PSlib.eps Catalogue

—use of PostScript libraries —

Kees van der Laan

Hunzeweg 57, 9893PB Garnwerd, Gr, NL
phone: (+31/0)858776631 email: kisa1@xs4all.nl

Abstract

PostScript definitions collected in my `PSlib.eps` library and documented as e-book catalogue is presented. Now and then variant pictures have been included from `pic.dat` which comes with `Blue.tex`. Old Metafont codes have been included which may be useful for Metapost programmers. Variants of pictures enriched by postprocessing in Photoshop show other possibilities. Escher's doughnut is a teaser which has to be done in Metapost. Next to `PSlib.eps` comes the file `PDFsfromPSlib`, which contains the pictures in `.pdf` format. The complete `PSlib.eps`, `PDFsfromPSlib` as well as the catalogue as e-book, will be released, on `www.ntg.nl`, on occasion of NTG's 25th lustrum which will be celebrated in the fall of 2014. A prerelease has been offered to the GUST's file server. The (static) library for TEX alone pictures, `pic.dat`, packaged with `Blue.tex` in the 90-ies, will be redistributed as well. The Lorenz, Rössler and Hénon attractor pictures have been included. Comments, suggestions and improvements as well as contributions are welcomed: kisa1@xs4all.nl.

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

Contents

Introduction

Gutenberg organized in Toulouse in the 90-ies a meeting themed Graphics and TEX. Later, especially after Metapost had been released in the public domain, GUST paid attention to graphics. Piotr&Piotr developped `PSView`. LaTEX came with its picture environment and Knuth c.s. developed the `gkp`-macros with the picture environment functionality for use in plain TEX. `Blue.tex` appeared with its library `pic.dat` of TEX alone pictures, biased by the `gkp`-macros. On the EuroTEX92 in Prague Karel Horak showed his math pictures created by Metafont.[1] Later Timothy van Zandt released `PSTricks`, which uses PostScript as the graphics engine and LaTEX's picture environment as user interface. Herbert Voß maintains and extends `PSTricks`. PostScript was there all the time, since 1985 to be precise, but ... was apparently overlooked for graphics, except by Don Lancaster, who came with his `PSSecrets` and `Gonzo` collection.[2] PostScript was in use as (abstract) page description language for the upcoming laser printers.

In relation to TEX, we could in the mid-90-ies include `.eps` pictures in documents via the `\psfig` command, to be processed by `DVIPS`.[3] Jackowski developed `MFtoEPS`, to transform Metafont pictures into

[1] Karel works nowadays in Metapost, with `.eps` and `SVG` output.

[2] Gonzo utilities facilitate using PostScript as formatter for texts. Don does not use local dictionaries for library robustness.

[3] My papers of that time can't be processed by pdfTEX. -;(. They have to be adapted and the pictures must be converted into `.pdf`, `.jpg`, ...

PostScript for inclusion in TEX documents. Alas, pdfTEX does not allow direct inclusion of `.eps` files, a retrograde. Happily, ConTEXt not only allows inclusion of `.eps` files, but also works interactively with Metapost.[4]

Of my ≈25 years of involvement with, and using, TEX, I spent ≈5 years on developing `Blue.tex` and ≈20 years on creating and collecting pictures. Lauwerier's BASIC pictures, especially his fractals, I have been translated into PostScript. As an aside I exercised HTML for my WWW-site.
`PSlib.eps` was introduced at the EuroTEX2009. The library facilitates programming in PostScript. Hundreds of codes for the pictures are stored in a ordered way, facilitating its use. The catalogue is the accompanying documentation,[5] where next to the code the result of the code has been shown. It contains Adobe's Blue book `def`s as starting point. Much more, what I needed, has been added. The collection has been tested by example. I don't claim that it has been thoroughly tested nor is the collection robust, although the numerical concept robustness does not apply to graphics, in general, it occurs. Since EuroTEX2012 it has grown substantially. Older PostScript codes with local dictionaries have been added. For the 25^{th} lustrum meeting of NTG, in the fall of 2014, I decided to make the library more easily available, and come up with a catalogue, such that the desired code for the wanted graphics can be more easily spotted and used, hopefully. Numerici have their program libraries, we should have our catalogues for pictures, to start with Adobe's Blue book collection and my `PSlib.eps` extensions. I consider catalogues for graphics useful, because we lack a taxonomy, the world of pictures is too diverse. May the catalogue contribute to the 'literature' of PostScript graphics programming. May PostScript start a second glorious graphics life. Hang on and enjoy!

Why PSlib.eps?

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

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

My answer is

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

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

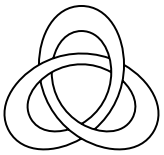
However, ... 20 years after the birth of Metapost, I still don't have a

[4] An example where TEX-Graphics interaction occurs naturally is in typesetting crosswords. But, ... a crossword is a simple graphic, so we can do with TEX alone.

[5] Physical thick, logical thin!

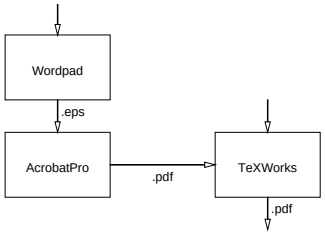
[6] Your old Acrobat Pro 7 is a treasure for converting `.eps` into `.pdf` and viewing it. Cherish Acrobat Pro 7!

pleasing Metapost working environment on my PC.[7] The TEXlive DVD does not provide me such an IDE, except as integrated part of ConTEXt. Besides, much graphics can be programmed in PostScript, as is demonstrated in my `PSlib.eps`. A notational exception is the Escher doughnut, and ilks, with its hidden lines.



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

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



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

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

Processing the graphics separately and include the resulting `.pdf`, or converted `.jpg` which is more efficient in TEXworks, into your AnyTEX(, or Word, or ...) document, adheres the Separation of Concerns principle. A wider audience than the TEXcommunity, such as PostScript users, users of Word, ..., is served. `PSTricks` based packages look nice and easy, but they suffer intrinsically from the deficiency of LaTEX's picture environment, as 'graphical language,' such as the limited orientation of lines. But, ... the advantage is that the user is hidden from PostScript, does not have to know PostScript.

Why PostScript?

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

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

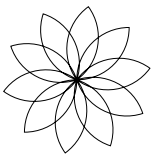
[7] Troy Henderson has his Metapost viewer, which can be accessed and used remotely. It is just a pity that he has not released a local, strand-alone version.

[8] Operators given in Lancaster's PSSecrets are hard to read and yielded non-working results. For example his Archimedian spiral, where he did not use polar coordinates, is unnecessarily complicated and did not work. On the other hand his Fonts for Free are a gem.

then to generate commands that invoke those higher-level operations. The LRM 3 is the defining document for the syntax and semantics of the language, the imaging model, and the effects of the graphics operators.

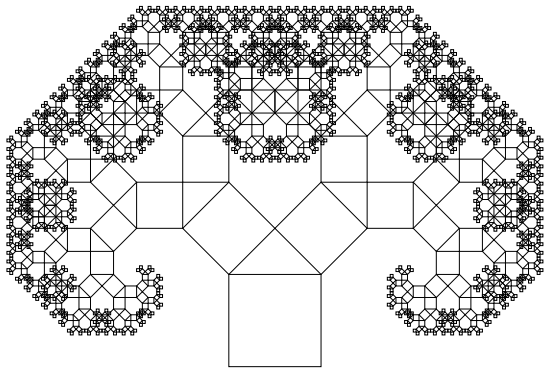
Powerful concepts

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



← Stylistic Flower

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



Pythagorean Tree →
BachoTEX 2011 pearl

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

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

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

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

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

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

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

- operand stack `pop exch dup ...`
- arithmetic and math `add div ... srand`
- array `array ... getinterval`
- dictionary `dict ...dictstack`
- string `string ... run ... token`
- relation, boolean, and bitwise `eq ... bitshift`
- type, attribute, and conversion `type ... cvs`
- file `file = == ... echo`
- file inclusion `run`
- virtual memory `save restore ...`
- miscellaneous `bind null usertime version`
- graphics state `gsave grestore ... currenttransfer`
- coordinate system and matrix `matrix ... invertmatrix`
- path constructing `newpath`, `moveto lineto curveto arcto ... clip eoclip ... charpath ...`

  Only one path is possible, although by juggling with several graphics states one may maintain several collateral paths
- painting to the current page of
  paths `stroke` paints lines, `fill eofill ...` fills an area
  strings `show ...`
  a sampled image `image shshow ...`
- device setup and output `showpage ...`
- character and fonts `findfont scalefont setfont` (or level 2 onward optimized concatenation of the 3 into `selectfont`) `makefont` (transforms more general than `scalefont`) `... kshow ... cshow ...`
- font cache `setcharwidth ...`
- errors `dictfull ...VMerror`.

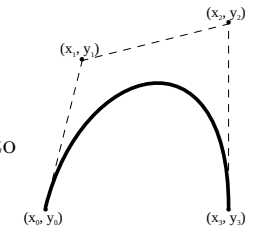
Overwhelming, isn't it?

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

`def` associates names with procedures or values available on the operand stack, and stores the associated pair in the current dictionary

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

curveto



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

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

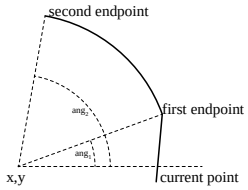
a_1 `lineto` adds a straight line to the path.

A point $z(t)$, t∈[0,1] on the line is given by $z(t) = (1-t)a_0 + ta_1$.

a_1 a_2 a_3 `curveto` adds a B-cubic to the path with end point a_3 and control points a_1 and a_2.

A point $z(t)$, t∈[0,1] on the B-cubic is given by $z(t) = \sum_{k=0}^{3}(1-t)^{3-k}t^k a_k$.

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



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

The Red, Green and Blue books — Reference Manual, PostScript Language Program Design, respectively Tutorial and Cookbook — are published by Addison-Wesley and also available for free on the WWW, even the level 1 and 2 (774p) LRM's. The Blue Book, which was aimed at to set a standard for effective PostScript programming, contains examples of procedures, such as `oshow outsidecircletext insidecircletext pathtext pathlength printposter DrawPieChart centerdash smallcaps` and `arrow`, to name but a few.[9] The Red Book is indispensable. The Green Book is about software engineering in PostScript, not only to get the programs to work, but to create correct, readable, efficient, maintainable and robust PostScript programs. The underlying goal is to develop a printer driver. There is also an Adobe White Book about Type 1 fonts, also for free on the WWW. Mnemonics: the RGB-collection of Adobe :-). PostScript programs, in ASCII, are generally generated by programs, hardly self-written. They facilitate exchange of (stand alone EPSF) picture descriptions, and of course (the pages of) a complete publication. The structuring conventions of Appendix C of the Red book level 1 have grown out into an Adobe Technical note #5001, 109p. Nowadays, illustrations are easily exchanged in `.pdf`, and everybody can view them everywhere, because of the ubiquitous, free Acrobat reader. The Mathematica reader is also freely available, and facilitates the exchange of Mathematica `notebooks`. The exchange of ASCII PostScript goes without problems and is useful.

Although a powerful graphical language, PostScript is considered low-levelish by the TEX community at large. They favour John Hobby's preprocessor MetaPost, Knuth included, with exceptions: Taco Hoekwater,

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

me ... . Taco includes PostScript on-the-fly in `escrito`, his PostScript compatible interpreter in Lua. He is also on the Metapost maintenance team and works on extensions of Metapost. At BachoTEX 2010 he announced the release of Metapost2.000.

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

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

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

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

Finally, and in contrast with TEX, a mnemonic

All what you type in PostScript are

comments after %

numbers

operators

names to be looked up in the dictionaries, and at last

strings which contain text.

From the LRM

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

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

So ...

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

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

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

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

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

- `%!` the start of a PostScript program with `%!PS-Adobe-3.0 EPSF-3.0` the complete line for illustrations to be encapsulated
- `%%` at the beginning of a line starts structural information about the PostScript program, as explained in Appendix C of the LRM version 1, or see ATN #5002; syntax `%%<keyword>: parameter values`.

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

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

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

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

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

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

Contents of the library

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

Constants are for example

- `/sqrt2 1.414213562 def`
  `/sqrt3 1.732050808 def`
  `/sqrt5 2.236067977 def`
  `/pi 3.14159265358979 def`
  `/reus 2 30 exp dup 1 sub add def` ie 2.14748e+09.
- font abbreviations, such as `/H14pt /Helvetica 14 selectfont def`
- colour presettings, such as `/lightred{1 0.3 0 setrgbcolor}`.

Definitions for operations, the utilities

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

Definitions for 2D fractal pictures mostly on the square inch as examples of 20th century Math geometry

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

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

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

```
/linearconstructionno1dict 50 dict def
linearconstructionno1dict begin
/reversevideo {-130 -135 260 270 rectfill} def %Mimics BoundingBox
%Data
/origin { 0 0 } def
/a0f  { .1 s  -2 s  -2 s ptp  }def%y constant
/a1f  { .6 s  -2 s  -1 s ptp  }def
/a2f  { .6 s  -2 s   1 s ptp  }def
/a3f  { .2 s  -2 s   2 s ptp  }def
...
/sampleellipsestroke{gsave -45 rotate .6 1.2 scale%kind of ellipse
...
stroke grestore}def
%
/approxellipsestroke{gsave -45 rotate .6 1.2 scale%kind of ellips
```

[10] An undocumented feature? Non-universal?

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

```
...
grestore} def
/frame{a0f moveto  a1f  a2f  a3f curveto
...} def
/dostringing{0.02 .02 .5001{/t exch def
...
closepath }for } bind def  .1 setlinewidth stroke
/annotation{-32 -125  moveto ...}
 ..} def
end%linearconstructionno1dict
```

Structure of each listed code contribution

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

Utilities

A few utilities have been selected and discussed, which exhibit PostScript programming paradigms.

Length. A too simple operator? Not so.

