the file to read.

```
<?xml version="1.0" encoding="UTF-8"?>
<includes>
  <define name="mypart">
    .. code to include ..
  </define>
  .. other definitions ..
</includes>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <include file="includes.xml"/>
  .. other nodes ..
</root>
```

A <define> node carrying content is a *definition* node. If more than one <include> file is loaded, then later inclusions have priority above the earlier ones. The first match of a definition breaks off the search.

A <define name="name"/> node not carrying content implies *retrieval* of the definition. Thus the presence or absence of the node's content determines its role. In the following example code the definition mypart is substituted for <define name="mypart"/>.

```
<includes>
  <define name="mypart">
    .. code of mypart ..
  </define>
</includes>
<text><define name="mypart"/></text>
```

The <define> can have a type attribute if it is a definition node. In that case some special action depending on its value is taken when called up. The preprogrammed actions are:

1. type="image"
   When the <define>'s content resolves to the location of a file, this is put into the document as an \externalfigure.
2. type="mpgraphic"
   The <define> must resolve to code for a graphic.

For example a definition can be

```
<define type="mpgraphic" name="example"
    parameters="color=green,variant=0">
\startuseMPgraphic{example}{color,variant}
  if \MPvar{variant} = 0:
    draw (0,0) -- (10,10)
```

```
      withcolor \MPvar{color};
  else:
    draw (0,0) .. (10,10);
  fi
\stopuseMPgraphic
</define>
```

We then call this definition with

```
<define name="example"
    parameters="color=orange"/>
```

On the call the defined default color green is replaced by orange, while the variant keeps its default of 0. Note that the names on the `<define>` and the graphic definition must be the same.

Other attributes provided for are `height` and `width` or `scale` for its dimensions, the former having priority above the latter. The dimensions might be given of a percentage, which is taken as a fraction of `\textheight` and `\textwidth` respectively. The `rotation` attribute specifies a rotation of the figure given in degrees.

Beware. The inclusion of a `<define>` is not the same as macro substitution in a programming language as for instance C. Complicated formulas in MathML are good candidates for a definition.

Vocabulary

It is not uncommon to program in English for all terms that can appear in the typeset document. However, it would be nice if instead these could be customized and typeset in the target language selected with the ConTEXt macro `\language[]`. The `<vocabulary>` node can help to enlighten this task. With it one defines for a chosen series of words the equivalent in each of several languages. The next example defines the English word *file*, the Dutch *bestand* and the German *Datei* as equivalents. Note the use of the two letter language designators.

```
<vocabulary>
  <word>
    <language name="en">file</language>
    <language name="nl">bestand</language>
    <language name="de">Datei</language>
  </word>
</vocabulary>
```

One of the languages will function as the *reference language* in which the program specifies the terms (English by default). The macro calls `\translate{file}` and `\Translate{file}` are typeset as *bestand* and *Bestand* for either the 'nl' or the 'de' current language. The same is accomplished with the nodes `<word>` and `<Word>`.

Further customization is the possibility to specify another language as the reference language than the default English. Give the reference language as its two letter value on the vocabulary attribute:
`<vocabulary referencelanguage="">`

The vocabulary can be placed in a file or in a buffer whose name must be given as attribute:
`<vocabulary file="" buffer="">`

Convenience Nodes

Not HTML but probably useful now and then. A save-restore, current time and date macros are provided for.

☐ `<currendate/>`
  Typesets the current date in European style.
☐ `<currenttime/>`
  Typesets the current time in a 24-hour clock.
☐ `<store name="">`
  Store content of the node under `name`.
☐ `<restore name="">`
  Call up content stored under `name`.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
store then
<store name="test">
  stored text
</store>
restore:
<restore name="test"/>
<div/>
current date: <currentdate/>
<div/>
current time: <currenttime/>
</root>
```

store then restore: stored text
current date: 12-07-2013
current time: 11:27

Hans van der Meer
H.vanderMeer@uva.nl

Yet Another Table

Ancient table macro reworked and extended for ConTEXt

Abstract
Described is a module for the typesetting of tables. The module resembles the LATEX `tabular` environment but is in fact based on a much older package, the origins of which are lost to the author.

Keywords
ConTeXt, table

Introduction

My ConTEXt `hvdm-tbx` module for tables has been long in production. It originated when the author began to feel somewhat uncomfortable with the LATEX `tabular` environment for tables. Nothing bad said about LATEX, just a personal feeling. Developing a private solution for tables seemed a nice way to make a module that is easy to use from my point of view and at the same time offered a chance to implement everything I felt necessary and/or convenient. In order to avoid clashes with the macro naming of other packages, the calling name has been made `tablex` instead of just simply `table`. Look at the trailing 'x' as something 'extra'. As is said in the abstract, the first ideas about how to typeset a table and the earliest lines of code came from an old macro package. So old in fact, that I have sought in vain to find it either among my old backups or on the internet. But an hommage to the unknown author of that package may not be lacking here.

Table structure

The contents of a table must be enclosed in macros `\starttablex` and `\stoptablex`. The first argument of the opening macro denotes the column structure, the second is optional and can be used to specify tablewide settings. The cells in the first column of the example are aligned left, those in the second column are centered and those in the third are put to the right. The optional argument in this example specifies the text style. The last row ends with `\nr`. It will be explained later that this ends the row without placing a visible separator. Separating the columns is done with `&` and `|`. The whole table is enclosed in a frame.

```
\usemodule[hvdm-tbx]
\starttablex[lcr][style=italic]
  \mulcols[3]{tablex}\cr
  1 & 2 & 3 \nr
  cell-1 | cell-2 | cell-3\nr
\stoptablex
```

tablex		
1	*2*	*3*
cell-1	*cell-2*	*cell-3*

After typesetting a table, its dimensions become available in the three macros `\tablexwidth`, `\tablexheight` and `\tablexdepth`.
In the example above these are:

width=89.34pt
height=34.99997pt
depth=0.0pt.

Having access to these values is useful when for example the width of a table is needed for refined placement.

Two types of cells

The table cells can be of two kinds. The first kind is cells where the width of the columns is dynamic, i.e. the widest cell determines the width of its column. Cells with fixed dimensions are the second kind of cell provided by the module. In this case the cell dimensions will be specified in the optional second argument of `\starttablex`.

In both cases it is the first argument where the alignment of the cells in the corresponding column is specified. The possibilities differ for the two kinds of cells. First the three usual cases of left, right and centered horizontal alignment with no variation in the vertical direction. These are for columns with dynamic width.

- `l` = horizontal left, vertical at baseline
- `c` = horizontal centered, vertical at baseline
- `r` = horizontal right, vertical at baseline

An example of text placement for all three of the above options follows. In the table on the left the standard `\strut` is placed in each cell after the text, on the right there is no strut at all. It is possible to suppress these cellbound struts in favour of the one strut that is normally placed at the end of each row. But this may have irregular baselines in adjacent cells as a consequence, depending on the exact cell content. See later on for a demonstration of this effect.

```
\starttablex[lcr][strut=yes]
  left | center | right\cr
  a | b | q\nr
\stoptablex
\quad
\starttablex[lcr][strut=no]
  left | center | right\cr
  a | b | q\nr
\stoptablex
```

left	center	right
a	b	q

left	center	right
a	b	q

When the cell dimensions have fixed values, the following alignments can be chosen.

- □ 1 = horizontal left, vertical at top
- □ 2 = horizontal centered, vertical at top
- □ 3 = horizontal right, vertical at top
- □ 4 or L = horizontal left, vertical centered
- □ 5 or C = horizontal centered, vertical centered
- □ 6 or R = horizontal right, vertical centered
- □ 7 = horizontal left, vertical at bottom
- □ 8 = horizontal centered, vertical at bottom
- □ 9 = horizontal right, vertical at bottom

The width and height of the cell can be separately specified, its depth is always set to zero. The nine columns in the next example show all the possibilities for the placement of cell contents.

```
\starttablex[123456789]
  [cellwidth=2em,cellheight=8ex,strut=no]
  a | b | q | a | b | q | a | b | q\nr
\stoptablex
```

a	b	q	a	b	q	a	b	q

The placement of contents within cells is a tricky business for cells of fixed dimensions. The different heights caused by the ascender of the b and descender of the q play havoc with the baselines when forced to the top of the cell (options 1, 2 and 3). The baselines in the three cells on the left do not line up as one should expect. On the right the wobbling of the baselines which is caused by the descender of the q, has been neutralized by setting the depth of its containing box to zero. But of course that letter now descends below the frame. In the middle things are looking better.

Various strategies can be used to solve the problem demonstrated above, but there is no silver bullet. Here the following behaviour is chosen. All contents have the depth of its natural box set to zero. Then the vertical position varies as follows.

- □ Options 1, 2 and 3 have the top of the natural box placed directly against the top of the cell with a `\vfil` at the bottom.
- □ Options 3, 4 and 5 have the natural box placed between `\vfil`'s.
- □ Options 7, 8 and 9 have the baseline of the natural box placed directly against the bottom of the cell with a `\vfil` above.

One may dislike that strategy, but it works out well when all contents have the same height and no depth. See the next example.

```
\starttablex[123456789]
  [cellwidth=2em,cellheight=8ex,strut=no]
  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9\nr
\stoptablex
```

1	2	3	4	5	6	7	8	9

The solution to the first example is to use a strut adjacent to the item in each cell. This can be effectuated by turning on the `strut` parameter, as is the default behaviour. A much more pleasing table results.

```
\starttablex[123456789]
  [cellwidth=2em,cellheight=8ex,strut=yes]
  a | b | q | a | b | q | a | b | q\nr
\stoptablex
```

a	b	q	a	b	q	a	b	q

If that is not enough, an inset is put around the items in the cell by specifying the `celloffset` parameter. It can be given a specific value or most conveniently the depth of the strut in use.