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

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

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

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

Overloading length[12]

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

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

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

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

[12] Courtesy Taco Hoekwater

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

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

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

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

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

x y angle `rot` x' y', rotates the point x y over the angle `angle` in degrees, counter-clockwise in the PostScript tradition

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

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

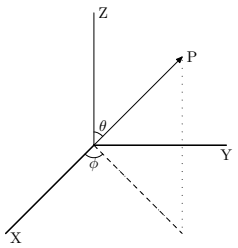
Projection of a point, operator ptp. For projections I specify the graphics in 3D and project the data onto the plane by the operator `ptp` with viewing angles as parameters.[13]

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

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

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

```
/ptp{% point x y z ==> x' y'
       % use: /pair { x y z ptp } def
       % parameters: a, b viewing angles
   0 begin
   /z exch def/y exch def/x exch def
   x neg a cos mul y a sin mul add
   x neg a sin mul b sin mul y neg a cos mul b sin mul add
   z b cos mul add
   end}def
/ptp load 0 3 dict put
```



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

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

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

[13] The projection I also coded in MetaPost.

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

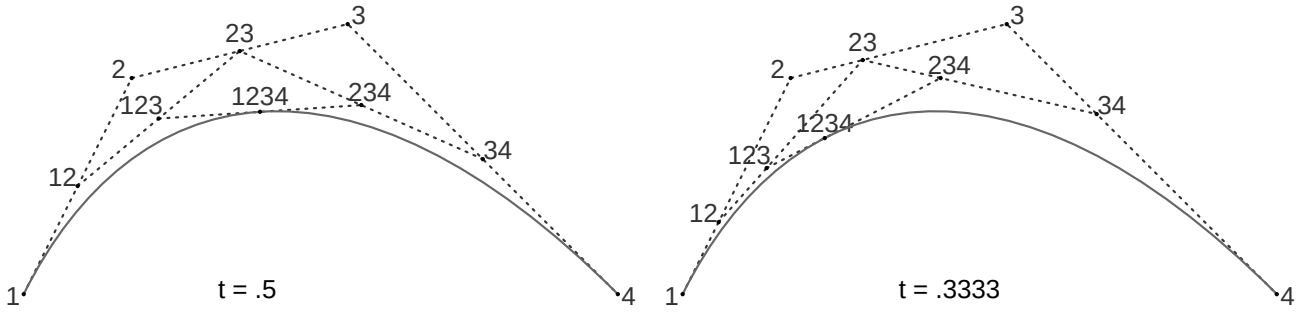
tonSpline yields the coordinates of a point on a spline given by the points (a0,a1,a2,a3)[14] as function of the time parameter $t \in [0,1]$: $z(t) = \sum_{k=0}^{3}(1-t)^{3-k}t^k a_k$. Algorithmically,[15]

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

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

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

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



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

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

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

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

Association of natural numbers with colours This functionality was needed in `LuliaP` where the esacpe number of the iterarion is associated with a colour.

```
/black{0 0 0 setrgbcolor}def .../yellow{1 1 0 setrgbcolor}% from PSLIB.eps
/colours{[/black .../yellow]}def
...                  %calculate escape number
colours k get cvx exec%k is escape number
```

The k*th* colour is now in use.

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

[14] It is interesting to see the control points back as coefficients in the Bernsteĭn polynomial representation.

[15] Named after de Casteljau.

shadow

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

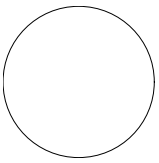
BoundingBox calculation by PostScript.

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

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

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

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



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

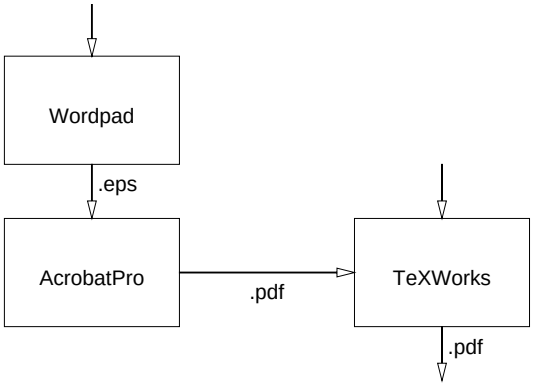
2D pictures

Merging .eps pictures in .tex documents: flowchartTeXandPS

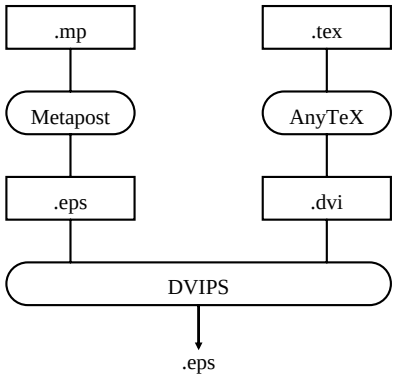
`.eps` pictures as such can't be included when processing with the TEX-engine pdfTEX, which I use in TEXworks. They first have to be converted into `.pdf`. They are more efficiently processed by TEXWorks when a priori converted into `.eps`.

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

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



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

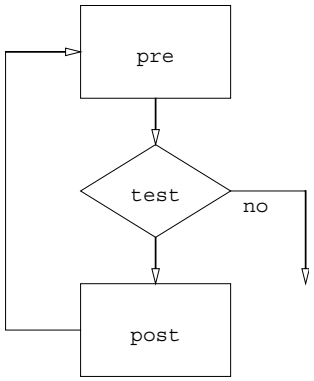


Loop flowcharts: flowchartloop

The picture has been discussed and presented in *Recreational Use of TEX&Co*

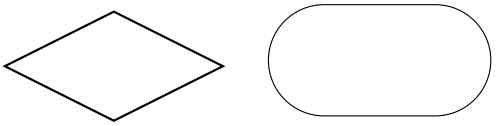
[16] GKP picture lost alas after 20 years, so I programmed it anew in PostScript.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Loopflowchart
%%BoundingBox: -101 -31 102 221
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%
/flowchartloopdict 11 dict def
flowchartloopdict begin%local auxiliaries
/Courier 14 selectfont 1.45 setmiterlimit
/s 50 def  /t s .61803 mul def
/-s s neg def /-t t neg def /2s s s add def /-2s 2s neg def
/2t t t add def /-2t 2t neg def/3t 2t t add def /6t 6 t mul def
end

/flowchartloop{flowchartloopdict begin
%used form PSlib.eps: arrow
-s -t 2s 2t rectstroke 0 -7 moveto (post) centershow   %post
rectangle
-s 0 moveto -2s 0 lineto 0 6t rlineto                  %left line
-2s 6t -s 6t .5 5 10 arrow stroke                      %left arrow
%diamond
0 3t translate
%-s 0 moveto 0 t lineto s 0 lineto 0 -t lineto closepath stroke
-s -t 2s 2t diamondstroke
0 -7 moveto (test) centershow
0 2t 0 t .5 5 10 arrow                                 %connection
arrow
0 -t 0 -2t .5 5 10 arrow                               %connection
arrow
s 0 moveto 2s 0 lineto 2s 0 2s -2t .5 5 10 arrow stroke%result arrow
s -14 moveto ( no) show
%
0 3t translate
-s -t 2s 2t rectstroke 0 -7 moveto (pre) centershow    %pre rectangle
end}def
%%EndProlog
flowchartloop showpage
%%EOF
```

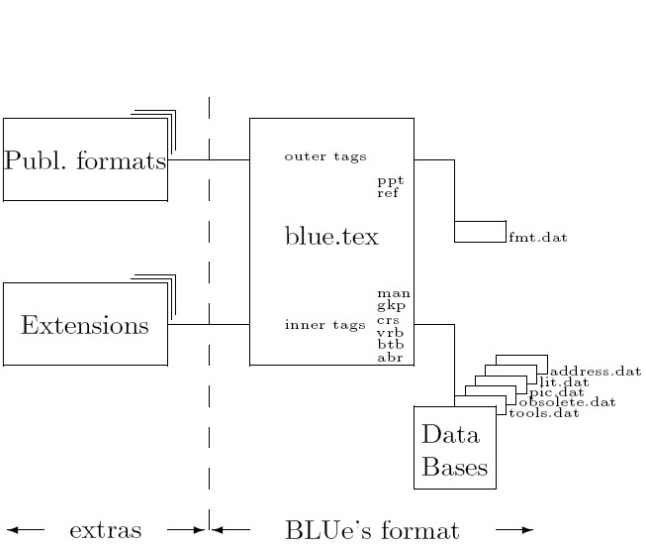


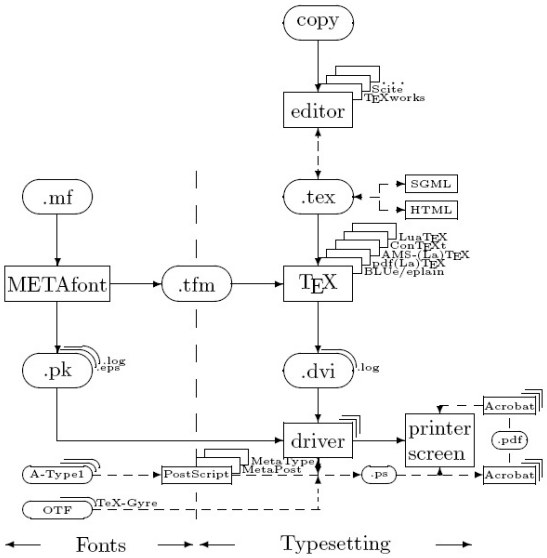
`PSlib.eps` contains `diamondstroke` and `ovalstroke` as building blocks for flow charts. PostScript already provides `rectstroke`.

```
/ovalstrokedict 7 dict def
/ovalstroke{%stack: llx lly sx sy%lower left point and size
ovalstrokedict begin
/sy exch def /sx exch def /lly exch def/llx exch def
/r sy 2 div def
/x llx r add def /y lly r add def
llx sx add r sub lly moveto
x y r 270 90 arcn
/x llx sx add r sub def /y lly r add def
x y r 90 -90 arcn  closepath stroke
end}def
```



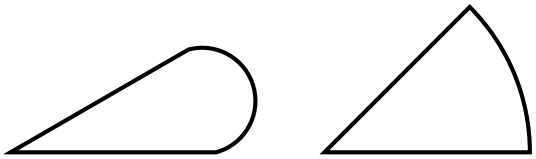
My most complex TEX-alone flowcharts of old. The TEX-MF picture has been adapted on occasion of EuroTEX2012. What I called databases at the time function as libraries. The pictures are included in `pic.dat`.





Wedges : wedge from Blue book and Red book (arc documentation)

Borrowed from Adobe's Bluebook P1 p129, which is available for free on the WWW. The code has been collected in the library `Bluebook.eps` and `PSlib.eps`. Wedge is an exercise in transformation of User Space combined with the use of arc. The right wedge is borrowe from Adobe's Red book.

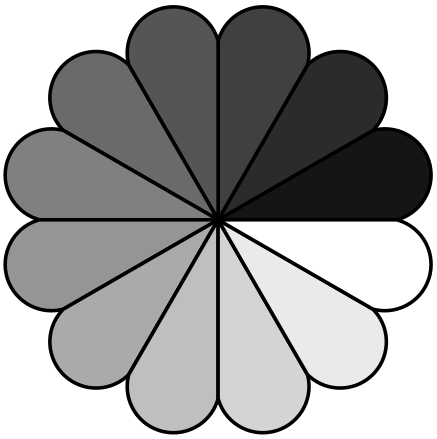


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Wedge Blue Book Program 1, p129
%%BoundingBox:-2 -1 187 52
%%BeginSetup
%%Endsetup
%%BeginProlog
(C:\\PSlib\\Bluebook.eps) run
/inch {72 mul} def
/wedge{ 0 0 moveto
        1 0 translate 15 rotate 0 15 sin translate 0 0 15 sin -90 90 arc
  closepath
} def
/wedgerefman{0 0  moveto 0 0 1 0 45 arc closepath}def
%%EndProlog
gsave 1 inch dup scale
  wedge 0.02 setlinewidth stroke
grestore
110 0 translate 1 inch dup scale
wedgerefman 0.02 setlinewidth stroke
showpage
%%EOF
```

Repeated shapes: rosette

Rosette is borrowed from Adobe's Blue book P1 p129, which is available for free on the WWW. The code has been collected in the library `Bluebook.eps` and `PSlib.eps`. Rosette is an exercise in transformation of User Space combined with the use of `arc` and shading with nuances of grey.

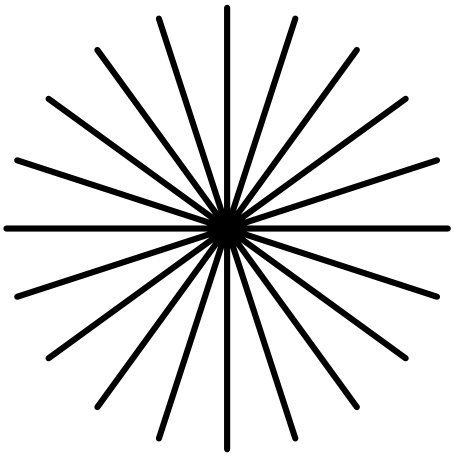
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Rosette, Blue Book Program 1  p129
%%BoundingBox:-160 -160 160 160
%%BeginSetup
%%Endsetup
%%BeginProlog
(C:\\PSlib\\Bluebook.eps) run
/rosette{gsave
  1.75 inch dup scale 0.02 setlinewidth
  1 1 12{ 12 div setgray
  gsave wedge gsave fill grestore
    0 setgray stroke%contour
  grestore
  30 rotate
} for
grestore
} def
%%EndProlog
rosette showpage
%%EOF
```



Line bundle

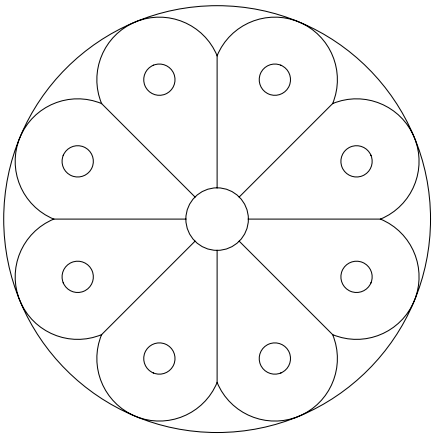
PSTricks should have difficulties with the line bundle, because it is based on the picture User Interface of LaTeX.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Linebundle. cgl 1997
%%BoundingBox: -37 -37 37 37
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/linebundledict 1 dict def
/linebundle{linebundledict begin %jalxii
/r 36 def
20{0 0 moveto r 0 lineto 18 rotate}repeat
stroke end}def
%%EndProlog
linebundle showpage
%%EOF
```



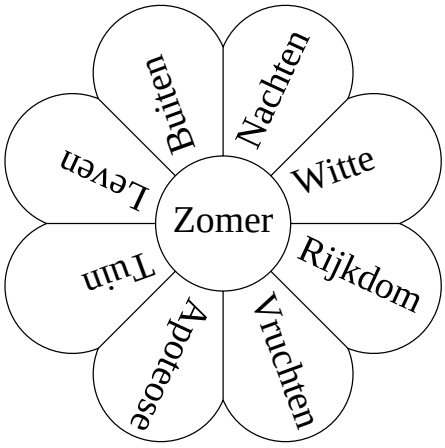
Rosette variants

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title:  Barnwindow
%%BoundingBox: -180 -180 180 180
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
/barnwindowdict 3 dict def
barnwindowdict begin
/l 36 def  /r {l 22.5 sin mul} def /m {l 22.5 cos mul} def
end
/barnwindow{barnwindowdict begin
8{r .5 mul 0  moveto  l 0 lineto
  currentpoint   %begin circular arc
  22.5 rotate m 0%center
  r   -90 90 arc  22.5 rotate
}repeat
/rin {r .5 mul} def rin 0  moveto 0 0 rin 0 360 arc   %inner circle
/rout {r m add} def rout 0  moveto 0 0 rout 0 360 arc %outer circle
%extra circles
/rin {r .25 mul} def
22.5 rotate
8{m rin add 0 moveto
  m 0 rin 0 360 arc
  45 rotate }repeat stroke
end} def
barnwindow showpage
%%EOF
```



Variant zomer, with texts

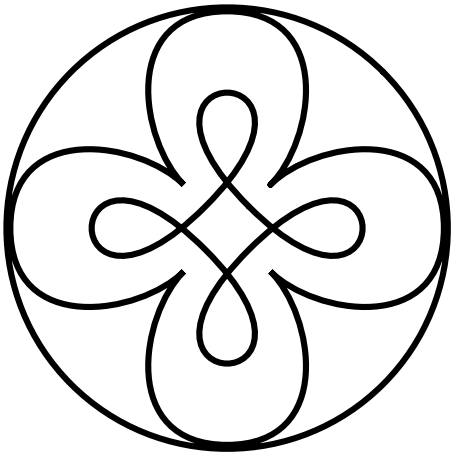
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Zomer, vormgedicht, CGL, April 2007, Barnwindow basis
%%BoundingBox: -180 -180 180 180
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
/zomerdict 4 dict def
zomerdict begin /l 140 def /Times-Roman 30 selectfont
                /m l 22.5 cos mul def /r l 22.5 sin mul def
          end
/zomer{zomerdict begin
8{r 0 moveto 22.5 rotate
   m 0 r -90 90 arc 22.5 rotate}repeat
%inner circle
r 0 moveto 0 0 r 0 360 arc stroke
-40 -7 moveto (Zomer) show
/texts[(Witte) (Nachten) (Buiten) (Leven) (Tuin)
         (Apoteose) (Vruchten) (Rijkdom)]def
0 0 moveto 20 rotate /r r 10 add def
0 1 7{r 0 moveto texts exch get show 45 rotate}for
end} def
zomer showpage
%%EOF
```



Nice curves

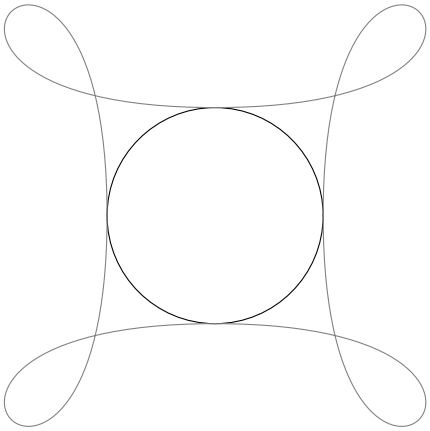
The problem is how to choose the control points.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Malbork
%%BoundingBox: -37 -37 37 37
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/malbork{45 rotate 10 0 moveto
4{20 0 37.5 12.5 25 25 curveto
  12.5 37.5 0 20 0 10  curveto
  90 rotate}repeat
5 0 moveto
4{5 35 35 5 0 5 curveto
  90 rotate}repeat
36 0 moveto 0 0 36 0 360 arc
stroke}def
%%Endprolog
malbork
showpage
%%EOF
```

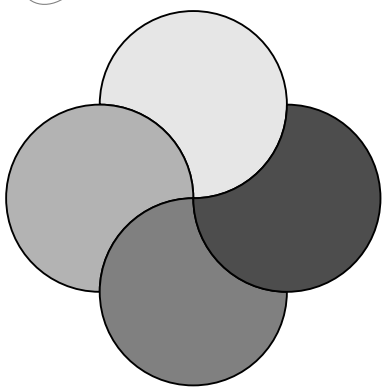


Variant

```
%!PS-Adobe-3.0 EPSF-3.0
%BoundingBox: -210  -210  210  210
%%BeginSetup
%%EndSetup
(c:\\PSlib\\Pslib.eps) run
%%EndProlog
0 0 100   0 360 arc 0 setgray stroke
%B-cubic
4{100   0 moveto  100  370   370  100  0 100 curveto
  90 rotate}repeat .5 setgray stroke
showpage
%%EOF
```

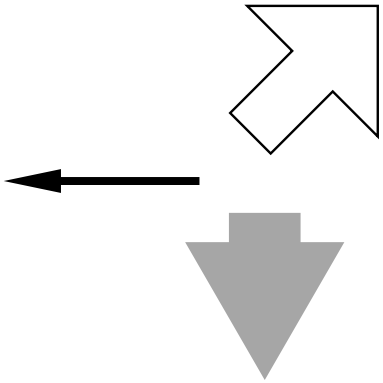


```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Overlappingdisc. cgl 1997
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/overlapddict 3 dict def
/overlapdisc{overlapddict begin %jalxxvii
/r 50 def /tint .9 def
0 0 moveto
4{0 r r -90 180 arc
  r neg  0 r 90 0 arcn
  closepath
  gsave tint setgray fill grestore stroke
  /tint tint .2 sub def
  90 rotate
}repeat
end}def
%%EndProlog
overlapdisc showpage
%%EOF
```



Arrows : arrow. Blue book P4 p137

Borrowed from Adobe's Blue book, which is available for free on the WWW. The code has been collected in the libraries `Bluebook.eps` and `PSlib.eps`. I use arrows abundantly in flowcharts, for coordinate axes, ...

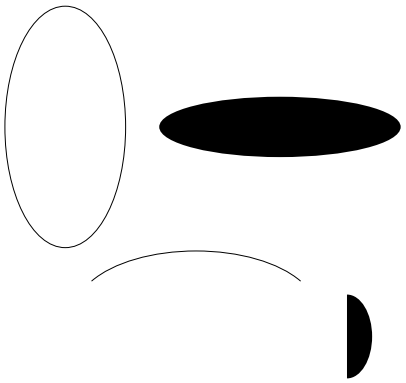


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Blue Book Arrow Program 4, page 137
%%BoundingBox: 300 200 450 600
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSLib\\Bluebook.eps) run
%
/arrowdict 14 dict def%local dictionary
arrowdict begin /mtrx matrix def end
/arrow{ arrowdict begin
  /headlength exch def%parameters are assigned
  /halfheadthickness exch 2 div def
  /halfthickness exch 2 div def
  /tipy exch def /tipx exch def
  /taily exch def /tailx exch def
%
  /dx tipx tailx sub def%auxiliaries
  /dy tipy taily sub def
  /arrowlength dx dx mul dy dy mul add sqrt def
  /angle dy dx atan def
     /base arrowlength headlength sub def
%
  /savematrix matrix currentmatrix def
%
  tailx taily translate angle rotate
%
  0 halfthickness neg moveto
  base halfthickness neg lineto
  base halfheadthickness neg lineto
  arrowlength 0 lineto
  base halfheadthickness lineto
  base halfthickness lineto
  0 halfthickness lineto
  closepath
  savematrix setmatrix
end } def
%%EndPrologue
newpath 318 340 72 340 10 30 72 arrow fill
newpath 382 400 542 560 72 232 116 arrow 3 setlinewidth stroke
newpath 400 300 400 90 90 200 200 3 sqrt mul 2 div arrow .65 setgray fill showpage
%%EOF
```

Elliptical Arcs. Blue book P3 p133

I use elliptical arcs for example in Mondrian pictures as cropping area. Elliptical arcs are easily obtained by scaling `arc`, as shown in the code.

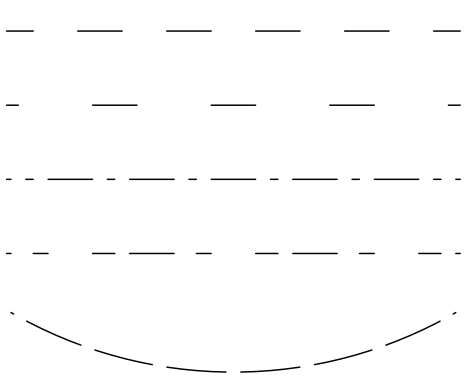
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Blue Book Elliptical Arcs Program 3, page 134
%%BoundingBox: 65 99 547 550
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSLib\\Bluebook.eps) run
%
/ellipsedict 8 dict def
ellipsedict /mtrx matrix put
/ellipse{ ellipsedict begin
    /endangle exch def
    /startangle exch def
    /yrad exch def
    /xrad exch def
    /y exch def
    /x exch def
%
    /savematrix mtrx currentmatrix def
    x y translate xrad yrad scale
    0 0 1 startangle endangle arc
    savematrix setmatrix
    end
} def
%%EndPrologue
newpath 144 400 72 144 0 360 ellipse stroke
newpath 400 400 144 36 0 360 ellipse fill
newpath 300 180 144 72 30 150 ellipse stroke
newpath 480 150 30 50 270 90 ellipse fill
showpage
%%EOF
```



Centered Dash Patterns. Blue book P5 p143

In illustrations dashed lines occur frequently.

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



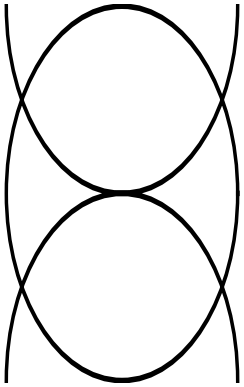
Interesting is the use of `flattenpath` in `pathlength` for calculating the length of a path, be it a Bézier cubic (or a circulare arc), or a line, respectively an arbitrary combination.

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

CGLLOGO

Created as PostScript def on the occasion of EuroTEX2012.

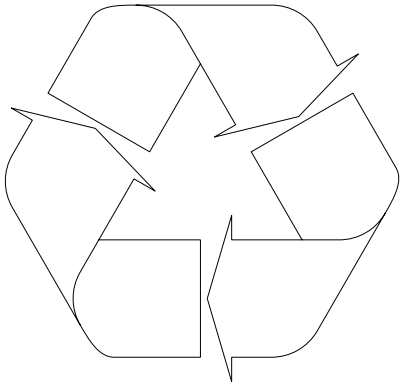
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: CGLLOGO
%%BoundingBox: -63 -101 63 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/cgllogodict 2 dict def
cgllogodict begin /r 100 def /-r r neg def end
/cgllogo{cgllogodict begin
3 setlinewidth .618 1 scale
r 0 moveto  0 0 r 0 360 arc
2{r -r moveto 0 -r r 0  180 arc
  1 -1 scale }repeat
stroke
end}def
%%EndProlog
cgllogo
showpage
%%EOF
```



Recycle logo

The old Metafont code was adapted to and processed by Metapost and finally edited, where redundances have been omitted.

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -161 -165 163 145
%%BeginProlog
%%BeginResource:
%%EndResource
%%EndProlog
%%BeginSetup
%%EndSetup
1 setlinecap 1 setlinejoin 10 setmiterlimit
3{newpath -83.33 -48.11 moveto
0 -48.11 lineto 0 -144.34 lineto -70.83 -144.34 lineto
-83.33 -144.34 -91.59 -130.04 -98.61 -117.88 curveto
-106.35 -104.48 -106.35 -87.97 -98.61 -74.57 curveto
-54.47 1.89 lineto -37.15 -8.11 lineto -86.13 43.26 lineto
-155.12 60 lineto -137.80 50 lineto -154.17 21.65 lineto
-161.90 8.25 -161.90 -8.25 -154.17 -21.65 curveto
-98.61 -117.88 lineto 120 rotate stroke}repeat
showpage
%%EOF
```



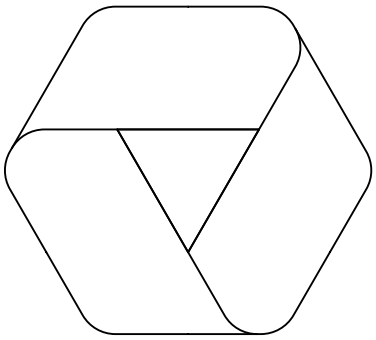
My old Metfont code of 20 years ago, adapted to Metapost.