```
\starttablex[123456789]
  [cellwidth=2em,cellheight=8ex,
  strut=yes,celloffset=strut]
  a | b | q | a | b | q | a | b | q\nr
\stoptablex
```

a	b	q	a	b	q	a	b	q

The following example demonstrates that a missing `cellheight` parameter defaults to a square cell. The `cellwidth` better be present, because it defaults to zero.

```
\starttablex[123456789]
  [cellwidth=2em,
  strut=no,celloffset=2pt]
  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9\nr
\stoptablex
```

1	2	3	4	5	6	7	8	9

Note that it is possible to mix within a table columns with fixed and dynamic widths. The cells with dynamic width will then have their contents put on the same baseline as those of variants 7–9.

```
\starttablex[13c79]
  [cellwidth=2em,strut=no]
  1 | 3 | c | 7 | 9\cr
  1 | 3 | cccgccc | 7 | 9\nr
\stoptablex
```

1	3	c	7	9
1	3	cccgccc	7	9

In the above example the effect of a dynamic column between otherwise fixed cell columns is demonstrated. On close examination its is apparent that the descender on the letter g has an effect on the height of the cells in its row: they acquired a somewhat larger height.

It is even possible to specify the width of individual columns. This is done by affixing the column number to the `cellwidth` parameter. In the following example the second and third columns have been given a different width from the standard.

```
\starttablex[5555][cellwidth=2em,
  cellwidth2=4em,cellwidth3=5em]
  1   | 2   | 3   | 4   \cr
  2em | 4em | 5em | 2em \nr
\stoptablex
```

1	2	3	4
2em	4em	5em	2em

The height of rows can be adapted on the fly. Do this preferably directly behind a separator. With `\rowheight[dimension]` the size of one row is changed. The next row will revert to the previous value. For a permanent change use `\Rowheight [dimension]`.

```
\starttablex[CCC]
  [cellwidth=2cm,cellheight=6mm]
  6mm|6mm|6mm\cr\rowheight[3mm]
  3mm|3mm|3mm\cr
  6mm|6mm|6mm\cr\Rowheight[3mm]
  3mm|3mm|3mm\cr
  3mm|3mm|3mm\nr
\stoptablex
```

6mm	6mm	6mm
3mm	3mm	3mm
6mm	6mm	6mm
3mm	3mm	3mm
3mm	3mm	3mm

Placement

Tables are just hboxes and thus can be put besides one another or aligned left or right or centered on the line. The optional parameter `align` governs the alignment. Setting `align = left`, `middle` (default) and `right` puts them into the corresponding alignment, value none leaves the hbox containing the table as is. The following examples show the effect of the first three options.

left

middle

right

The vertical placement can be changed with the `location` parameter, having values `top`, `middle` (default) and `bottom`. A feature needed to place tables of varying height on a line, lined up as required. The first two tables in the next example are aligned in the `middle`, the third and fourth table are typeset with `location = top/bottom`.

middle

middle
middle

top

bottom

Apart from the placement of the table as a whole, there is the placement of contents in the cells. The first controlling item in a column is the alignment selector of the template. But one can force another alignment for individual cells. In the example below the natural alignment of the template row is changed to left, center and right in the subsequent rows. This is accomplished by enclosing the contents of the cells in `\l`, `\c`, `\r`, respectively.

:	l-template	c-template	r-template
natural :	left	center	right
\l :	left	center	right
\c :	left	center	right
\r :	left	center	right

Often the contents of the cells is simple and short. But now and then more information must be placed which doesn't fit on a short line. In that case one can put it in the cell inside a vertical box. Placement is effected with the macros `\v` and `\t` for respectively a `\vbox` and a `\vtop`. The next example demonstrates the different vertical line up for `\vbox` and `\vtop` placement.

```
\starttablex[ccc]
  vbox|vbox|vtop\cr
  \v{\hbox{verticalvertical}\hbox{stuff}}|
  \v{\hbox{vertical}\hbox{stuff}
    \hbox{vertical}\hbox{stuff}}|
  \t{\hbox{vertical}\hbox{stuff}}\cr
  vbox|vbox|vtop\nr
\stoptablex
```

vbox	vbox	vtop
verticalvertical stuff	vertical stuff vertical stuff	vertical stuff
vbox	vbox	vtop

These macros are of limited use in the case of fixed cell dimensions. Here the alignment is not perfect, because the recalculation of row height has no effect on the placement within cells earlier in the row. The example clearly demonstrates this: the contents of the first cell of the middle row is not vertically centered, in contrast to that of the third one. Furthermore the width of the first column does not adapt to the width of the box and its contents protrude on the right .

```
\starttablex[CCC][cellwidth=15mm,
  cellheight=5mm]
  < same as previous example >
```

vbox	vbox	vbox	vtop
verticalvertical stuff	vertical stuff vertical stuff	vertical stuff	vertical stuff
vbox	vbox	vbox	vtop

Frames

The module provides for three tailormade tableframe options. By default the frame has the type indicator `alternative` set to the value `normal`. It is a table with or without an outside frame. That outside frame is switched on and off by the parameter `frame`, having values `on/off` with `yes/no` as synonyms. The two other predefined types are the `open` and `opendouble` variants, chosen with parameter `alternative=open` and `alternative=opendouble`, respectively. The appearance of the outer frame in these two is obtained using the `before=` and `after=` optional parameters.

on	
1	2
3	4

off	
1	2
3	4

open	
1	2
3	4

opendouble			
1	2	3	4
5	6	7	8

A somewhat different fourth variant is the `\shortstack`. I must admit, a LaTeXism, for a table of one column. It has three arguments of which the first two are optional. The first argument gives the alignment of the cells, the default is `c`. The second argument is put into the optional argument section of the table, after shutting the outer frame off. The third argument is the contents of the table. For convenience the `\nr` row separator can be typed as `\\`. This is done with the `command` parameter through which this meaning is assigned to the macro `\\`.

```
\shortstack[l][style=mono]
{long\\longer\\longest}
```

```
long
longer
longest
```

Rows

Rows can be separated by drawn rules: thin, thick or of a given thickness, and by invisible separators. Some of the separators place the strut currently in force at the end of the row, others don't – see below for who does what. The corresponding macros are:

- `\cr` rule of standard thickness (default value is `\linewidth`) and strut.
- `\CR` does as `\cr` with rule of greater thickness (default is `2\linewidth`).
- `\crule[dimension]` does as `\cr` with rule of specified thickness.
- `\crr`, `\CRR`, `\nrule[dimension]` same as the above, no strut placed at the end of the row; but take notice of the fact that because of the `strut=always` setting all cells could have received a strut anyway.
- `\nr`, `\nrr` typeset a rule with zero height, respectively with and without placing a strut.
- parameter `rulethickness=dimension` sets the thickness of the `\cr` and the `\nr` separator.
- parameter `framethickness=dimension` sets the thickness of the surrounding frame and the `\CR` separator.

`\cr`
`\CR`
`\crule[1mm]`
`\nrr`
`\nr`

Further variations are the placement of a gap between two rows. The first variant `\rowsep` keeps the surrounding frame intact. The alternative is `\rowskip` which generates a break in the frame.

A `\cr` after the skip places a visible separator rule, a following `\nr` an invisible one. Note the use of the separators `\crr` and `\nrr` instead of `\cr` and `\nr` in places where the strut spanning up a table row must be suppressed.

first row
rule before and after `\rowsep` `\cr\rowsep[1mm]\crr`
rule before and after `\rowskip` `\cr\rowskip[1mm]\crr`
rule before not after `\rowskip` `\cr\rowskip[1mm]\nrr`
last row

It is possible to place partial rules of various heights. The general macro is `\ccrule[#][height]`. Specialized variants are `\crrule`, `\CRrule` and `\nrrule` on which the height argument is implicit. Their function is readily understood by comparison with the rule names above. Position these macros as if they were items in a cell, i.e. they make a pseudo row in the table. So they need to be separated with all variants of `&`, `|`, etc.

One caveat applies. Do not forget to end the row with `\nr` and do this with the previous row as well. Otherwise the rule that is typeset will interfere with the partial rules. Note how for columns 3–5 the vertical bars between the columns have been kept intact through judicious choice of the partial rules and the separators between them. Compare these with columns 5 and 6. Macro `\norule` is a convenient shortcut for `\nrrule[1]`.

```
\starttablex[cccccccc]
  1|2|3|4\db5|6|7|8\nr
  \ccrule[2][2pt]|\norule|\norule\db
    \nrrule[2]&\ccrule[2][2pt]\nr
  1|2|3|4\db 5|6|7|8\nr
\stoptablex
```

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Columns

Columns are formed by placing separators between the items in a table row. The simplest are `&` and `|`, shortcuts for `\tb` and `\vb` respectively. The first one places an invisible separator, the second a vertical bar of the width defined by parameter `rulethickness`. Around these separators horizontal whitespace is put, offsetting the contents of the table cell from the column separators. The size of the surrounding white can be set with parameter `columnsep`. In the example below the zero value in the second table forces the cell contents tight against the separators.

```
\starttablex[rl][rulethickness=2pt]
  right & left\nr
  right | left\nr
\stoptablex
\quad
\starttablex[rl]
  [rulethickness=2pt,columnsep=0pt]
  right & left\nr
  right | left\nr
\stoptablex
```

right	left
right	left

right	left
right	left

There are several separators to choose from.

- `\tb` or `&` is an invisible separator.
  In the form `\tb[separator]` a custom separator is placed.
- `\vb` or `|` is a vertical bar of size `rulethickness`.
- `\VB` is a vertical bar of size `framethickness`.
- `\vbsep[dimension]` is a vertical bar of given size.
- `\db` is a double vertical bar of size `rulethickness` separated by whitespace `rulesep`.
- `\DB` is like `db` but has size `framethickness`.
- `\dbsep[dimension]` is a double vertical bar of given size.

The example below has the separator which is used named at its left. Note that between the last two columns in the second row the `\tb` separator has been used to place a custom separator, an equals sign in this case.