```
\beginfig(0)
s:=500;                                 %size of the circumscribing triangle
       d:=s*(sqrt3)/9;                  %width arrow
       dd:=s/3;                         %d*sqrt3
       x:=.02s;                         %parameter for arrowhead
       delta:=.025s;                    %distance in absis from corners to get rounded corners
pair a, ah;
a:=(-.5d*sqrt3,-.5d);                   %‘center’ point
ah:=a+(.5(.6d-.5dd)+x,.5(.5d+.6dd)+x*sqrt3);%arrowhead
path p;
%arrowtail and one line extra
p:=
%tail
   a--(0,-.5d)--(0,-1.5d)--
   a+(delta,-d){left}...
   {dir120}a-(dd/6-delta,.5d+delta*sqrt3)...
   {dir60}a-(dd/6-delta,.5d-delta*sqrt3)--a+(.3d,.3dd)--
%arrowhead
   a+(.3d+x*sqrt3,.3dd-x)--ah--a+(.5(.6d-dd-2x*sqrt3),.5(d+.6dd+2x))--
%folded part
   a+(.5(.6d-dd),.5(d+.6dd))--
   a+(-.5dd+delta,.5d+delta*sqrt3){dir-120}..
   {dir-60}a+(-.5dd+delta,.5d-delta*sqrt3)...
   a-(dd/6-delta,.5d+delta*sqrt3);
%drawdot origin;
draw p;
draw p rotated-120;
draw p rotated120;
addto currentpicture also currentpicture shifted (0,-.75s);
endfig
end
```

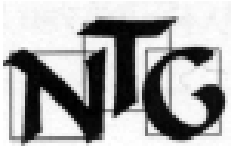
Old logo

The curved corners of the hexagon are special. Intersection points had to be calculated.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: ribbon. cgl 2014
%%BoundingBox: -100 -100 100 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/r 100 def /4pop{pop pop pop pop} def
6{0  r 60 sin mul moveto
  .5 r mul 60 sin r mul 0 60 sin r mul -60 rot r 5 div arcto 4pop
  0 60 sin r mul -60 rot lineto -60 rotate}repeat stroke
3{0 60 sin r mul -.5 mul moveto
0 60 sin r mul -.5 mul 120 rot lineto stroke 120 rotate}repeat
3{0 60 sin r mul -.5 mul 0 60 sin r mul -.5 mul 120 rot
r 0 r 0 60 rot intersect
/y exch def /x exch def
0 60 sin r mul -.5 mul moveto
x y  r 0 60 rot r 5 div
arcto stroke 120 rotate}repeat
%%EOF
showpage
%%EOF
```



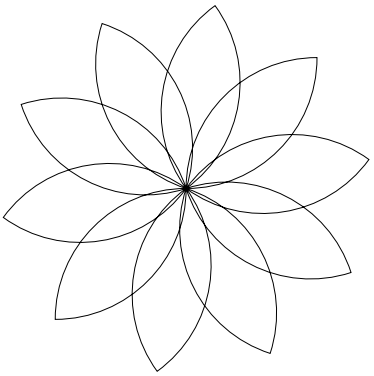
NTG's logo was renewed by Siep Kroonenberg in Adobe Illustrator.



Flower

Created as PostScript def on the occasion of EuroTEX2012.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Flower from Just a little bit of PS
%%BoundingBox: -26 -26 26 26
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/flower{%jalviii
//r 18 def
10 {0 r r 270 360 arc
      r 0 r   90 180 arc
     36 rotate} bind repeat
stroke
}def
/Endprolog
flower showpage
%%EOF
```



Postprocessed in Photoshop, enriched with colours.

Adobe's Blue book CD label, part1

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

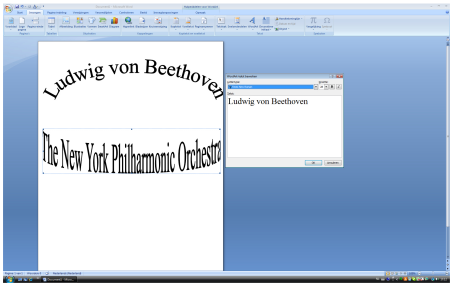
Insidecircletext prints the text in a counter-clockwise fashion with its baseline along the circumference, on the inside of a circle.

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

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

Circular Text by Photoshop and Word

← Photoshop

Word→



Adobe's Blue book CD label, part2

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

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

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

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

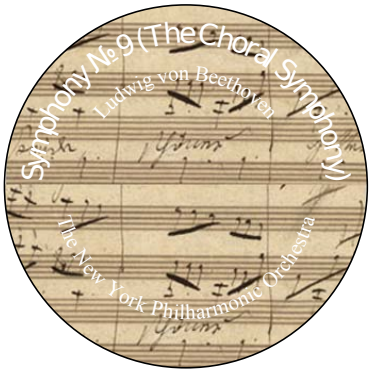
Adobe's Blue book CD label, complete

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


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Blue Book CD Label Program 10 p167
%%BoundingBox: -166 -166 166 166
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\Bluebook.eps) run

/circtextdict 19 dict def
circtextdict begin /pi 3.1415923 def
/findhalfangle{ stringwidth pop 2 div
  2 xradius mul pi mul div 360 mul} def
/outsideplacechar{ /char exch def
/halfangle char findhalfangle def
gsave halfangle neg rotate
  radius 0 translate -90 rotate
  char stringwidth pop 2 div neg 0 moveto
  char show
grestore halfangle 2 mul neg rotate} def
/insideplacechar{ /char exch def
/halfangle char findhalfangle def
gsave  halfangle rotate
  radius 0 translate 90 rotate
  char stringwidth pop 2 div neg 0 moveto
  char show
grestore halfangle 2 mul rotate} def
end%circtextdict

/outsidecircletext{ circtextdict begin
        /radius exch def /centerangle exch def  /ptsize exch def  /str exch def
  /xradius radius ptsize 4 div add def
  gsave
    centerangle str findhalfangle add rotate
    str{ /charcode exch def
    ( ) dup 0 charcode put outsideplacechar
  } forall
  grestore
end} def
/insidecircletext{ circtextdict begin
  /radius exch def  /centerangle exch def /ptsize exch def  /str exch def
  /xradius radius ptsize 3 div sub def
  gsave centerangle str findhalfangle sub rotate
    str{ /charcode exch def
    ( ) dup 0 charcode put insideplacechar
  } forall
  grestore
end} def
 %%EndProlog
/cddvdlabel{0 0 165 0 360 arc stroke
/Times-Bold 22 selectfont (Symphony No. 9 (The Choral Symphony)) 22 90 140 outsidecircletext
/Times-Roman 15 selectfont
   (Ludwig von Beethoven) 15 90 118 outsidecircletext
   (The New York Philharmonic Orchestra) 15 270 118 insidecircletext
}def
 /EndProlog
cddvdlabel
%%EOF
```

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

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




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

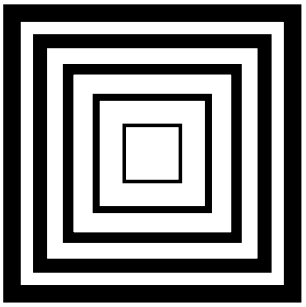
In MS Word and Nero (Version 10 with LightScribe) one can create labels interactively. LightScribe, a HP technique to burn labels on special discs as well, allows only for black and white labels.

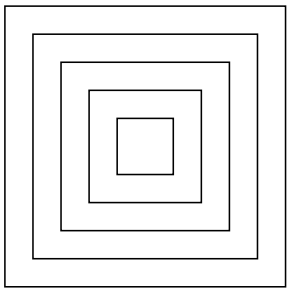
My wife, not at all a TEXie, designs labels in Photoshop and prints them on prefab glued paper by the tool CDFACE1.6 (Media labeling software templates for: CDs, DVDs, jewel cases, envelopes, floppy discs, audio cassettes, dat tapes, zip discs). The special tool CDFACE can handle all, no PhotoShop is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

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

Nest of squares. Blue book, P2 p131

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

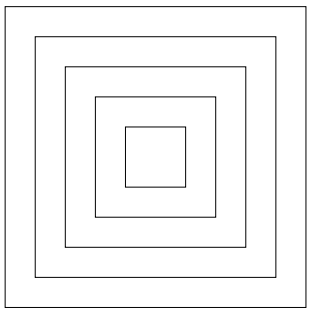




```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nest of squares Adobe
%%BoundingBox: 80 335 525 529
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nestofsquares% used from library: centersquare
{gsave 2.5 inch 6 inch translate 1 16 div setlinewidth
  1 1 5{ gsave  .5 mul inch dup scale
    centersquare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

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

Variant nest of squares



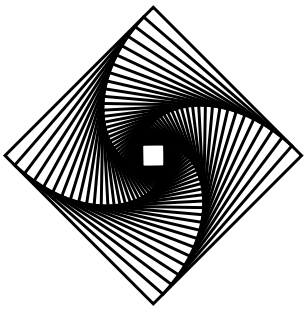
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nest of squares, variant
%%BoundingBox: -181 -181 181 181
%%BeginSetup
%%EndSetup
1 1 5{ 36 mul /s exch def /-s s neg def /2s s s add def
          -s -s 2s 2s rectstroke
       } for
showpage
%%EOF
```

Straight forward procedural approach, with `linewidth` invariant.

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

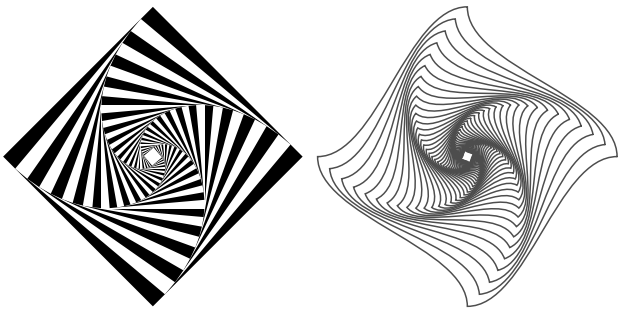
Shrinking squares

A triviality? Barnsley used the picture to demonstrate shrinking metric spaces.



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Shrinking squares, cgl 97, 2007
%%BoundingBox: -51 -51 51 51
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/shrinkingsquaresdict 4 dict def
shrinkingsquaresdict begin
/r 50 def /alpha 4 def /c 1 alpha cos alpha sin add div def
/square{r 0 moveto 0 r lineto r neg 0 lineto 0 r neg lineto
  closepath stroke}def
end
/shrinkingsquares{shrinkingsquaresdict begin
35{square  /r r c mul def  alpha rotate}repeat
end}def
%%EndProlog
shrinkingsquares showpage
%%EOF
```

Variant by Jackowski, where he changed the side

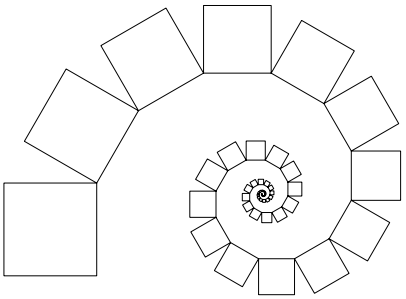


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nest of squares with curved sides
%%BoundingBox: -181 -181 181 181
%%BeginSetup
%%EndSetup
%%Title: shrinking ‘squares’
%%BoundingBox: -101 -101 311 101
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run % used: anglemark s ptp S12pt H12pt
/shrinkingsquaresjackodict 7 dict def
shrinkingsquaresjackodict begin
0 setlinejoin 1 setlinecap
/r 100 def /alpha 5 def
/c 1 alpha cos alpha sin add div def
/square{r 0 moveto
  0 r lineto r neg 0 lineto 0 r neg lineto
  closepath fill}def
/flipflop true def
end
/shrinkingsquaresjacko{shrinkingsquaresjackodict begin
gsave 36{flipflop{0 setgray}{1 setgray}
           ifelse square
   /flipflop flipflop not def
   /r r c mul def
   alpha rotate}repeat
grestore
210 0 translate  /r 100 def
/square{r 0 moveto
  4{r .4 r mul 0 .6 r mul 0 r curveto 90 rotate}repeat
  stroke }def
.3 setgray
40{square
   /r r c mul def
   alpha rotate}repeat
end}def
%%EndProlog
shrinkingsquaresjacko
%%EOF
```

Shrinking squares II

A triviality?

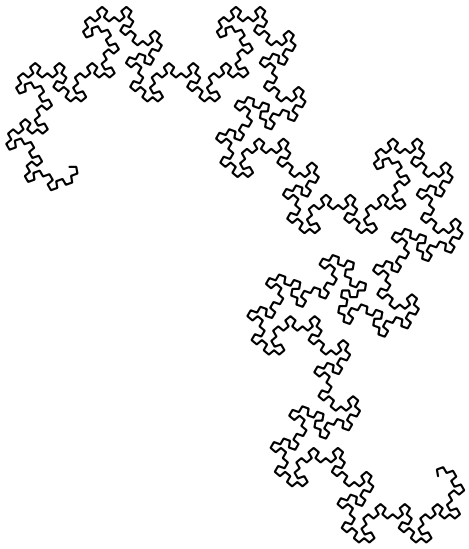
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: G-squares. cgl 1997
%%BoundingBox: -51 -14  168 149
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/G-squares{%jalx
/r 50 def
0 0 moveto 90 rotate
50{r 0 rmoveto
   1 1 4{0 r rlineto
         90 rotate}for
   -30 rotate
   .9 .9 scale
}repeat
stroke
}def
G-squares showpage
%%EOF
```



Dragon curve

Inevitable when working with TEX to neglect the dragon curves.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Draak. cgl 2011
%%BoundingBox: -40 -190 200 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/draakdict 10 dict def
/draak %dragon curve with angle A and order p, global scaling s
       %angle p ==> dragon curve
{draakdict begin /p exch def /b 180 3 -1 roll sub def%angle B
/s {100 mul} def /h 2 p -2 div exp s def %size piece scaled
/s 0 def
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 1 def}{/d -1 def}ifelse
  /s s d sub def
  h s b mul cos mul h s b mul sin mul rlineto
}for %n
end}def
100 10 draak stroke showpage
%%EOF
```



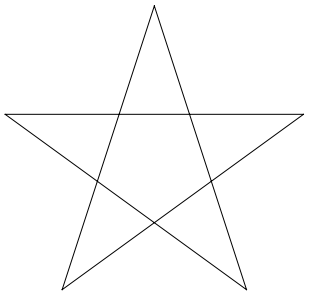
The above code is a translation of Lauwerier's BASIC code

```
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
```

Adobe's Blue book star, p51

Bogusław Jackowski taught NTG members in the early 90-ies Metafont and showed us how to draw a star, similar to Adobe. Using Metafont for graphics nowadays is not practical. MetaPost is used for the purpose, although the resulting PostScript is hard to read, and ipso facto to modify. I simplified Adobe's star by neglecting the `fill` cq `eofill`.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Star Adobe, Blue book p51
%%BoundingBox: -2 -45 73 28
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/starside{72 0 lineto currentpoint translate -144 rotate}def
/star5{5{0 0 moveto starside}repeat closepath stroke}def
%%EndProlog
star5
showpage
%%EOF
```

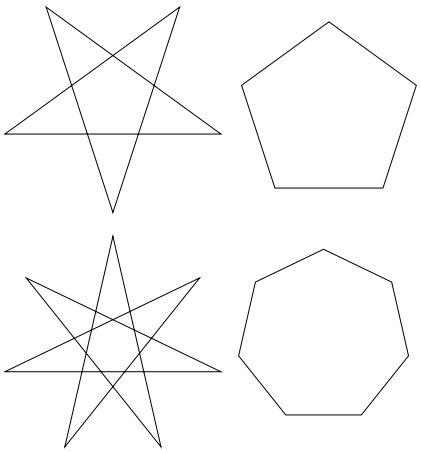


Interesting use has been made of `currentpoint` and the user space transformations `translate` and `rotate`. PostScript programming paradigms.

Variant stars cq polygons

In *Classical Math Fractals in Postscript* I generalized the use of drawing an n-star, such that also the n-gon can be drawn, depending on the values of the arguments on the stack, especially the angle. Interesting.

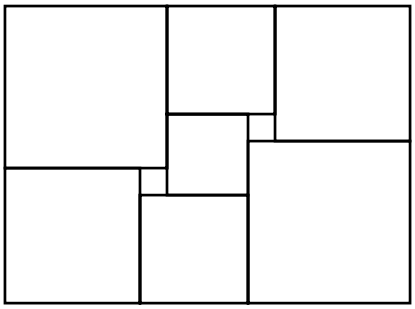
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Polygon and star
%%BoundingBox: -100 -150 95 60
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/gonstardict 2 dict def
/gonstar%stack: p (=angle) v (=vertices) ==> star cq gon
{gonstardict begin /v exch def /angle exch def
0 0 moveto v{angle rotate l 0 rlineto}repeat closepath
end} bind def
%%EndProlog
%
/l 100 def   144  5 gonstar stroke %5-star
gsave 75 -25 translate /l 50 def 1.415 setmiterlimit
72 5 gonstar stroke grestore       %5-gone
%extra: 7 star and 7 gon
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%7-gone
%%EOF
```



The n-gon is obtained with angle 360/n. This occurs also with Adobe's fractal arrow, which may yield interesting pictures with a different angle parameter.

Squaring the square

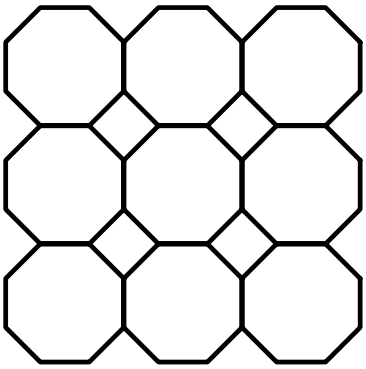
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: squaringthesquare, cgl 1997
%%BoundingBox: -76 -56 76 56
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/squaringthesquaredict 4 dict def
/squaringthesquare{squaringthesquaredict begin %tilLxiii
0 setlinejoin 1 setlinecap
/square{%s(ide) on stack
  /s exch def /hs .5 s mul def /mhs hs neg def
  mhs mhs moveto  hs mhs lineto
  hs  hs  lineto  mhs hs lineto
  closepath stroke}def
30 square
2{gsave -45 25 translate 60 square grestore
  gsave   5 35 translate 40 square grestore
  gsave  50 30 translate 50 square grestore
  -1 -1 scale}repeat
end}def
%%EndProlog
squaringthesquare showpage
%%EOF
```



Tiling

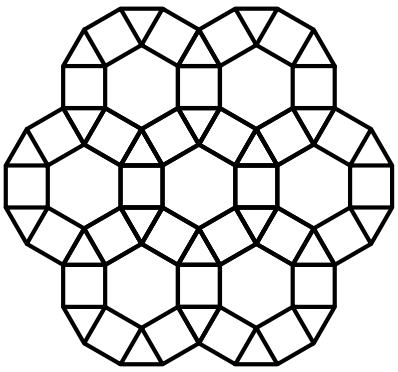
Tilings are classified according to the polygons at the corner points. Here 2 octogons and one square come together. The square is spurious. No `ngon` is invoked.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Tiling 4,8,8. cgl 96
%%BoundingBox: -40 -40 40 40
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/tile488dict 6 dict def
/tile488{tile488dict begin
/a 25 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/r ha 22.5 cos div def
/tile{22.5 rotate r 0 moveto
  8{45 rotate r 0 lineto}repeat
}def
ma a a{/i exch def
ma a a{/j exch def
   gsave i j translate tile stroke grestore
}for}for
end}def
 %%EndProlog
tile488 showpage
%%EOF
```



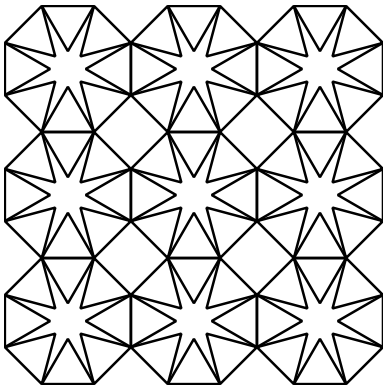
Just squares appropriately placed. Much in common with Pythagoras trees, no diminishing square size. Mininutieus use of translate and rotate.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Tiling 3.4,6,4. cgl 96
%%BoundingBox: -50 -45 50 45
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/tile3464dict 5 dict def
/tile3464{tile3464dict begin /a 10 def
/ha .5 a mul def /mha ha neg def
/s 1.366 a mul def
/tile{6{gsave s 0 translate %to center of square
  ha  ha moveto mha  ha lineto mha mha lineto ha mha lineto closepath
   %0 2.732 ha mul lineto stroke %to top on square
 grestore 60 rotate}repeat %ring
}def
6{gsave 2 s mul 0 translate tile grestore 60 rotate}repeat%next ring
end}def
%%EndProlog
tile3464 showpage
%%EOF
```



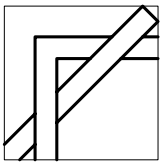
Tiling with octagons filled with stars

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Octagons. cgl 1997
%%BoundingBox: -75 -75 75 75
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/octagonsdict 13 dict def
/octagons{octagonsdict begin%tilxxxiv
/a 50 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/r ha 22.5 cos div def
/hs r 22.5 sin mul def /mhs hs neg def
/hss 1.732 hs mul def /mhss hss neg def
/s 2 hs mul def
/tile{8{ha hs moveto
  mhss mhs rlineto
  ha mhs lineto closepath
  stroke
  45 rotate}repeat
}def
ma a a{/i exch def
ma a a{/j exch def
   gsave i j translate
         tile stroke grestore
}for}for
end}def
%%EndProlog
octagons showpage
%%EOF
```

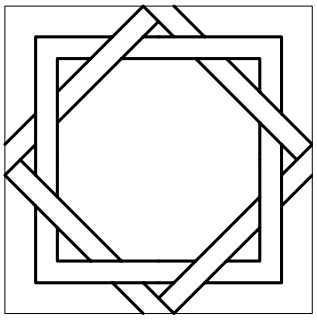


Square knots

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Square knots. cgl 1997
%%BoundingBox: -51 -71 125 52
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/squareknotsdict 11 dict def
/squareknots{squareknotsdict begin %tiliii BoundingBox: -51 -71 125 52
1 setlinejoin /s 50 def  /ms s neg def
/d 5 def /dst d 1.414 mul def /md d neg def
/a s d 2 mul sub def           /ma a neg def
/aa a dst sub def              /maa aa neg def
/element{
ms d moveto
    ma d 3 mul lineto    maa d 3 mul dst add moveto
    md s lineto    0 s d sub lineto
    maa d dst add lineto
ma  d moveto     ma d sub 0 lineto %
ma 0 moveto
    ma a lineto
    d -3 mul a lineto
    md a moveto
    0 a lineto    maa 0 moveto
    maa aa lineto
    d -3 mul dst sub aa lineto
md dst sub aa moveto
    0 aa lineto
 stroke}def
/tile{4{element 90 rotate}repeat}def %
/Times-Roman findfont 10 scalefont setfont
%
element%of tile
gsave 0 0 moveto
      ms 0 lineto
      ms s lineto
      0 s lineto closepath
     .2 setlinewidth stroke
ms 10 add -20 moveto (element)show
grestore
s 1.5 mul 0 translate
tile
gsave s ms moveto
      ms ms lineto
      ms s lineto
      s s lineto closepath
     .2 setlinewidth stroke
      ms 25 add ms 20 sub moveto (square/tile)show
grestore
end}def%end square knots
%%Endprolog
squareknots showpage
%%EOF
```



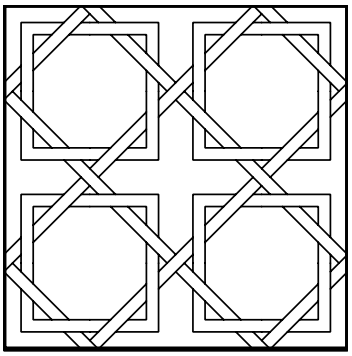
element



square/tile

Square knots tile

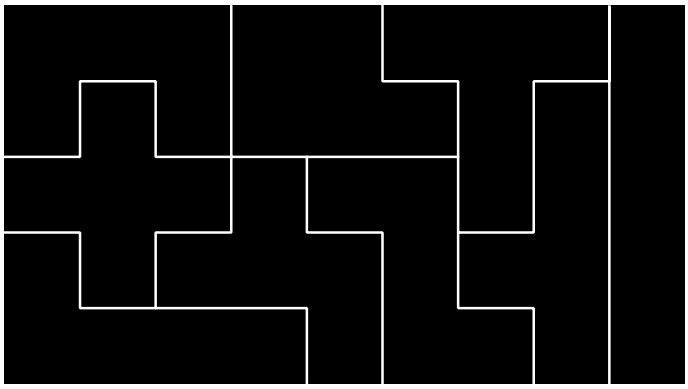
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Tile of twisted squares. cgl 1997
%%BoundingBox: -100 -120 100 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/twistedsquaresdict 13 dict def
/tiletwistedsquares{twistedsquaresdict begin %tiliv
/s 50 def                       /ms s neg def
/d 5 def                        /md d neg def
/dst d 1.414 mul def
/a s d 2 mul sub def            /ma a neg def
/aa a dst sub def               /maa aa neg def
/element{4{
ms d moveto
           ma d 3 mul lineto
           maa d 3 mul dst add moveto
           md s lineto
0 s d sub lineto
           maa d dst add lineto
ma  d moveto
           ma d sub 0 lineto
           ma 0 moveto
           ma a lineto
d -3 mul a lineto
md a moveto
           0 a lineto
           maa 0 moveto
           maa aa lineto
d -3 mul dst sub aa lineto
md dst sub aa moveto
           0 aa lineto
           90 rotate}repeat stroke
}def %element
%
/Times-Roman findfont 10 scalefont setfont
gsave ms s  translate element grestore
gsave  s s  translate  element grestore
gsave ms ms translate element grestore
gsave  s ms translate element grestore
/ts s 2 mul def
/fs ts ts add def
ts neg dup moveto
fs 0       rlineto
0 fs       rlineto
fs neg  0  rlineto
closepath
3 setlinewidth stroke
ms 20 add ms 2 mul 20 sub moveto (clipped pattern)show
end}def %tile twisted squares
%%EndProlog
tiletwistedsquares showpage
%%EOF
```



clipped pattern

Pentominoes

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pentominoes. cgl 1997
%%BoundingBox: -270 -150 270 150
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/pentominoesdict 10 dict def
/pentominoes{pentominoesdict begin %tilv
/u 30 def /mu u neg def
/t u 3 mul def /v u 5 mul def /s u 7 mul def /n u 9 mul def
/mt u -3 mul def /mv u -5 mul def /ms u -7 mul def /mn u -9 mul def
mn mv moveto n mv lineto n v lineto mn v lineto closepath fill
mn  u moveto
ms  u lineto
ms  t lineto
mv  t lineto
mv  u lineto
mt  u lineto mt v lineto
mt  u moveto
 t  u lineto
 t  t lineto
 u  t lineto
 u  v lineto
 t  u moveto
 t mu lineto
 v mu lineto
 v  t lineto
 s  t lineto
 s  v lineto
 s mv lineto
 v mv moveto
 v mt lineto
 t mt lineto
 t mu lineto
 u mv moveto
 u mu lineto
mu mu lineto
mu  u lineto
mu mv moveto
mu mt lineto
mv mt lineto
mv mu lineto
mt mu lineto
mt  u lineto
mv mt moveto
ms mt lineto
ms mu lineto
mn mu lineto
2 setlinejoin 1 setgray 2 setlinewidth
stroke end}def % end
%%EndProlog
pentominoes showpage
%%EOF
```



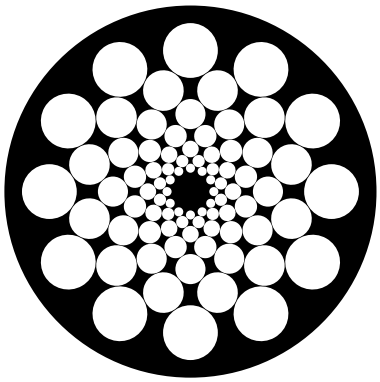
Hexpuzzle

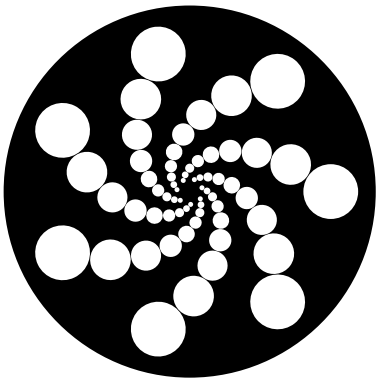
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Hexpuzzle. cgl 1997
%%BoundingBox: -103 -100 103 103
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/hexpuzzledict 4 dict def
/hexpuzzle{hexpuzzledict begin %tilvi
/r 103 def
r 0 moveto 0 0 r 0 360 arc fill
/r 100 def /rh r 2 div def /rq r 4 div def
/top r .866 mul def /toph top 2 div def
r 0 moveto 6{60 rotate r 0 lineto}repeat
0 top moveto
rq toph lineto
rq neg toph lineto
0 0 lineto
rq neg toph neg lineto
0 top neg lineto
r neg 0 moveto rh 0 lineto
rh rq add toph moveto
rq toph neg lineto
rh top neg lineto
rq neg toph moveto
rh neg 0 lineto
2 setlinejoin 1 setgray 2 setlinewidth stroke
end}def%hexpuzzle
%%EndProlog
hexpuzzle showpage
%%EOF
```