```
\starttablex[cccccc]
  [rulesep=2pt,framethickness=2pt]
  1\vb 2\VB 3\db 4\DB 5 &6\nr
  vb\vb VB\VB db\db DB\DB
  $3\times4$\tb[=] $12$\nr
\stoptablex
```

1	2	3	4	5	6
vb	VB	db	DB	3×4 =	12

A second equivalent series of separators has variable whitespace before and after it.

- `\tbs[size][before][after]` is an invisible separator with the given amounts of whitespace before and after, a missing `after` defaults to the same space as is put before.
- `\vbs` is the equivalent of `\vb`.
- `\VBS` is the equivalent of `\VB`.
- `\vbssep` is the equivalent of `\vbsep`.
- `\dbs` is the equivalent of `\db`
- `\DBS` is the equivalent of `\DB`.
- `\dbssep` is the equivalent of `\dbsep`.

Skipping and combining columns

One can combine columns and skip columns. Skipping is provided by two macros. A specified number of columns is skipped with `\skipcols[#]`. The remaining columns in a row may be skipped all at once with `\skipall`. An example follows.

```
\starttablex[ccccccccc]
1|2|3|4|5|6|7|8|9\cr
1|\skipcols[2]|3|4|\skipcols[3]|9\cr
1|2|\skipall\cr
1|2|3|4|5|6|7|8|9\nr
\stoptablex
```

1	2	3	4	5	6	7	8	9
1			3	4				9
1	2							
1	2	3	4	5	6	7	8	9

Combining columns is done with the `mulcols` macros. There are five variants, all combine the number of columns given in the argument. They do not adapt the total width of the columns spanned. Therefore a long text may stick out of the available space.

- `\cmulcols[#]` or `\mulcols[#]` has its contents centered.
- `\lmulcols[#]` contents forced to the left.
- `\rmulcols[#]` contents forced to the right.
- `\tmulcols[#]` contents horizontally centered, in case of fixed cell dimensions vertically forced to the top.
- `\bmulcols[#]` contents horizontally centered, in case of fixed cell dimensions vertically forced to the bottom.

no cellwidth				
five				
five and much much too long				
1	three			5
five left				
five right				
top				
bottom				
1	2	3	4	5

cellwidth=5mm				
five				
five and much much too long				
1	three			5
five left				
five right				
top				
bottom				
1	2	3	4	5

A variant that does adapt to the width of its contents is `\Mulcols`. Note that the extra space goes into the rightmost column, which may be an unwanted side effect.

Mulcols				
five and not too long anymore				
1	2	3	4	5

Struts

For the selection of struts the module presents three possibilities.

- □ Use the standard `\strut` of the font in use when typesetting starts. On `\starttablex` select this option with `strut=yes` or `strut=on`.
- □ Do not use a strut, i.e. make a null strut. Select this option with `strut=no` or `strut=off`.
- □ Define the strut yourself. To this end the values of optional parameters `strutheight` and `strutdepth` are used; they default to value 0pt. Select this one with `strut=tablex`.

By default a strut is placed in every cell, but this can be disabled for a given row from a specific cell onwards. Do this by calling the macro `\notablestrut` in the first cell of the row. At the start of the next row the strut is automatically reenabled with `\dotablestrut`. This way to disable the struts also disables the strut that is placed at the end of the row. In the example a large strut is used and disabled at the start of the second row. Use this mechanism when a specific row must be compressed in the vertical dimension.

```
\starttablex[ccccc][strut=tablex,
  strutheight=3mm,strutdepth=3mm]
  1|2|3|4|5\cr
  \notablestrut 1|2|3|4|5\cr
  1|2|3|4|5\nr
\stoptablex
```

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

A second customization of strut placement is given by the parameter `cellstrut`. Value `always` for this parameter places a strut in every cell, value `never` disables these struts, but keeps the strut at the end of each row alive. Its effect is demonstrated in the irregular baselines in the table on the right.

```
\starttablex[321][cellwidth=6mm,
  cellstrut=always]
  q & t & b\nr
\stoptablex
\quad
\starttablex[321][cellwidth=6mm,
  cellstrut=never]
  q & t & b\nr
\stoptablex
```

q	t	b

q	t	b

In the case of fixed cell dimensions the value `strutdepth` will be used as celloffset with the optional parameter setting `celloffset=strut`.

Font changes

Unless set on a call to `\setuptablex`, the font used for typesetting the table is the current font. Among the optional parameters one finds the `font` parameter which may be given any value that is acceptable to `\switchtobodyfont`. If this parameter is present, it is the first to be applied.

Secondly the parameter `style` is used. It provides many alternatives, for example to change the style to italic, use a smaller font, etcetera.

Finally a parameter `fontchange[dimension]` is applied. This parameter is exclusively used for changing the size of the font. A positive dimension increases the size by that amount, a negative one diminishes it.

Colors

Coloring the text is simple. Just set the optional parameter `foregroundcolor` or simply `color` to the color required. Of course the general ConTEXt-macro `\color[]{}` can be used to change the color at will. When the color name has been made into a macro, changing the color of the whole cell is even simpler; as can be seen in the third cell.

```
\starttablex[cc][color=blue]
  blue\vb a \color[red]{red} rose\cr
  \orange orange\vb blue\nr
\stoptablex
```

blue	a red rose
orange	blue

Separators between rows and within rows get the color of the outer frame by default, i.e. the value `framecolor`.

```
\starttablex[cccc][framecolor=red]
  a\vb b\db c\vbsep[2mm]d\cr
  e\vb f\db g\vbsep[2mm]h\nr
\stoptablex
```

a	b	c	d
e	f	g	h

But all column and row separators can be colored differently by appending a color between brackets behind them. In the example below the outer frame, the separators between the rows and those between the columns all have been given a different color. Colored horizontal rules stay within the outer frame.

```
\starttablex[cccc][framecolor=darkgreen]
  a \VB[blue] b \DB[blue]
  c \vbsep[2mm][blue] d
  \crule[1mm][cyan]
  e \VB[red] f\DB[red]
  g \vbsep[2mm][red] h\nr
\stoptablex
```

a	b	c	d
e	f	g	h

Backgrounds may be colored when the parameter `background` has been set to the value `color`, as is the default. The optional parameter `backgroundcolor` or `bgcolor` governs the general background of the table and is white by default. Individual cells can be given a background color by placing macro `\bg[color]` somewhere in the first cell to receive individual background coloring. That color remains in effect until either another color is set or an empty `\bg` is encountered. Set the parameter `background` to `none` or `off` in order to disable background coloring regardless of the color macros present. There is one restriction: coloring the background of individual cells is implemented for cells with fixed dimensions only.