TIMTOWTDI

Circle tiles

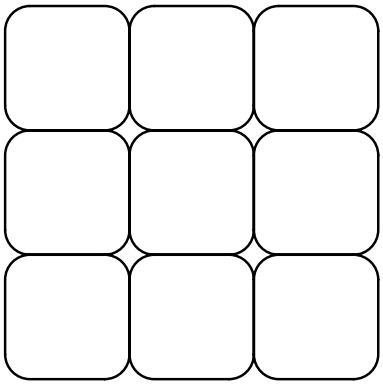
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Circle tiles. cgl 1997
%%BoundingBox: -67 -67 217 67
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/circletilesdict 10 dict def
/circletiles{circletilesdict begin %tilxxxviii
1 setlinejoin
/r 50 def
/hs r 15 sin mul def /s 2 hs mul def
/ring{12{gsave
         r 0 translate
         circle
         grestore
         30 rotate
        }repeat
}def
/circle{.75 hs mul 0 moveto
  0 0 .75 hs mul 0 360 arc
  closepath fill
}def
%
/rs r hs add 3 add def /mrs rs neg def
/frame{rs 0 moveto
   0 0 rs 0 360 arc
   closepath
}def
/f r hs sub r div def
%
gsave
frame 0 setgray fill
1 setgray
7{ring
  f f scale
  15 rotate
 }repeat
grestore
%
150 0 translate
/ring{7{gsave
         r 0 translate
         circle
         grestore
         360 7 div rotate
        }repeat
}def
frame 0 setgray fill
1 setgray
9{ring f f scale 15 rotate
 }repeat
end}def
%%EndProlog
circletiles showpage
%%EOF
```



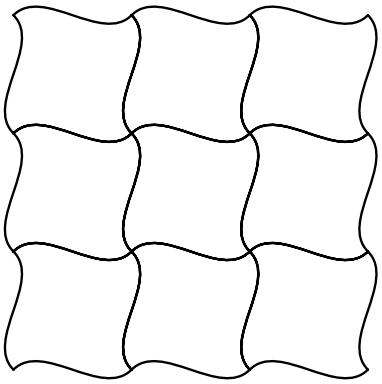


9 squares

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: 9 squares. cgl 1997
%%BoundingBox: -76 -76 76 76
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/9squaresdict 6 dict def
/9squares{9squaresdict begin %tilxii
0 setlinejoin 1 setlinecap
/a 50 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/r 10 def
/tile{4{ha r sub mha moveto
  mha mha
  mha mha r add
  r arcto
    90 rotate}repeat
}def
ma a a{/i exch def
ma a a{/j exch def
   gsave i j translate
         tile stroke grestore
}for}for
end}def%9squares
%%EndProlog
9squares showpage
%%EOF
```

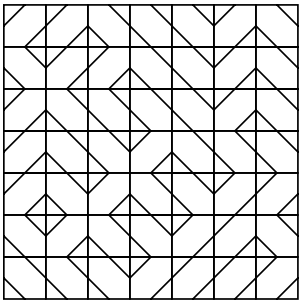


9 squares II



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: 9 squares II. cgl 1997
%%BoundingBox: -80 -80 80 80
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/9squaresiidict 9 dict def
/9squaresii{9squaresiidict begin %tilxxvii
0 setlinejoin 1 setlinecap
/a 50 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/qa a 4 div def /mqa qa neg def
/tile{4{mha ha moveto
  mqa ha qa add qa qa ha ha curveto
  90 rotate}repeat
}def
ma a a{/i exch def
ma a a{/j exch def
   gsave i j translate
         tile stroke grestore
}for}for
end}def
%%EndProlog
9squaresii showpage
%%EOF
```

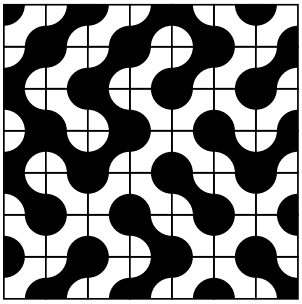
Truchet tiling



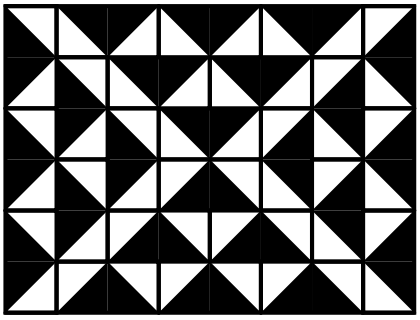
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Truchettiling, cgl 1997
%%BoundingBox: -90 -90 90 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/truchettilingdict 9 dict def
/truchettiling{truchettilingdict begin %tilLxvi
0 setlinejoin 1 setlinecap
/a 25 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/tile{rand dup 2 idiv 2 mul eq {90 rotate}if
   filledsquare
}def
/filledsquare{mha mha moveto ha mha lineto
  ha  ha lineto mha ha lineto
  closepath  stroke
  0 ha moveto ha 0 lineto
  0 mha moveto mha 0 lineto
  1 setlinewidth stroke
}def
/dotiling{f ma mul a f a mul{/i exch def
          f ma mul a f a mul{/j exch def
             gsave i j translate
             tile stroke grestore
          }for}for
}def
/f 3 def 5 srand dotiling
end}def
%%EndProlog
truchettiling showpage
%%EOF
```

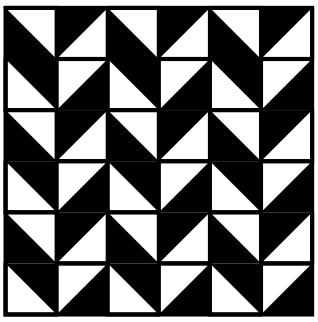
Truchet tiling II

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Truchettiling II, cgl 1997
%%BoundingBox: -90 -90 90 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/truchettilingIIdict 15 dict def
/truchettilingII{truchettilingIIdict begin %tilLxvii
0 setlinejoin 1 setlinecap
/a 25 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/sq{mha mha moveto ha mha lineto
  ha  ha lineto mha ha lineto closepath
}def
/tile{gsave flipflop{oneofi}{oneofii}ifelse grestore
      sq stroke
}def
/arcs{ha  ha moveto  ha  ha ha 180 270 arc closepath
     mha mha moveto mha mha ha   0  90 arc closepath
}def
/oneofi{rand dup 2 idiv 2 mul eq
  {90 rotate /gl 0 def}
  {sq fill /gl 1 def}ifelse
  arcs gl setgray fill
}def
/oneofii{rand dup 2 idiv 2 mul eq
  {90 rotate sq fill /gl 1 def}
  {/gl 0 def}ifelse
  arcs gl setgray fill
}def
/dotiling{/flipflop true def
  f ma mul a f a mul{/i exch def
  f ma mul a f a mul{/j exch def
   gsave i j translate tile stroke grestore
   /flipflop flipflop not def
  }for}for
}def
/f 3 def 10 srand dotiling
end}def
%%EndProlog
truchettilingII showpage
%%EOF
```



Douat parquets

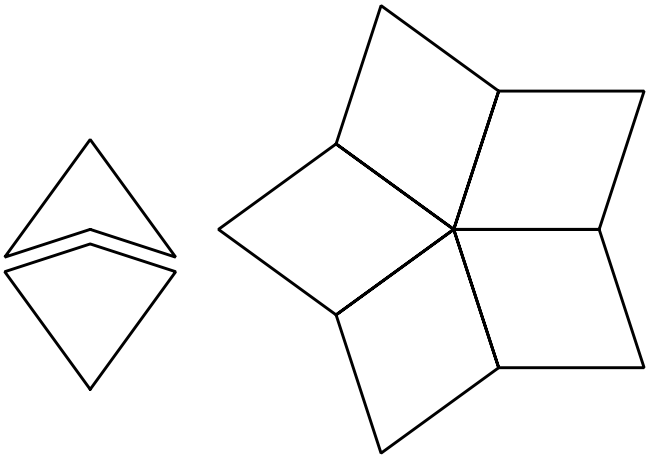


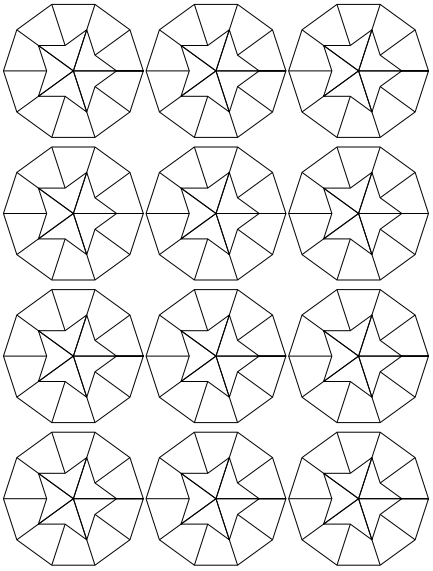


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Douatparquets, cgl 1997
%%BoundingBox: -55 -40 55 145
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/douatparquetsdict 18 dict def
/douatparquets{douatparquetsdict begin %tilLxxi
0 setlinejoin 1 setlinecap
/a 25 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/qa ha 2 div def /mqa qa neg def /f 1 def
/ll{gsave qa mqa moveto
    mqa mqa lineto
    mqa  qa lineto closepath fill
     qa mqa moveto
     qa  qa lineto
    mqa  qa lineto stroke
    grestore
}def
/lr{gsave  90 rotate ll grestore}def
/ur{gsave 180 rotate ll grestore}def
/ul{gsave -90 rotate ll grestore}def
/tile{gsave mqa  qa translate ll grestore
      gsave mqa mqa translate ur grestore
      gsave  qa mqa translate lr grestore
      gsave  qa  qa translate ul grestore
}def
f ma mul a f a mul{/i exch def
f ma mul a f a mul{/j exch def
   gsave i j translate
         tile grestore
}for}for
%frame
/r f a mul ha add def /mr r neg def
 r mr moveto r  r lineto mr  r lineto mr mr lineto closepath strok
0 4 a mul translate
gsave
2{gsave 2{gsave
qa 5 qa mul translate ul
ha        0 translate ur
ha        0 translate ul
ha        0 translate lr
grestore gsave
qa 3 qa mul translate ur
ha         0 translate ul
ha         0 translate lr
ha         0 translate ur
grestore gsave
qa qa translate ul
ha  0 translate lr
ha  0 translate ur
ha  0 translate lr
grestore
-1 1 scale}repeat
grestore 1 -1 scale}repeat
grestore
%frame
/ri 4 ha mul def /mri ri neg def
/rj 3 ha mul def /mrj rj neg def
 ri mrj moveto
 ri  rj lineto
mri  rj lineto
mri mrj lineto
closepath stroke
end}def
%%EndProlog
douatparquets showpage
%%EOF
```

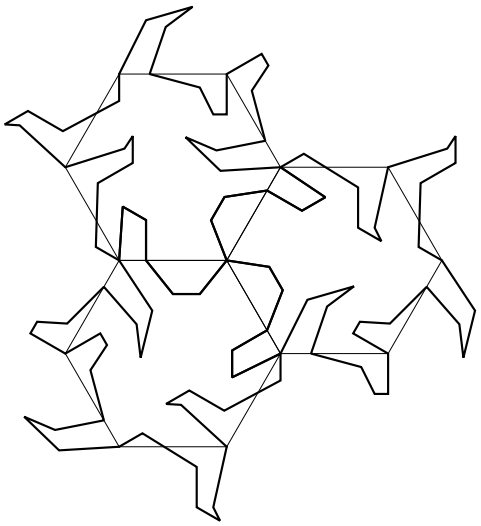
Penrose tile

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Penrose, cgl 1997
%%BoundingBox: -35 -23 195 133
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/penrosedict 10 dict def
/penrose{penrosedict begin%tilLxviii
%Er is betere versie van, aug 2012
/r 50 def  /phi .618 def 2 setlinejoin
/turtle{%direction and stepsize on stack
/step exch def
/dir exch def
  currentpoint translate
     dir rotate
     step 0 lineto
}def
/phir phi r mul def
/kite{0 0 moveto
      54    r turtle
      108 phir turtle
      36 phir turtle
      108    r turtle
      54    0 turtle
}def
/dart{0 0 moveto
      -18 phir turtle
      144    r turtle
      108    r turtle
      144 phir turtle
      -18    0 turtle
}def
/diamond{0 0 moveto
      0 r turtle
      72 r turtle
      108 r turtle
      72 r turtle
      108 0 turtle
}def
%
gsave kite stroke grestore
0 1.1 r mul translate
gsave dart stroke grestore
2.5 r mul 0 translate
5{diamond
  72 rotate}repeat
stroke
end}def
%%EndProlog
penrose showpage
%%EOF
```





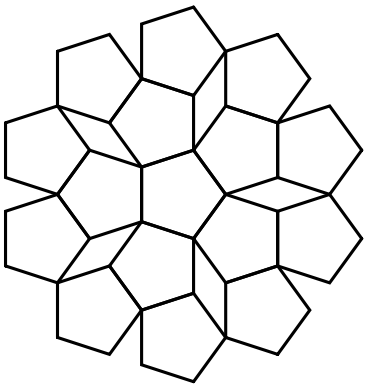
Escher's frogs



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: 3 frogs Escher. cgl 1997
%%BoundingBox: -81 -81 146 165
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/3frogsescherdict 20 dict def
/3frogsescher{3frogsescherdict begin %tilxiv
0 setlinejoin 1 setlinecap
/r  50 def            /mr r neg def
/hr r 2 div def       /mhr hr neg def
/qr r 4 div def       /mqr qr neg def
/er r 8 div def       /mer er neg def
/hrx hr 1.732 mul def /mhrx hrx neg def
/qrx hrx 2 div def    /mqrx qrx neg def
/erx qrx 2 div def    /merx erx neg def
/delta er 2 div def
/head{hr 0 moveto
      qr mqr 1.25 mul lineto
      0  mqr 1.25 mul lineto
      mqr 0 lineto
      mqr qr 1.5 mul lineto
      merx er rlineto
      mhr 0 lineto
      stroke}def
/side{hr 0 moveto
   mer 2 div delta sub
   mhr delta sub  rlineto
   delta mer rlineto
   merx er  rlineto
   0 hr er sub rlineto
   mhr erx add er lineto
   mhr 0 lineto stroke}def
/tail{hr 0 moveto
   qr mhr  lineto
   mqrx mer  rlineto %end
   qr er 1.5 mul  rlineto
   erx 0 lineto
   mqr er lineto
   mer qr rlineto
   mer 0 rlineto
   mhr 0 lineto stroke}def
/grid{gsave r 0 moveto
      6{60 rotate r 0 lineto}repeat
      .05 setlinewidth stroke grestore}def
grid
/reptile{
gsave 0 hrx translate head grestore
gsave hr qr add qrx translate
      120 rotate head grestore
gsave 0 mhrx translate
      side grestore
gsave hr qr add mqrx translate
      -120 rotate side grestore
gsave mhr mqr add qrx translate
      60 rotate tail grestore
gsave mhr mqr add mqrx translate
      -60 rotate tail grestore}def
reptile
gsave r hr add hrx translate 120 rotate grid reptile grestore
gsave 0 hrx 2 mul translate -120 rotate grid reptile grestore
end }def %3frogsescher
%%EndProlog
3frogsescher showpage
%%EOF
```

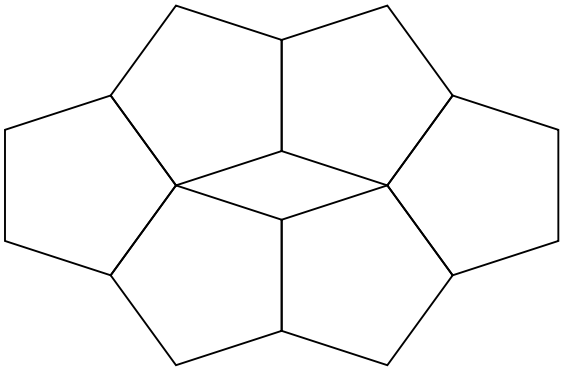
Pentagons with spurious diamonds

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pentagons. cgl 1997
%%BoundingBox: -60 -61 60 61
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/pentagonsdict 5 dict def
/pentagons{pentagonsdict begin %tilxv
0 setlinejoin 1 setlinecap
/r 15 def
/s 2 r mul 54 cos mul def
/rin r 54 sin mul def
/p{gsave r 0 moveto
   5{72 rotate r 0 lineto}repeat
   stroke grestore
}def
/tr{2 rin mul 0 translate}def
p 36 rotate
5{gsave tr
  p gsave  36 rotate tr p grestore
    gsave -36 rotate tr p grestore
  grestore 72 rotate}repeat
end}def %pentagons
%%EndProlog
pentagons showpage
%%EOF
```



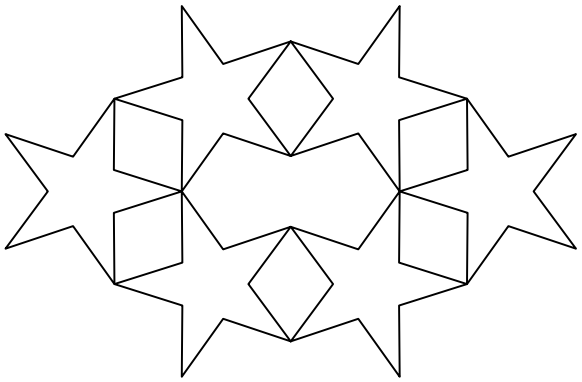
Pentagons with spurious diamonds II

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lips. cgl 1997
%%BoundingBox: -155 -100 155 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/lipsdict 9 dict def
/lips{lipsdict begin%tilxxxiii
0 setlinejoin 1 setlinecap
/r 50 def /mr r neg def
/rin r 36 cos mul def
/hs r 36 sin mul def
/s 2 hs mul def
/hdy s 18 sin mul def /mhdy hdy neg def
/hdx s 18 cos mul def /mhdx hdx neg def
/pentagon{r 0 moveto
  5{72 rotate r 0 lineto}repeat
stroke}def
%
/figure{pentagon}def
2{gsave mhdx mr add 0 translate
  figure
  2{gsave 36 rotate 2 rin mul 0 translate
        figure
    grestore 1 -1 scale
   }repeat
  grestore -1 1 scale
 }repeat
end}def
%%EndProlog
lips showpage
%%EOF
```



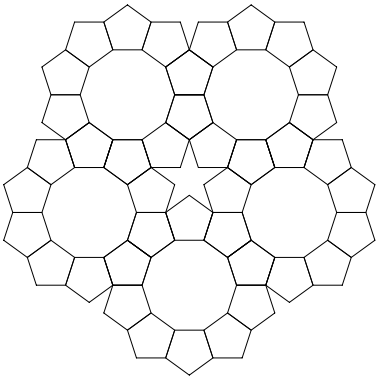
Lips with stars

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lips with stars. cgl 1997
%%BoundingBox: -155 -97 155 97
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/lipswithstarsdict 17 dict def
/lipswithstars{lipswithstarsdict begin %tilLiii
1 setlinejoin 1 setlinecap
/r 50 def /mr r neg def
/rin r 36 cos mul def
/hs r 36 sin mul def
/s 2 hs mul def
/hdy s 18 sin mul def /mhdy hdy neg def
/hdx s 18 cos mul def /mhdx hdx neg def
/pentagon{r 0 moveto
  5{72 rotate r 0 lineto}repeat
stroke}def
/star % n r on stack, n>=5
{/rstar exch def
 /nstar exch def
 /alfahstar 180 nstar div def
 /rinstar rstar 2 div alfahstar cos mul 1.5 sub def
 gsave -90 rotate 0 rstar moveto
 nstar{alfahstar rotate
     0 rinstar lineto
     alfahstar rotate
     0 rstar lineto
  }repeat stroke
 grestore
}def
%
/figure{5 r star}def
2{gsave mhdx mr add 0 translate
  figure
  2{gsave 36 rotate 2 rin mul 0 translate
         figure
    grestore 1 -1 scale
   }repeat
  grestore -1 1 scale
 }repeat
end}def
%%EndProlog
lipswithstars showpage
%%EOF
```



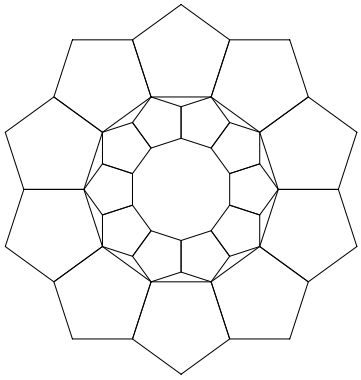
Nontight Pentagons

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nontight pentagons. cgl 1997
%%BoundingBox: -200 -210 200 190
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nontightpentagonsdict 13 dict def
/nontightpentagons{nontightpentagonsdict begin %tilxxx
0 setlinejoin 1 setlinecap
/r 50 def
/pentagon{rp 0 moveto
  5{72 rotate rp 0 lineto}repeat
stroke}def
/tile{10{gsave xp 0 translate
         pentagon
   grestore
   36 rotate}repeat
}def
/hs r 18 sin mul def  /mhs hs neg def
/s 2 hs mul def
/rp hs 36 sin div def
/rip rp 36 cos mul def
/xp r 18 cos mul rip add def
/hdia rp 72 sin mul def
/rstar  hdia 36 sin div def
%
/shift r 18 cos mul
   s 72 sin mul add
  rstar 36 cos mul add
def
%
-18 rotate
5{gsave
  shift 0 translate
  tile
  grestore
  72 rotate
 }repeat
end}def
%%EndProlog
nontightpentagons showpage
%%EOF
```



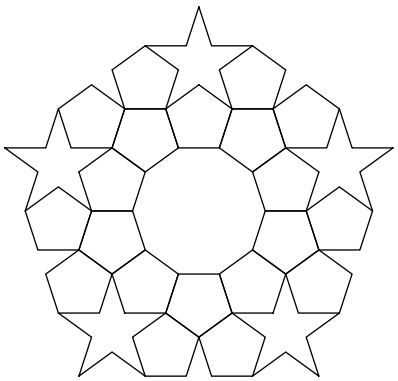
Nontight Pentagons II

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nontight pentagons II. cgl 1997
%%BoundingBox: -175 -185 175 185
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nontightpentagonsiidict 13 dict def
/nontightpentagonsii{nontightpentagonsiidict begin%tilxxxi
0 setlinejoin 1 setlinecap
/r 50 def
/pentagon{rp 0 moveto
  5{72 rotate rp 0 lineto}repeat
stroke}def
/tile{10{gsave xp 0 translate
         pentagon
   grestore
   36 rotate}repeat
}def
%
2{/hs r 18 sin mul def  /mhs hs neg def
/hss 1.732 hs mul def
/s 2 hs mul def
/rp hs 36 sin div def
/rip rp 36 cos mul def
/xp r 18 cos mul rip add def
tile
18 rotate
/r xp rp add def
}repeat
%gsave
%0 4 r mul translate tile
%0 4 r mul translate tile
%grestore
%4 r mul 0 translate tile
%0 4 r mul translate tile
%0 4 r mul translate tile
%grestore
end}def
%%EndProlog
nontightpentagonsii showpage
%%EOF
```



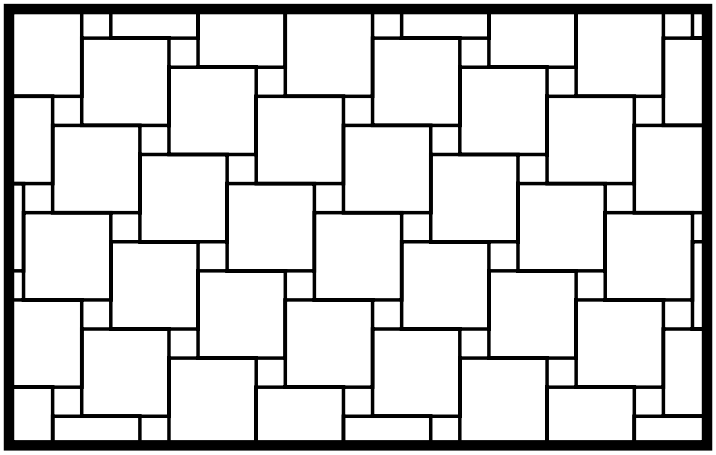
Nontight Pentagons III

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nontight pentagons III. cgl 1997
%%BoundingBox: -155 -130 155 155
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nontightpentagonsiiidict 13 dict def
/nontightpentagonsiii{nontightpentagonsiiidict begin %tilxxxii
0 setlinejoin 1 setlinecap
/r 50 def
/pentagon{rp 0 moveto
  5{72 rotate rp 0 lineto}repeat
stroke}def
/tile{10{gsave xp 0 translate
           pentagon extra
         grestore
         36 rotate
        }repeat
}def
/extra{pents}def
/pents{gsave 36 rotate
            2 rip mul 0 translate
         pentagon
         grestore
         gsave -36 rotate
             2 rip mul 0 translate
         pentagon
       grestore
       /extra{star}def
}def
/star{gsave rp 72 cos mul rstar 36 cos mul add 0 translate
      rstar 0 moveto
      rstar 72 cos mul dup  72 cos 1 sub -72 sin div mul lineto
      rstar 72 cos mul rstar 72 sin mul lineto
      rstar 0 moveto
      rstar 72 cos mul dup  72 cos 1 sub 72 sin div mul lineto
      rstar 72 cos mul rstar -72 sin mul lineto
      stroke grestore
      /extra{pents}def
}def
/hs r 18 sin mul def  /mhs hs neg def
/hss 1.732 hs mul def
/s 2 hs mul def
/rp hs 36 sin div def
/rip rp 36 cos mul def
/xp r 18 cos mul rip add def
/hdia rp 72 sin mul def
/rstar  hdia 36 sin div def
%
-18 rotate tile
end}def
%%EndProlog
nontightpentagonsiii showpage
%%EOF
```