```
\starttablex[CCC]
  [bgcolor=yellow,cellwidth=2em]
  a\vb\bg[green] b\vb\bg[cyan] c\cr
  \bg d\vb e\bg[orange]\vb\white f\nr
\stoptablex
```

a	b	c
d	e	f

Variables and setup

All parameters that can be given a value in the option argument of `\starttablex` can also be given an initial value with the macro

`\setuptablex[..=..]`

Next follows a list of variables occurring in the table typesetting. Be warned, changing them on the fly can easily work havoc on the table, because the underlying `\halign` may behave wildly when not treated mildly. Variables having a name starting with `\tablex` are internal variables. The others are settable on `\setuptablex` and `\starttable`. Note that for these a parameter named `param` ends up in macro `\tablexparam`. Be aware of the fact that changes to macros and registers inside a cell have a local effect only, it is the way `\halign` works. Lasting changes require the application of `\global` changes.

- □ `tabsep` Glue normally placed at the start and end of a row and around separators.
- □ `location` Vertical placement of the table.
  values: `none`, `top`, `middle` (default), `bottom`.
- □ `align` Horizontal placement of the table.
  values: `none`, `left`, `middle` (default), `right`.
- □ `alternative` Table style framed or open type.
  values: `framed` (default), `open`, `opendouble`.
- □ `before` Code to execute just before the table is typeset.
- □ `after` Code to execute just after the table is typeset.
- □ `strut` Select a strut.
  values: `yes`, `on` for the standard `\strut` of the font (default); `table` the given `strutheight` and `strutdepth`; `no`, `off` no strut.
- □ `cellstrut` governs a strut in every cell for value `always` (default) or `never`.
- □ `strutheight` Height custom strut, default 0pt.
- □ `strutdepth` Depth custom strut, default 0pt.
- □ `rulethickness` Size of standard horizontal and vertical separators.
- □ `rulesep` Distance between bars in double separator.
- □ `columnsep` Glue around separators.
- □ `cellwidth` Width of fixed dimension cells.
  `cellwidth#` Width of column number #.
- □ `cellheight` Height of fixed dimension cells, defaults to `cellwidth`.
- □ `celloffset` Inset for fixed dimension cells.
  values: dimension (default, initial 0pt) or `strut` for the value of `strutdepth`.
- □ `color` Color of text, defaults to `black`; may also be given as `foregroundcolor`.
- □ `framecolor` Color of outside frame and separators, defaults to `black`.
- □ `background` Enable background coloring.
  values: `color`, `yes`, `on` (default), `none`, `no`, `off`.
- □ `bgcolor` Color of general background, defaults to `white`; may also be given as `backgroundcolor`.
- □ `font` General font for text. If not empty executes contents within `\switchtobodyfont[]`, defaults to empty, applied first.
- □ `style` Change style of general text font. Applied after `font` and before `fontchange`.
  values: `bold`, `italic`, `bolditalic`, `italicbold`, `slant`, `slanted`, `boldslanted`, `slantedbold`, `smallcaps`, `oldstyle`, `mediaeval`, `normal`, `serif`, `regular`, `roman`, `sans`, `sansserif`, `mono`, `type`, `teletype`, `handwritten`, `calligraphic`, `big`, `small`, `smallmono`, `tiny`, `tinymono`

- □ `fontchange` Change size of font with the amount given, defaults to empty, applied last.
- □ `command` Gets a value that is defined as macro `\\`, the default is `\cr`.
- □ `\tablexwidth`, `\tablexheight`, `\tablexdepth` Width, height and depth of the table just typeset.
- □ `\tablexcellwdt`, `\tablexcellhgt` Dimen registers for width and height of the current cell in case of fixed dimensions.
- □ `\tablexcolumns` Count register for number of columns in template.
- □ `\tablexcolumn` Count register current column.
- □ `\ifinfirstcol` Tells when in first column.
- □ `\ifdimensionsfixed` Tells if fixed dimensions.
- □ `\gettablexcellwidth{number}` Resolves to width of given column when fixed dimensions.

Hans van der Meer
H.vanderMeer@uva.nl

Making the ConTEXt wiki easier to improve

Abstract
An effort is underway to encourage both reading and editing of the ConTeXt wiki. This article names nine concrete improvements that are part of this effort, and makes a case for each of them. These nine items are the following. To impose structure and to ease navigation: predictable article names; navboxes; and a simple Main Page. To coordinate efforts: a 'How this wiki works' page; a village pump; and templates for flagging problems. To make things easy for our editors: templates for common things; template documentation; sandboxes and testcases for templates.

Keywords
documentation, projects, wiki, infrastructure, ConTeXt

Introduction

The ConTEXt wiki, eight years old already, is still not what it should be: people on the mailing list often say that they checked it and came away disappointed or confused, or even that they have stopped checking it at all. Not many people edit the wiki, either: the lack of structure makes it hard to know where to put things, and the tools to instruct and coordinate are also lacking.

MAPS articles usually involve TeX code, but this one does not — this article is about the current effort to make the ConTeXt wiki an editor-friendly and reader-friendly place. It is not about what we want, but about what we are doing: the specific things we are implementing on the wiki to bring it back to health. This article will hopefully be useful to other projects setting up their own wikis, and inspiring to ConTEXt users who want to get involved in shaping the wiki.

A very short introduction for those with other wikis: a **wiki** is a collection of articles that anybody can edit, like Wikipedia. The idea is that the barrier to helping out is as low as humanly possible. Linking to other articles is done by typing `[[Article title]]`; this is called a **wikilink**. A **template** is a page containing text that can be reused on many other pages; you transclude the template `X` by typing `{{X}}`. Templates can take **template parameters**: on the ConTEXt wiki, for example, a link to the reference page of `\starttext` is produced with `{{cmd|starttext}}`.

Nine things every wiki should have

To impose structure and to ease navigation:
1. Predictable article names
2. Navboxes
3. A simple Main Page

To coordinate efforts:
4. A 'How this wiki works' page
5. A Village Pump
6. Templates for flagging problems

To make things easy for our editors:
7. Templates for common things
8. Template documentation
9. Sandboxes and testcases for templates

Structure and navigation: Predictable article names

Being able to guess article names is good for users: they can go straight to where they need to be, they know what to expect when they get there, and it gives them a pleasant feeling of knowing where things are.

Being able to guess article names is even better for editors. This section could also be called 'A place for everything, and everything in its place': knowing the proper page names is the same as knowing where to put things. Especially while a wiki is not yet mature it is vital that editors can instantly add information to the right page, and it is vital that they know when to start a new article and what to call it.

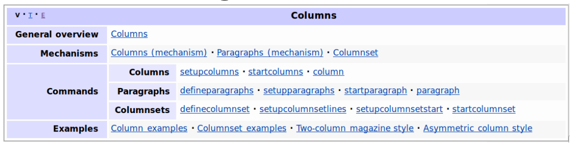
What if the article titles on a wiki are not systematic, and it is not always clear what goes where, like on the ConTEXt wiki? If a user knows something and wants to add it to the wiki, here are some things that could happen — these are all things I have done myself.

- ☐ They could find themselves investing five minutes in assessing what article pages already exist on the topic, which is time they should not need to invest.
- ☐ They might have to start a new page and fret for a while on what to call it, because the seven similar pages employ four different naming conventions.

- ☐ They could just put their contribution at the end of an existing article and be done with it, thus making that article even more of a hodge-podge than it already was.
- ☐ They might remember the effort they had to make the last few times, and end up not making the edit after all.

For the ConTEXt wiki, we will soon have a discussion on what article name patterns we need. On the topic of columns, for example, we will need to describe both the 'columns' mechanism and the 'columnset' mechanism, compare them, document their commands, have a place to put examples, and list relevant example documents. For an example of how to call the various articles, see the navbox in the next section.

Structure and navigation: Navboxes



V · T · E	Columns	
General overview	Columns	
Mechanisms	Columns (mechanism) · Paragraphs (mechanism) · Columnset	
Commands	Columns	setupcolumns · startcolumns · column
	Paragraphs	defineparagraphs · setupparagraphs · startparagraph · paragraph
	Columnsets	definecolumnset · setupcolumnsetlines · setupcolumnsetstart · startcolumnset
Examples	Column examples · Columnset examples · Two-column magazine style · Asymmetric column style	

This is a navbox, or navigation box[1]. It is a fantastically good way to (1) provide an overview of articles available on a topic; (2) highlight gaps and double entries; (3) highlight inconsistent article naming; (4) instantly make a new article reachable from many other pages. If there is anything more useful to structure a wiki, I don't know what it is. Every wiki should have a navbox template, and I don't understand why MediaWiki doesn't include one in the main distribution.

Structure and navigation: A simple Main Page

When it comes to the Main Page, project wikis are different from Wikipedia[2]. Especially when the wiki is the website of the (open source) project, as well as its documentation: Wikipedia's main page fires a hundred impressions at you at once, but project websites should not. This is a guess at the most common reasons for visiting an open source project's wiki/website:

- ☐ To learn about the program or project in general
- ☐ To look up an answer to a usage question
- ☐ To join the project's development community
- ☐ To learn about improving the wiki

A simple and attractive main page gives readers confidence that they will find their answer. The ConTEXt wiki's main page's current eighty-odd links inspire fear of getting lost, instead.

Coordinating efforts: A 'How this wiki works' page

People can't do what they don't know, and they can't know what they can't discover, so we need to write our guidelines down. Explain how the pieces of the wiki work, and how articles are named and structured, and how we'd like pages to look: that will make it easy to get new editors started, and it will give them confidence in their edits. This may turn into multiple pages, of course, and it probably will.

Coordinating efforts: A Village Pump

The village pump is the 'Welcome to this wiki' page for editors. It points to the FAQ. It points to the 'How this wiki works' page. It points to the pages of the various wikiprojects, and on a small wiki it may mention which one has priority. It has a place for people to ask questions, and to make proposals and discuss plans. No wiki should be without.

The project pages will be located in the same namespace as the village pump; this namespace is generally named after the wiki it is on, and used for issues concerning (parts of) the wiki in general. On Wikipedia the namespaces is `Wikipedia` and the village pump is located at `Wikipedia:Village Pump`[3]; the equivalent location on the ConTEXt wiki would be `Context wiki:Village Pump`[4].

'Village Pump' may seem a slightly odd name; it is the name of the main forum of the English-language Wikipedia. You may know the Village Pump by another name on your own language's Wikipedia project: it is called *De Kroeg* on the Dutch site, *Le Bistro* on the French site, *La Café* on the Spanish site, and *Fragen zur Wikipedia* on the German Wikipedia.

Lastly: is a village pump better than a mailing list? It is. Unlike the mailing list, it is located on the wiki. Imagine browsing the wiki and stumbling across a link to a mailing list; conversely, imagine stumbling across an actual discussion that you can instantly contribute to. Which is the more compelling? Which one is easier to contribute to, especially if one was editing the wiki anyway?

1. Navboxes on Wikipedia: `http://en.wikipedia.org/wiki/Template:navbox`
Navboxes on the ConTEXt wiki: `http://wiki.contextgarden.net/Template:navbox`
2. Main pages: `http://en.wikipedia.org/wiki/Main_Page`; `http://wiki.contextgarden.net/Main_Page`;
3. `http://en.wikipedia.org/wiki/Wikipedia:Village Pump`;
4. `http://wiki.contextgarden.net/Context_wiki:Village_Pump`;

Coordinating efforts: Templates for flagging problems

Wikis always take a while to settle in, and pages will end up needing attention that doesn't boil down to 'It's empty. Somebody should write something here'. When people come across such problems and don't have time to fix them right away, they should have a template they can slap on that will mark the problem. Templates for the following, at least, should be at hand:

- ☐ This page X should be **split** into two articles, X and Y.
- ☐ This page should be **merged** with page X.
- ☐ This page should be **moved** to page X.
- ☐ The information on this page needs to be **checked** for accuracy.
- ☐ The writing or formatting does not conform to this wiki's **style**.
- ☐ A generic TODO template that shows a description of the problem.

Sometimes the template adds a box to the page to warn or inform the reader; nearly always the template adds the page to a category like `Pages to split`. These categories make sure the problems are not forgotten, and also serve as a source of simple tasks.

Easing improvement: Templates for common things

Make typing easy. How many typing aid templates you can add for general usage depends on your project; the ConTEXt wiki has templates for linking to command pages and to source files. In addition, it has typing-aid templates that are used when documenting the wiki itself: `{{tag|b}}` is easier to write than `<code><nowiki><b>...</b></nowiki></code>`; and for linking to, for example, `Template:navbox`, `{{tl|navbox}}` is hard to beat.

These are the templates that should at least be available for documenting one's wiki, because typing their result requires mucking about with `<code>` and/or `<nowiki>` tags: `{{tag}}`[5] to produce HTML tags; `{{tlx}}`[6] to type template names with parameters; and `{{para}}`[7] to type parameters alone. A template for box templates is also good; consider stealing `{{ambox}}`[8] from Wikipedia.

Easing improvement: Template documentation

Document your wiki's templates! You really want to, especially because the MediaWiki template syntax is so hard to read. In this case, the Dutch Wikipedia's system is better than that of the English Wikipedia: `{{Sjablooninfo}}`[9] is a very short and simple template that puts its contents in a yellow box.

The English Wikipedia's template documentation system is massively over-engineered compared to the Dutch one: it involves separate `/doc` subpages, the templates `{{documentation}}` and `{{documentation subpage}}`, and close to a dozen sub-templates[10]. Unfortunately, that is the system the ConTEXt wiki now uses, because it was transferred by the author of this article before he discovered the simpler system. Changing over to the Dutch system is still very tempting, and may yet happen.

Easing improvement: Sandboxes and testcases for templates

We want our templates to be improvable, yes? Then we need a way to make that improvement risk-free. So, we place a testing version of our template X at `X/sandbox`, and have `X/testcases` display various examples of `{{X|...}}` and `{{X/sandbox|...}}` side-by-side. Proposed changes can be made and tweaked and improved in the sandbox first. Once the editor has a final version, they can move that to the template page. This way, they can be confident of getting it right the first time.

In conclusion

By the time you are reading this, the ConTeXt wiki will not be perfect yet — but it will have become easy to improve. Come and have a look, visit the village pump, and see if you like the new direction. Give thanks to Patrick Gundlach, who originally created the wiki; and to Taco, who is its sysadmin. Help out if you spot something fun, and by all means kibitz. The wiki is getting better every weeki.

Sietse Brouwer
sbbrouwer@gmail.com

5. `http://en.wikipedia.org/wiki/Template:tag`
6. `http://en.wikipedia.org/wiki/Template:tls`
7. `http://en.wikipedia.org/wiki/Template:para`
8. `http://en.wikipedia.org/wiki/Template:ambox`
9. `http://nl.wikipedia.org/wiki/Sjabloon:Sjablooninfo`
10. `http://en.wikipedia.org/wiki/Template:documentation` and `Template:documentation subpage`. For the subtemplates, edit `Template:documentation` and look under the 'Templates included on this page' heading at the bottom.

MetaPost: Numerical engines

Abstract
After years of talks about future plans for MetaPost 2.0, finally real progress is being made. This paper introduces a pre-release of MetaPost 2 that can optionally use IEEE floating point for its internal calculations instead of the traditional 32-bit integers.

Introduction

I am sure some readers are curious to know why it is taking so long before MetaPost 2 comes out, considering that I have been giving talks on the subject for years now. To get started, a recap from the initial project proposal dating back to May 2009:[1]

> In the original MetaPost library proposal we wrote in May 2007, one of the big user-side problem points that was mentioned was this:
>
> □ All number handling is based on fractions of a 32-bit integer. User input often hits one of the many boundaries that are a result of that. For instance, all numbers must be smaller than 16384, and there is a noticeable lack of precision in the intersection point calculations.
>
> The current proposal aims to resolve that issue once and for all. The goal is to replace the MetaPost internal 32-bit numeric values with something more useful, and to achieve that goal the plan is to incorporate one of these libraries:
>
> | | |
> |---|---|
> | GNU MPFR library | `http://www.mpfr.org/` |
> | IBM decNumber | `http://www.alphaworks.ibm.com/tech/decnumber` |
>
> We have not decided yet which one. MPFR will likely be faster and has a larger development base, but decNumber is more interesting from a user interface point of view because decimal calculus is generally more intuitive. For both libraries the same internal steps need to be taken, so that decision can be safely postponed until a little later in the project. The final decision will be based on a discussion to be held on the MetaPost mailing list.

Since then, there has been a small change to that statement: MetaPost 2 will in fact contain four different calculation engines at the same time:

□ scaled 32-bit (a.k.a. compatibility mode)
□ IEEE floating point (a.k.a. double)
□ MPFR (arbitrary precision, binary)
□ decNumber (arbitrary precision, decimal)

The internal structure of the program will also allow further engines to be added in the future.

The traditional scaled 32-bit engine is the default, thus retaining backward compatibility with older versions of MetaPost. The other engines will be selected using a command line switch.

Working backwards from that final goal, some sub-projects could be formulated.

> □ Because values in any numerical calculation library are always expressed as C pointers, it is necessary to move away from the current array-based

1. This article is an updated version of 'MetaPost 1.750: Numerical engines'. Published in TUGboat, Volume 32.2, pages 136–138. The current content reflect Metapost 1.803.

memory structure with overloaded members to a system using dynamic allocation (using malloc()) and named structure components everywhere, so that all internal MetaPost values can be expressed as C pointers internally.

As a bonus, this removes the last bits of static allocation code from MetaPost so that it will finally be able to use all of the available RAM.

This first sub-project was a major undertaking in itself, and was finally completed when MetaPost 1.5 was released in July 2010.

The current 1.80x release of MetaPost implements most of two other sub-project goals (in fact so far only the PostScript backend has been updated):

- □ An internal application programming interface layer will need to be added for all the internal calculation functions and the numeric parsing and serialization routines. All such functions will have to be stored in an array of function pointers, thus allowing a start-up switch between 32-bit backward-compatible calculation and the arbitrary precision library.

  As a bonus, this will make it possible to add more numerical engines in the future.
- □ The SVG and PostScript back-ends need to be updated to use double precision float values for exported points instead of the current 32-bit scaled integers.

  In the picture export API, doubles are considered to be the best common denominator because there is an acceptable range and precision and they are simple to manipulate in all C code. This way, the actual SVG and PostScript backend implementations and the Lua bindings can remain small and simple.

So, not accounting for hunting for bugs and fixing documentation, there is only one large step that remains to be taken before MetaPost 2 can be released, namely the actual integration of the two arbitrary precision libraries. That is why the version is set at 1.80x at the moment.

Some internal stuff

One thing that is not immediately obvious from the project goals as written above is that moving all the core arithmetic operations into functions that must be swappable instead of resolved at executable compilation time meant a whole lot of editing work, almost none of which could be automated. This is the main reason why everything took so long. Let me illustrate that with an example.

An example: a simple procedure

Let's look at the `trans` procedure, that applies a transform to a pair of coordinates. It calculates the following formula:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} txx & tyx \\ txy & tyy \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

First, here is the original pascal implementation of that function:

```
procedure trans(p,q:pointer);
var v:scaled; {the new |x| value}
begin
  v:=take_scaled(mem[p].sc,txx)+take_scaled(mem[q].sc,txy)+tx;
  mem[q].sc:=take_scaled(mem[p].sc,tyx)+take_scaled(mem[q].sc,tyy)+ty;
  mem[p].sc:=v;
end;
```

The meaning of all those variables:

p,q	The variables for the x and y coordinates that have to be transformed
txx, txy, tyx, tyy, tx, ty	The six components of the transformation matrix, in global variables
v	An intermediate value that is needed because p cannot be updated immediately: its old value is used in the calculation of the new q
mem[]	The statically allocated memory table where Pascal MetaPost stored all its variables
mem[].sc	The structure object that holds the scaled value of a variable
take_scaled(a,b)	This function calculates $p = \lfloor (a \cdot b)/2^{16} + \frac{1}{2} \rfloor$

Table 1. Pointless caption

In the conversion of MetaPost from Pascal web to C web (in version 1.2), not all that much has changed:

```
static void mp_trans (MP mp,pointer p, pointer q) {
  scaled v; /* the new |x| value */
  v=mp_take_scaled(mp, mp->mem[p].sc,mp->txx)+
    mp_take_scaled(mp, mp->mem[q].sc,mp->txy)+mp->tx;
  mp->mem[q].sc=mp_take_scaled(mp, mp->mem[p].sc,mp->tyx)+
    mp_take_scaled(mp, mp->mem[q].sc,mp->tyy)+mp->ty;
  mp->mem[p].sc=v;
}
```

The only big difference here is the use of a global mp object instead of global variables. MetaPost 1.5 uses dynamic allocation instead of the mem array, and that makes the function a lot easier to understand:

```
static void mp_trans (MP mp, scaled * p, scaled * q) {
  scaled v;     /* the new |x| value */
  v = mp_take_scaled (mp, *p, mp->txx) +
    mp_take_scaled (mp, *q, mp->txy) + mp->tx;
  *q = mp_take_scaled (mp, *p, mp->tyx) +
    mp_take_scaled (mp, *q, mp->tyy) + mp->ty;
  *p = v;
}
```

It would be great if that could stay, but unfortunately, when numerical variables become objects, it is no longer allowed to use the simple C + operator for addition. In turn, that means that more local variables are needed to store intermediate results. To make matters even worse, these local variables have to be allocated and released.

The end result is that the same function looks like this since MetaPost 1.750:

```
static void mp_number_trans (MP mp, mp_number p, mp_number q) {
  mp_number pp, qq;
  mp_number r1, r2;
  new_number (pp);
  new_number (qq);
  new_number (r1);
  new_number (r2);
  take_scaled (r1, p, mp->txx);
  take_scaled (r2, q, mp->txy);
  number_add (r1, r2);
  set_number_from_addition(pp, r1, mp->tx);
  take_scaled (r1, p, mp->tyx);
  take_scaled (r2, q, mp->tyy);
  number_add (r1, r2);
  set_number_from_addition(qq, r1, mp->ty);
  number_clone(p,pp);
  number_clone(q,qq);
  free_number (pp);
  free_number (qq);
  free_number (r1);
  free_number (r2);
}
```

The variables r1, r1, pp and qq exist only for storing intermediate results. To be honest, qq is not really needed, but it adds a nice bit of symmetry and the overhead is neglectable.

The new arithmetic functions do not return a value since that would force the introduction of even more new_number and free_number calls. Instead, they adjust their first argument. Stripped down to only the actual actions, the function looks like this:

```
  r1 = p * mp->txx;
  r2 = q * mp->txy;
  r1 = r1 + r2;
  pp = r1 + mp->tx;

  r1 = p * mp->tyx;
  r2 = q * mp->tyy;
  r1 = r1 + r2;
  qq = r1 + mp->ty;

  p = pp;
  q = qq;
```

Where the first four lines match the first statement in the previous versions of the function, the next four lines the second statement, and the last two lines do the final assignments.

In the listing above, all those identifiers like new_number and take_scaled are not really functions. Instead, they are C preprocessor macros with definitions like this:

```
#define take_scaled(R,A,B)  (mp->math->take_scaled)(mp,R,A,B)
```

Here the right-hand side take_scaled is one of the function fields in the structure mp->math. Each of the arithmetic engines defines a few dozen such functions each for its own type of mp_number. With this new internal structure in place adding a new arithmetic engine is not much more work than defining a few dozen – mostly very simple – functions.

Using the new MetaPost

As said, there are currently only two engines: scaled 32-bit and IEEE double. Switching to IEEE double is done on the command-line by using

```
   mpost --numbersystem=double mpman
```

It is not (yet?) possible to change the numerical engine, but to make it possible to do tests inside of macros, there is a new read-only internal variable called numbersystem that returns the current engine as a string, either scaled or double.

Warning checks

In MetaPost 1, the parameter warningcheck can be set to a positive value. This will downgrade the limit on numerical ranges from 16384 to 4096, but it has the advantage that is guards against various internal cases of overflow.

With the `double` numerical engine, numerical values can range up to 1.0E+307. The warning check could be set at something like 2.5E+306, but that is actually not the most important point for a warning to take place.

Because of the way double values are stored internally in the hardware, it is possible to store a certain range of integers *exactly*. However, when an integer value gets above a threshold (it has to fit in 52bits), precision is lost. For this reason, `warningcheck` now kicks in at this limit, which is $2^{52} \approx 4.5E{+}15$.

Input format

In `double` mode, Metapost uses an extended scanner for numerical values that accepts floating point numbers in scientific notation:

```
v := 1.23E+10;
w := -4.56e-10;
```

Both `E` and `e` are allowed (Metapost itself uses `e` when it needs to report a value) and the plus sign is optional.

Note: the extension to the value scanner means that legacy Metapost input files may fail to run correctly under the new `--numbersystem=double`. The new scanning that is done for `E` or `e` followed by a digit, plus or minus sign means that at that spot, any variable with that name will be overshadowed:

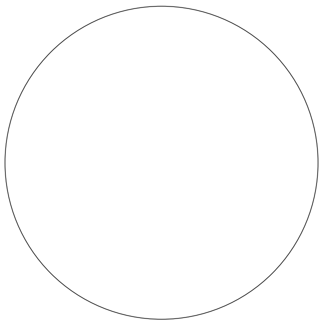
```
numeric e[];
e2 = 5;
v = 3e2;
show v;
end.
```

Normally, Metapost will show the value as `15`, but with `--numbersystem=double` it will report `300` instead. The simplest solution in cases where this happens is to add an extra `*` operator in between:

```
v = 3*e2;
```

An actual example

```
beginfig(1);
warningcheck:=0;
path p;
p = fullcircle scaled 23.45678888E-200;
p := p scaled 1E201;
draw p;
currentpicture := currentpicture scaled .5;
endfig;
end.
```



Before you try ...

- □ Some internals, like `intersectiontimes`, do not take advantage of the extra precision yet.
- □ Not all of the memory leaks have been found and dealt with.
- □ I am fairly certain that there are yet undiscovered bugs.

Taco Hoekwater

Simple Spreadsheets

Introduction

Occasionally a question pops up on the ConTEXt mailing list and answering it becomes a nice distraction from a boring task at hand. The spreadsheet module is the result of such a diversion. As with more support code in ConTEXt, this is not a replacement for 'the real thing' but just a nice feature for simple cases. The module is loaded with

```
\usemodule[spreadsheet]
```

So this is (at least currently) not one of the core functionalities but an add-on. Of course some useful extensions might appear in the future.

Spreadsheet tables

We can use Lua in each cell, because under the hood it is all Lua. There is some basic parsing applied so that we can use the usual `A..Z` variables to access cells.

```
\startspreadsheettable[test]
  \startrow
    \startcell 1.1         \stopcell
    \startcell 2.1         \stopcell
    \startcell A[1] + B[1] \stopcell
  \stoprow
  \startrow
    \startcell 2.1         \stopcell
    \startcell 2.2         \stopcell
    \startcell A[2] + B[2] \stopcell
  \stoprow
  \startrow
    \startcell A[1] + B[1] \stopcell
    \startcell A[2] + B[2] \stopcell
    \startcell A[3] + B[3] \stopcell
  \stoprow
\stopspreadsheettable
```

The rendering is shown in figure 1. Keep in mind that in Lua all calculations are done using floats, at least in Lua versions with version numbers preceding 5.3.
The last cell can also look like this:

```
\startcell
function()
  local s = 0
  for i=1,2 do
    for j=1,2 do
      s = s + dat[i][j]
    end
  end
  return s
end
\stopcell
```

1.1	2.1	3.2
2.1	2.2	4.3
3.2	4.3	7.5

Figure 1. A simple spreadsheet.

The content of a cell is either a number or a function. In this example we just loop over the (already set) cells and calculate their sum. The `dat` variable accesses the grid of cells.

```
\startcell
function()
  local s = 0
  for i=1,2 do
    for j=1,2 do
      s = s + dat[i][j]
    end
  end
  tmp.total = s
end
\stopcell
```

In this variant we store the sum in the table `tmp` which is local to the current sheet. Another table is `fnc` where we can store functions. This table is shared between all sheets. There are two predefined functions:

```
  sum(columnname,firstrow,lastrow)
  fmt(specification,n)
```

The `sum` function works top-down in columns, and roughly looks like this:

```
function sum(currentcolumn,firstrow,lastrow)
  local r = 0
  for i = firstrow, lastrow do
    r = r + cells[currentcolumn][i]
  end
  return r
end
```

The last two arguments are optional:

```
  sum(columnname,lastrow)
```

This is equivalent to:

```
function sum(currentcolumn,lastrow)
  local r = 0
  for i = 1, lastrow do
    r = r + cells[currentcolumn][i]
  end
  return r
end
```

while:

```
sum(columnname)
```

boils down to:

```
function sum(currentcolumn)
  local r = 0
  for i = 1, currentrow do
    r = r + cells[currentcolumn][i]
  end
  return r
end
```

1.1	2.1
2.1	2.2
7.5	**10.7**

Figure 2. Cells can be (complex) functions.

Let's see this in action:

```
\startspreadsheettable[test]
  \startrow
    \startcell 1.1 \stopcell \startcell 2.1 \stopcell
  \stoprow
  \startrow
    \startcell 2.1 \stopcell \startcell 2.2 \stopcell
  \stoprow
  \startrow
    \startcell
      function()
        local s = 0
        for i=1,2 do
          for j=1,2 do
            s = s + dat[i][j]
          end
        end
        context.bold(s)
      end
    \stopcell
    \startcell
      function()
        local s = 1
        for i=1,2 do
          for j=1,2 do
            s = s * dat[i][j]
          end
        end
        context.bold(fmt("@.1f",s))
      end
    \stopcell
  \stoprow
\stopspreadsheettable
```

The result is shown in figure 2. Watch the `fmt` call: we use an at sign instead of a percent to please TEX.
Keep in mind that we're typesetting and that doing complex calculations is not our main objective. A typical application of this module is in making bills, for which you can combine it with the correspondence modules. We leave that as an exercise for the reader and stick to a simple example.

```
\startspreadsheettable[test]
  \startrow
    \startcell[align=flushleft,width=8cm] "item one" \stopcell
    \startcell[align=flushright,width=3cm] @ "0.2f EUR" 3.50 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "item two" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" 8.45 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "tax 19\percent" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" 0.19 * (B[1]+B[2]) \stopcell
  \stoprow
  \startrow
```

```
    \startcell[align=flushleft] "total 1" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" sum(B,1,3) \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "total 2" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" B[1] + B[2] + B[3] \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "total 3" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" sum(B) \stopcell
  \stoprow
\stopspreadsheettable
```

Here (and in figure 3) you see a quick and more readable way to format cell content. The `@` in the template is optional, but needed in cases like this:

```
  @ "(@0.2f) EUR" 8.45
```

an `@` is only prepended when no `@` is given in the template.
In practice this table we can be less specific and let `\sum` behave more automatic. This can be simplified (see figure 4) and made a bit nicer looking.

```
\startspreadsheettable[test][frame=off]
  \startrow
    \startcell[align=flushleft,width=8cm] "The first item" \stopcell
    \startcell[align=flushright,width=3cm] @ "0.2f EUR" 3.50 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "The second item" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" 8.45 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "The third item" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" 5.90 \stopcell
  \stoprow
  \startrow[topframe=on]
    \startcell[align=flushleft] "VAT 19\percent" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" 0.19 * sum(B) \stopcell
  \stoprow
  \startrow[topframe=on]
    \startcell[align=flushleft] "\bf Grand total" \stopcell
    \startcell[align=flushright] @ "0.2f EUR" sum(B) \stopcell
  \stoprow
\stopspreadsheettable
```

item one	3.50 EUR
item two	8.45 EUR
tax 19%	2.27 EUR
total 1	14.22 EUR
total 2	14.22 EUR
total 3	42.66 EUR

Figure 3. Cells can be formatted by using `@` directives.

The first item	3.50 EUR
The second item	8.45 EUR
The third item	5.90 EUR
VAT 19%	3.39 EUR
Grand total	21.24 EUR

Figure 4. The sum function accumulates stepwise.

first	`@ "[@i]" 1`	[1]
second	`= 2`	2
third	`! 3`	
fourth	`4`	4
total one	`sum(C)`	10
total two	`= sum(C)`	20

Figure 5. Cells can be hidden by ! and can contain strings only.

There are a few more special start characters. This is demonstrated in figure 5. An = character is ignored.[1] When we start with an !, the content is not typeset. Strings can be surrounded by single or double quotes and are not really processed.

```
\startspreadsheettable[test][offset=1ex]
  \startrow
    \startcell[align=flushleft] "first" \stopcell
    \startcell[align=flushleft] '\type{@ "[@i]" 1}' \stopcell
    \startcell[align=flushright,width=3cm] @ "[@i]" 1 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "second" \stopcell
    \startcell[align=flushleft] '\type{= 2}' \stopcell
    \startcell[align=flushright] = 2 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "third" \stopcell
    \startcell[align=flushleft] '\type{! 3}' \stopcell
    \startcell[align=flushright] ! 3 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "fourth" \stopcell
    \startcell[align=flushleft] '\type{4}' \stopcell
    \startcell[align=flushright] 4 \stopcell
  \stoprow
  \startrow
    \startcell[align=flushleft] "\bf total one" \stopcell
    \startcell[align=flushleft] '\type{sum(C)}' \stopcell
    \startcell[align=flushright] sum(C) \stopcell
```

1. Taco suggested to support this because some spreadsheet programs use that character to flush a value.

```
  \stoprow
  \startrow
    \startcell[align=flushleft] ”\bf total two” \stopcell
    \startcell[align=flushleft] '\type{= sum(C)}' \stopcell
    \startcell[align=flushright] = sum(C) \stopcell
  \stoprow
\stopspreadsheettable
```

The sum function is clever enough not to include itself in the summation. Only preceding cells are taken into account, given that they represent a number.

100	test
20	7
120	37
1.57	
1.570	
12.000	

Figure 6. A sheet can be filled and accessed from regular tables.

Normal tables

In the previous examples we used TEX commands for structuring the sheet but the content of cells is Lua code. It is also possible to stick to a regular table and use specific commands to set and get cell data.

```
\bTABLE[align=middle]
  \bTR
    \bTD \getspr{100} \eTD \bTD test \setspr{30} \eTD
  \eTR
  \bTR
    \bTD \getspr{20} \eTD \bTD \getspr{4+3} \eTD
  \eTR
  \bTR
    \bTD \getspr{A[1] + A[2]} \eTD
    \bTD \getspr{B1 + B2} \eTD
  \eTR
  \bTR
    \bTD[nx=2] \bf \getspr{(A[3] + B[3]) /100} \eTD
  \eTR
  \bTR
    \bTD[nx=2] \bf \getspr{fmt(”@0.3f”,(A[3] + B[3]) /100)} \eTD
  \eTR
  \bTR
    \bTD[nx=2] \bf \getspr{fmt(”@0.3f”,(sum(A,1,2)) / 10)} \eTD
  \eTR
\eTABLE
```

The method to use depends on the complexity of the table. If there is more text than data then this method is probably more comfortable.

A few settings

It's possible to influence the rendering. The following example demonstrates this. We don't use any formatting directives.

```
\startspreadsheettable[test]
  \startrow
    \startcell   123456.78 \stopcell
  \stoprow
  \startrow
    \startcell  1234567.89 \stopcell
  \stoprow
  \startrow
    \startcell A[1] + A[2] \stopcell
  \stoprow
\stopspreadsheettable
```

123456.78
1234567.89
1358024.67

Figure 7. Formatting (large) numbers.

123,456.78
1,234,567.89
1,358,024.67

Figure 8. Formatting (large) numbers with style and color.

Figure 7 demonstrates how this gets rendered by default. However, often you want numbers to be split in parts separated by periods and commas. This can be done as follows:

```
\definehighlight[BoldAndRed]  [style=bold,color=darkred]
\definehighlight[BoldAndGreen][style=bold,color=darkgreen]

\setupspreadsheet
  [test]
  [period={\BoldAndRed{.}},
   comma={\BoldAndGreen{,}},
   split=yes]
```

The Lua end

You can also use spreadsheets from within Lua. The following example is rather straightforward:

```
\startluacode
context.startspreadsheettable { "test" }
  context.startrow()
    context.startcell() context("123456.78")   context.stopcell()
  context.stoprow()
  context.startrow()
    context.startcell() context("1234567.89")  context.stopcell()
  context.stoprow()
  context.startrow()
    context.startcell() context("A[1] + A[2]") context.stopcell()
  context.stoprow()
context.stopspreadsheettable()
\stopluacode
```

However, even more Lua-ish is the next variant:

```
\startluacode
  local set = moduledata.spreadsheets.set
  local get = moduledata.spreadsheets.get

  moduledata.spreadsheets.start("test")
    set("test",1,1,"123456.78")
    set("test",2,1,"1234567.89")
    set("test",3,1,"A[1] + A[2]")
  moduledata.spreadsheets.stop()

  context.bTABLE()
    context.bTR()
      context.bTD() context(get("test",1,1)) context.eTD()
    context.eTR()
    context.bTR()
      context.bTD() context(get("test",2,1)) context.eTD()
    context.eTR()
    context.bTR()
      context.bTD() context(get("test",3,1)) context.eTD()
    context.eTR()
  context.eTABLE()
\stopluacode
```

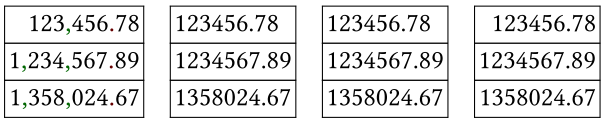
Of course the second variant does not make much sense as we can do this way more efficient by not using a spreadsheet at all:

```
\startluacode
  local A1, A2 = 123456.78, 1234567.89
  context.bTABLE()
    context.bTR()
      context.bTD() context(A1)    context.eTD()
    context.eTR()
    context.bTR()
      context.bTD() context(A2)    context.eTD()
    context.eTR()
    context.bTR()
      context.bTD() context(A1+A2) context.eTD()
    context.eTR()
  context.eTABLE()
\stopluacode
```

You can of course use format explicitly. Here we use the normal percent directives because we're in Lua, and not in TEX, where percentage signs are a bit of an issue.

```
\startluacode
  local A1, A2 = 123456.78, 1234567.89
  local options = { align = "flushright" }
  context.bTABLE()
    context.bTR()
      context.bTD(options)
        context("%0.2f",A1)
      context.eTD()
    context.eTR()
    context.bTR()
      context.bTD(options)
        context("%0.2f",A2)
      context.eTD()
    context.eTR()
    context.bTR()
      context.bTD(options)
        context("%0.2f",A1+A2)
      context.eTD()
    context.eTR()
  context.eTABLE()
\stopluacode
```

As expected and shown in figure 9, only the first and last variant get the numbers typeset nicely.



123,456.78	123456.78	123456.78	123456.78
1,234,567.89	1234567.89	1234567.89	1234567.89
1,358,024.67	1358024.67	1358024.67	1358024.67

Figure 9. Spreadsheets purely done as ConTEXt Lua Document.

Helper macros

There are two helper macros that you can use to see what is stored in a spreadsheet:

```
\inspectspreadsheet[test]
\showspreadsheet   [test]
```

The first command reports the content of `test` to the console, and the second one typesets it in the running text:

```
t={
 { 123456.78, 1234567.89, 1358024.67 },
}
```

Another helper function is `\doifelsespreadsheetcell`, You can use this one to check if a cell is set.

```
(1,1): \doifelsespreadsheetcell[test]{1}{1}{set}{unset}
(2,2): \doifelsespreadsheetcell[test]{2}{2}{set}{unset}
(9,9): \doifelsespreadsheetcell[test]{9}{9}{set}{unset}
```

This gives:

(1,1): set
(2,2): unset
(9,9): unset

There is not much more to say about this module, apart from that it is a nice example of a TEX and Lua mix. Maybe some more (basic) functionality will be added in the future but it all depends on usage.

Hans Hagen
PRAGMA ADE
Hasselt NL

5th International ConTEXt Meeting

Chateau Jekervallei

My first thought as I arrived at the Chateau Jekervallei was, “Listen to how quiet it is here”. This sense of peace and quiet persisted, and the chateau proved to be a respite from the hustle and bustle of everyday life for all of us who were there for the fifth annual ConTEXt meeting.

The chateau is located a mere fifteen kilometers south west of Maastricht on the southern edge of the Belgian village of Bessange-Bosis. The chateau lies in Wallonia, but the border between French speaking Wallonia and Dutch speaking Flandars meanders through the countryside, between, and even through villages. It is strange sight to drive through a village and see a *te koop* sign on one side of a street and a similar *à vendre* sign directly across the road.

The chateau was built in the mid 19th century for Baron de Moreaux de Melen. A large and rambling place with spacious rooms and large rectangular windows, it could comfortably accommodate twenty people, and I think we filled each and every bed.

Meals and meetings took place in a carriage house adjoining the chateau that dates from the mid 18th century. Speaking of meals, besides his many other talents, our host Jan proved to be an excellent chef. Throughout the week we enjoyed meals whose flavors delighted our palates while their names sounded sweet in our ears: ‘Cochelet (haantje) avec lentilles et purée’, ‘Poisson grillé et Paella au fruits de mer’ and ‘Lapin avec prunes et pour dessert poires au chocolat!’

Lectures

The technical portion of the program took place on Tuesday, Wednesday and Friday.

I will review three of the nearly thirty lectures in order to give you a taste of the fare that was on offer. I will not go into much detail. For a detailed description of the presentations, I recommend you lay hold of a copy of the ConTEXt Group’s first proceedings. Better yet, join the Context Group for an annual fee of 40 euros and receive the proceedings as part of your membership.

xml in ConTEXt mkiv

Thomas Schmitz gave two complementary talks. The first was an introduction to typesetting xml in ConTEXt mkiv. After remarking about xml’s simplicity, readability and generality, Thomas enumerated several advantages xml has over other formats including how xml encourages separating content and presentation, how it can be read by a myriad applications and how it can be processed to produce various outputs.

ConTEXt mkii uses an xml streaming parser, which means that elements are processed exactly in the order in which the parser reads them. ConTEXt mkiv introduced an xml DOM parser that reads an entire document into memory. You now have access to every element of your xml document at every moment. This makes reuse, selection and manipulation much easier.

One consequence of typesetting xml in ConTEXt is that you need to write a set of macros telling

ConTEXt how to associate the various xml elements in your document with their corresponding ConTEXt commands. Writing this additional layer between your source and your output requires careful work. The complexity of this layer is directly proportional to the amount of fine tuning you require. The time and effort required to write this layer is only justifiable when your documents have a predictable or repetitive structure, when this structure is translated into a repetitive look and when you want to generate several output formats from each source.

These are precisely the criteria for Thomas' lecture notes, and the subject of his second talk.

Presentations in xml with the simpleslides module

In his second talk Thomas described how, after hearing Hans' xml tutorial at the first ConTEXt user meeting, he resolved to code his lecture presentations in xml. His goal was to produce three distinct documents from a single xml file: the presentation itself, a simpler 'printable' version that students could easily download and print and a set of cards containing his notes for the presentation.

Thomas led us through an example presentation style where we progressively added code first to recognize and then to process various xml elements and their contents. Matching and processing individual elements allows you steadily and step-by-step to expand your style's breath and depth. I will not show any code here, rather I suggest you read Thomas' article in the C5M proceedings. Additional information can be found in Thomas' 'TEI xml' entry on contextgarden and in Hans' manual 'Dealing with XML in ConTeXt mkiv'.

One clever aspect of Thomas' style that must be mentioned is how he keeps multiple presentations in a single file. So how does he differentiate between them? He specifies a mode on the command-line which the presentation style compares to an attribute on the presentation elements, and processes only that presentation where the mode and attribute match.

As an added attraction on the printed version, Thomas automatically adds a word cloud on the last page. He uses a Lua program to extract the text from a presentation, and feeds it to the IBM Word Cloud Generator - a java application that creates the word cloud. A word's size in the resulting cloud is dependent on how frequently it appears in the text.

One advantage of maintaining several styles, one for each output format, is that it is easy to redefine commands and thus customize your result.

One drawback of maintaining several styles is that you tend to copy blocks of code between your various styles, and maintaining consistency across a number of separate yet similar style files quickly becomes bothersome.

If you have to process a single document producing several different forms of output, xml is for you. XML processing in ConTEXt mkiv is very cool. Thomas' two presentations at C5M are a good starting point for anyone interested in using this technology.

E-Books

Last October Julian Barns used the occasion of his Man Booker Prize acceptance speech to make an impassioned plea for the survival of printed books. Andrew Miller, last year's recipient of England's prestigious Costa Prize for Literature recently lamented that he soon expected most popular fiction would be read on screen. If you can believe the hype, printed books, along with those who collect them, are soon to be consigned to the dust bin of history; replaced by a new generation of techno savvy youngsters with e-books.

In his talk entitled 'Old Wine in New Bottles' Hans surveyed the current state-of-the-art of e-readers and e-books. Hans began by remarking that e-books are just books; which is true. Consider the results when publishers first scrambled to scan their printed books and re-market them as e-books. OCR content was poured into text files with little or no concern for structure or design.

There are essential differences though. Traditional books offer resolution, portability and longevity; while e-books offer economy, searching, indexing and reflow. We are familiar with how to read a printed book. We feel its weight and we can flick through the pages to find a favorite poem. E-books offers us a different experience. Some critics contend that by eliminating all variations in the appearance and weight of the material object in our hands, we are freed to focus on the words themselves. Whatever your persuasion, it appears that ConTEXt can readily accommodate both audiences.

ConTEXt has been producing PDF for paper and screen for quite some time. But Hans demonstrated how ConTEXt, with a slight processing overhead, can now also produce xhtml and css output suitable for an e-book.

The key is structure. Most novels and non-fiction books can be typeset with only an handful of structure elements, e.g., paragraphs, headings, emphasis, lists and images. Verbatim text and hyperlinks are a bit more problematic. Hans showed how the new mkiv structure style can facilitate both tagged PDF and xml output.

ConTEXt can also convert an e-book directly to

PDF. Publishers, unfortunately, are not consistent in how they markup their xhtml, and each 'style' has to be typeset separately. Still, ConTEXt's ability to directly process an e-book is eminently preferable to the usual double translation from html to MS Word to Adobe InDesign to print an e-book.

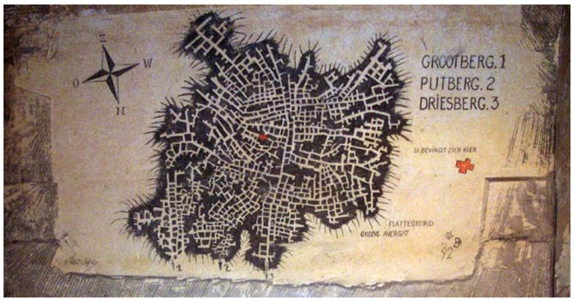
Looking to the future, Hans envisions e-books and e-reader technologies maturing and expanding to become the dominant way we consume the written word. If this happens, it would necessarily mean the demise of many a bookstore, and perhaps also the publishers that supply them. One unintended consequence of which might be a decentralization of the whole publishing industry where self-publishing becomes a viable option for both aspiring and accomplished authors.

I like to think that printed and electronic books are sufficiently different to comfortably co-exist for a long time to come. *Vive la différence!*

Excursion

Regardless of how pleasant the chateau surroundings were, after two days of lectures we were ready for a day out. We began our day's excursion with a walking tour of the nearby village of Canne. The village is split in two by the Albert canal that connects the industrial city of Liège and the port of Antwerp. While on the map you will see only one name 'Canne', the local residents refer to that part of Canne north of the Albert canal as Opcanne; and to the south Neercanne.

Mergel Caves



We began by walking from the center of Neercanne across the Albert canal to the caves of Opcanne. Like all the caves in the region, these caves are underground quarries from which mergel, a local limestone, was extracted. The tunnels of Opcanne stretch for sixteen kilometers. Signs of the labor exerted to extract so much stone are visible both in the daily block tallies written on the walls and the myriad smudge marks on the ceiling from lanterns that were placed on the top row of the tunnel face.

Interestingly, the tunnels are owned by whomever owns the ground above them. Some owners have formed cooperatives while others remain independent. Today, the spaces are used for cold storage, mushroom cultivation, even winter stalling for a local dairy herd. One enterprising individual has carved out an entire *feestzaal* replete with central heating.

One grotto that caught everyone's attention was where hundreds of bottles of Grottenbier were stored in pupitre. We were told that sugar and yeast are added to this beer so that it undergoes a second fermentation in the bottle. The beer is then stored in the grotto to mature for several months before sale. Needless to say, we stopped at a wine and spirits shop to sample the local brews.

Fort Eben-Emael

We then visited the fortress of Eben-Emael. Constructed between 1931 and 1935, it was considered the largest fort of its kind in the world, and could accommodate more than a thousand men. Its purpose was to prevent the Germans from crossing the Albert canal, which links the industries of Liège with the port of Antwerp. Unfortunately, like the *Hollandsche Waterlinie* or the French *maginot* line, these relics of the first world war proved ineffective in the new war. In the wee hours of Sunday 10 May 1940 German troops landed by glider on top of the fort, disabled its two large guns and forced the surrender of everyone inside. While the bridge over the Albert canal at Canne was destroyed, others were taken intact. With the threat of Eben-Emael removed, the Germans entered Belgium unhindered, flanked the maginot line then turned north to capture Belgium and France in a matter of days.

General George S. Patton said that, "Fixed fortifications are a monument to man's stupidity" Fort Eben-Emael remains as a testimony to its conception and construction, the bravery of its defenders, and its ultimate inadequacy in the face of modern warfare.

The photo of the main stairwell of Fort Eben-Emael courtesy of Kris Van Herzeele.

Château Neercanne

I didn't know that the Netherlands had a balcony, but it does have one. Its name is Château Neercanne. On December 9th 1991 Queen Beatrix of the Netherlands welcomed the government leaders of the member states of the European Union there to sign the Maastricht treaty.

Château Neercanne is now a restaurant boasting two Michelin stars. It is the only terraced castle in the Benelux, from which one has a panoramic view of the Jeker valley, and its baroque garden has been restored to its original glorious design.

A Renaissance Ball

Thursday evening we were whisked back in time five hundred years to partake in an evening of Renaissance music and dancing. The music was provided by Maaike Boekhout and Regina Albanez of the Dutch Renaissance/Baroque ensemble La Primavera, and our choreographer was Mari Voipio. Dressed in a period costume Mari looked every inch the Renaissance courtly lady. A handful of brave souls ventured to join the ball and learn the various sequence of steps Mari demonstrated. And regardless whether you were an aspiring dancer or a reluctant wallflower, everyone had a convivial and enjoyable evening.

ConTEXt Group

The conference was also the occasion for the first ConTEXt group meeting. Board members were confirmed and the bylaws were published. Here are the board members and the group's goals as stated in the bylaws. For the complete bylaws, please read Willi Egger's article in the conference proceedings, or read them online at: `http://group.contextgarden.net`.

Board members

President	Taco Hoekwater
Vice-president	Mojca Miklavec
E-communication	
Treasurer	Adelheid Grob
Secretary	Willi Egger
Editor Journal	John Haltiwanger
Member	Arthur Reutenauer
Member	Thomas A. Schmitz
Member	Jano Kula
Member	Peter Münster

Goals

The goals of the association are:

- ☐ To promote the typesetting system ConTEXt
- ☐ To provides its users with infrastructure and information
- ☐ To foster its development
- ☐ To provide a platform for discussing quality structured and automated typesetting
- ☐ To be a meeting place for people wanting to get different languages and writing systems typeset as well as possible

To that effect, the association may:

- ☐ Organize conferences and tutorials for ConTEXt users and developers to meet
- ☐ Provide financial support for people involved in documenting or developing ConTEXt, including, but not limited to, sponsoring attendees to conferences about ConTEXt and related topics
- ☐ Produce and sell printed material about ConTEXt
- ☐ Sell promotional items advertising ConTEXt

Michael Guravage