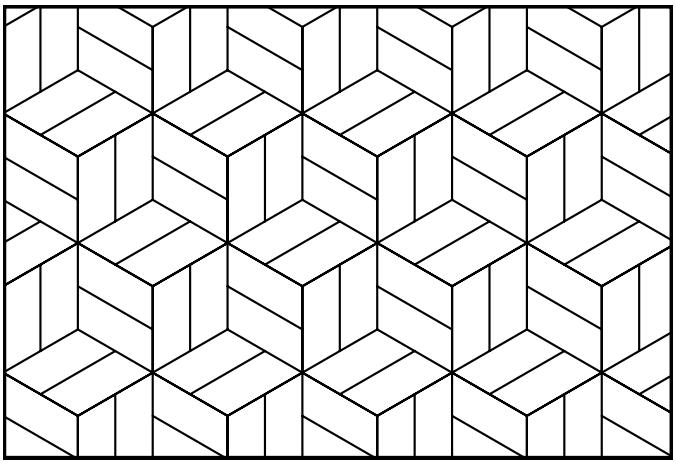
van Doesburg

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Van Doesburg. cgl 1997

%%BoundingBox: 23 -15 227 115
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/vandoesburgdict 8 dict def
/vandoesburg{vandoesburgdict begin %tilxxxv
0 setlinejoin 1 setlinecap
/s 25 def           /ms s neg def
/alfa s 3 div def   /malfa alfa neg def
/hs .5 s mul def    /mhs hs neg def
/sq{hs mhs moveto hs hs lineto
    mhs hs lineto mhs mhs lineto
    closepath stroke
}def
/frame{s mhs moveto 9 s mul mhs lineto
  9 s mul 4.5 s mul lineto
  s       4.5 s mul lineto closepath
}def
gsave frame clip
11{gsave
  11{sq s malfa translate}repeat
  grestore alfa s translate
 }repeat
grestore
frame 3 setlinewidth stroke
end}def
%%EndProlog
vandoesburg showpage
%%EOF
```



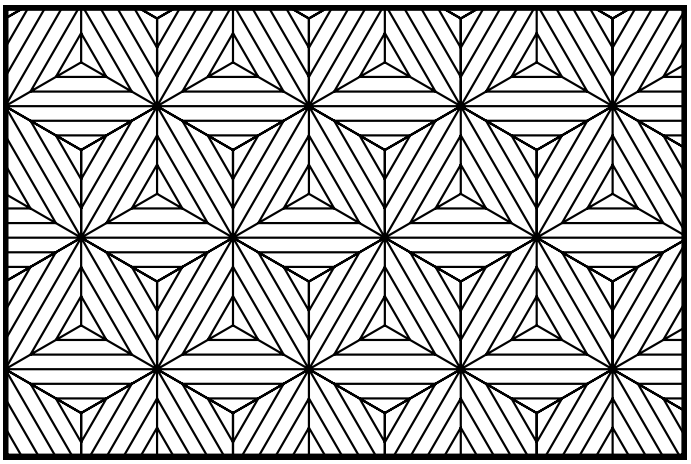
Cubes

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Cubes. cgl 1997
%%BoundingBox: 0 0 310 210
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/cubesdict 12 dict def
/cubes{cubesdict begin %tilxvi
0 setlinejoin
/s 20 def /ms s neg def
/sst s 1.732 mul def
/ts s 2 mul def /mts ts neg def
/hs s 2 div def
/dia{0 0 moveto 0 ts lineto
   sst ms rlineto 0 mts rlineto
   0 s rmoveto 0 s lineto
}def
/tile{gsave
  3{dia 120 rotate}repeat
  stroke grestore
}def
/pattern{gsave
2{gsave
  5{gsave tile
    sst s 3 mul translate
    tile
    grestore
    sst 2 mul 0 translate
   }repeat
  grestore 0 s 6 mul translate
}repeat grestore
}def
/a s 10.5 mul def /ma a neg def
/frame{0 0 moveto
  0 a rlineto
  s 15.5 mul 0 rlineto
  0 ma rlineto
  closepath
}def
gsave frame clip
      pattern
grestore
frame 3 setlinewidth stroke
end}def %cubes
%%EndProlog
cubes showpage
%%EOF
```



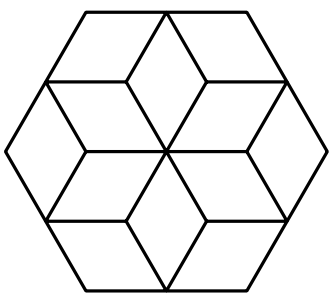
Tiling Cubes

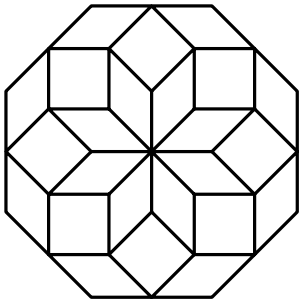
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Tiling cubes. cgl 1997
%%BoundingBox: -1 -1 311 206
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/tilingcubesdict 17 dict def
/tilingcubes{tilingcubesdict begin %tilxxxix
0 setlinejoin 1 setlinecap
/s 20 def /ms s neg def
/sst s 1.732 mul def
/ts s 2 mul def /mts ts neg def
/hs s 2 div def /mhs hs neg def
/hsst hs 1.732 mul def
/qs s 4 div def /mqs qs neg def
/qsst s 1.732 mul def
/dia{3{0 0 moveto 0 ts lineto sst s lineto 0 -2 s mul rlineto
       .333 sst mul 1.667 s mul moveto sst -.333 s mul lineto
       .667 sst mul 1.333 s mul moveto sst  .333 s mul lineto
       120 rotate
      }repeat
}def
/triangle{/f exch ts mul def
   0 f moveto
   3{120 rotate 0 f lineto}repeat
}def
/tile{gsave dia
  1    triangle
  .667 triangle
  .333 triangle
  stroke grestore
}def
/pattern{gsave
3{gsave
  5{gsave tile
    sst s 3 mul translate
    tile
    grestore
    sst 2 mul 0 translate
   }repeat
  grestore 0 s 6 mul translate
}repeat grestore
}def
/a s 10.25 mul def /ma a neg def
/frame{0 0 moveto
  0 a rlineto
  s 15.5 mul 0 rlineto
  0 ma rlineto
  closepath
}def
gsave frame clip
      pattern
grestore
frame 3 setlinewidth stroke
end}def
%%EndProlog
tilingcubes showpage
%%EOF
```



Hexagons

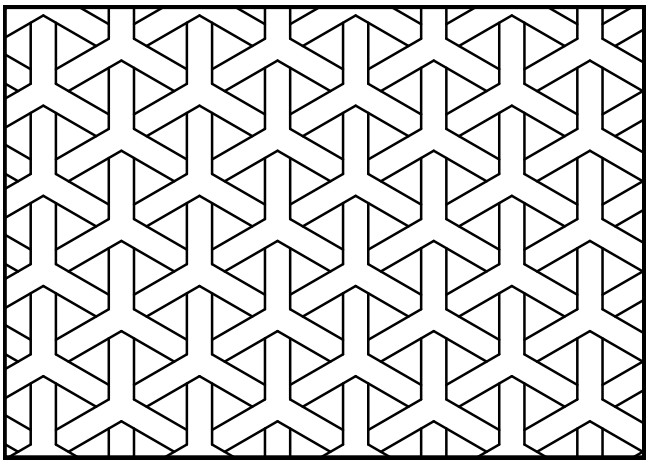
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Hexagons. cgl 1997
%%BoundingBox: -55 -47 175 47
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/hexagonsdict 3 dict def
/hexagons{hexagonsdict begin %tilxxxvi
0 setlinejoin 1 setlinecap
/turtle{%direction and stepsize on stack
  /step exch def /dir exch def
  currentpoint translate
  dir rotate step 0 lineto
}def
/r 25 def 2 setlinejoin
6{r 0 moveto -60 r turtle
             120 r turtle
              60 r turtle
             120 r turtle
             -60 r turtle
            -120 0 turtle
 }repeat stroke
%
125 0 translate /r .75 r mul def
8{r 0 moveto 45 r turtle
            -90 r turtle
            135 r turtle
             45 r turtle
            135 r turtle
            -90 r turtle
             45 r turtle
           -180 0 turtle
 }repeat stroke
end}def
%%EndProlog
hexagons showpage
%%EOF
```





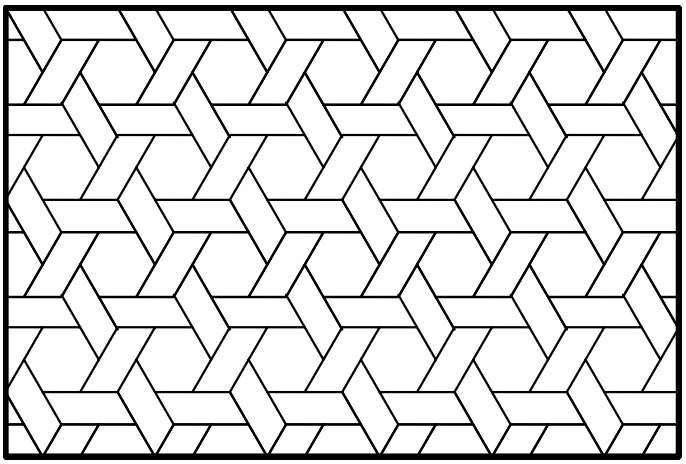
Fence

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Fences. cgl 1997
%%BoundingBox: 0 0 255 180
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/fencedict 16 dict def
/fence{fencedict begin %tilxvii
0 setlinejoin 1 setlinecap
/s 30 def /ms s neg def /ss s 1.732 mul def
/hs s 2 div def /mhs hs neg def
    /hss hs 1.732 mul def
/w  s 3 div def /wds w 1.732 div def
/hw w 2 div def /hwds hw 1.732 div def
/line{hw s hwds sub moveto
      hw hwds lineto
}def
/tile{3{gsave line -1 1 scale line stroke
        grestore 120 rotate}repeat
}def
/pattern{gsave
6{gsave
  7{gsave tile
    hw hss add hs hwds add translate
    tile
    grestore
    w ss add 0 translate
   }repeat
  grestore 0 s wds add translate
}repeat grestore
}def
/a s 6 mul def /ma a neg def
/frame{0 0 moveto
  0 a rlineto
  s 8.5 mul 0 rlineto
  0 ma rlineto
  closepath
}def
gsave frame clip newpath
      mhs mhs translate pattern
grestore
frame 3 setlinewidth stroke
end}def %fence
%%EndProlog
fence showpage
%%EOF
```



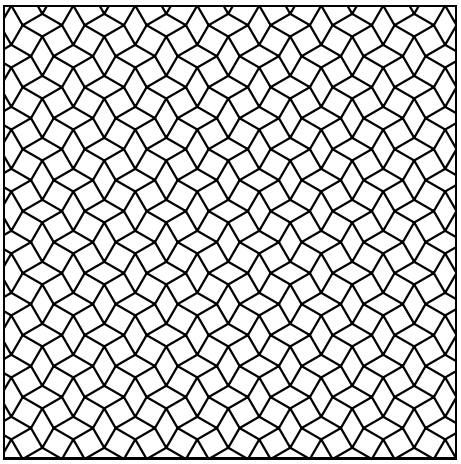
Fence II

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Fence II cgl 1997
%%BoundingBox: -101 -101 201 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/fenceIIdict 17 dict def
/fenceII{fenceIIdict begin %tilxxvi
/a 50 def       /ma a neg def
/ha a 2 div def /mha ha neg def
/r a 3 div def  /mr r neg def
/hr r 2 div def /mhr hr neg def
/hrt hr 1.73 mul def /mhrt hrt neg def
/element{6{r -1.5 mul mhrt moveto
   mhr hrt lineto
   60 rotate}repeat
}def
/tile{element
 gsave
 ha ha 1.73 mul translate
 element stroke
 grestore
}def
/ta a 2 mul def /mta ta neg def
/frame{mta mta moveto mta  ta lineto
        2 ta mul ta lineto  2 ta mul mta lineto
       closepath}def
gsave
frame clip
mta ta hrt sub ta {/j exch def
mta a 2 ta mul{/i exch def
   gsave i j translate
         tile stroke
   grestore
}for
}for
grestore
frame 3 setlinewidth stroke
end}def
%%EndProlog
fenceII showpage
%%EOF
```



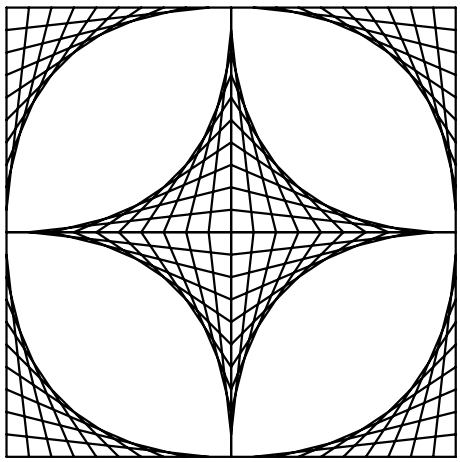
43tile

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lissajous. cgl 1997
%%BoundingBox: -100 -100 100 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/43tiledict dict def
/43tile{43tiledic begin %jalxxiii
/a 10 def
/ha {a .5 mul} def
/tile {% rhombus + 90 rotated rhombus
 ha 3 sqrt mul 0 moveto
 0 ha lineto
 ha 3 sqrt mul neg 0 lineto
 0 ha neg lineto
 closepath
 ha 1 3 sqrt add mul ha 3 sqrt mul lineto
 ha 2 3 sqrt add mul 0 lineto
 ha 1 3 sqrt add mul ha 3 sqrt mul neg lineto
 closepath
} def
/tena {a 10 mul}def
/frame {tena neg tena moveto
 tena 2 mul 0 rlineto
 0 tena -2 mul rlineto
 tena -2 mul 0 rlineto
 closepath
} def
/dotiling {
 a -11 mul tena neg translate
 9{gsave
  11{tile a 1 3 sqrt add mul 0 translate
    }repeat stroke
  grestore
  gsave ha 1 3 sqrt add mul dup translate
  11{tile a 1 3 sqrt add mul 0 translate
     }repeat stroke grestore
  0 a 1 3 sqrt add mul translate
 }repeat
end} def
%
%tile
%
gsave frame clip
%gsave
dotiling
grestore
}def
%%EndProlog
43tile showpage
%%EOF
```



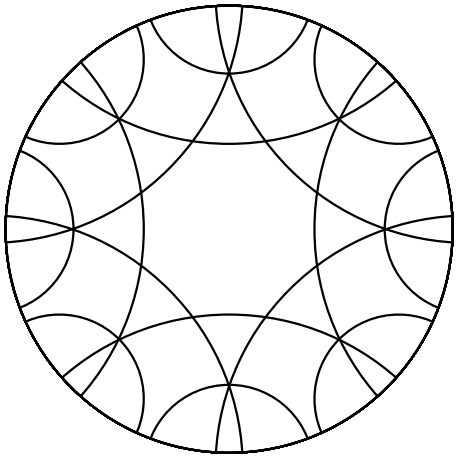
Envelope

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Envelope. cgl 1997
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/envelopdict 9 dict def
/envelope{envelopdict begin %tilxviii
0 setlinejoin 1 setlinecap
/r 100 def /n 5 def /mn n neg def
/hr .5 r mul def /mhr hr neg def
/h hr n div def
/tile{2{mn 1 n{/y exch h mul def
        hr y moveto y neg hr lineto}for
        180 rotate}repeat
}def
mhr r hr{/i exch def
mhr r hr{/j exch def
   gsave i j translate
      i j mul 0 lt{90 rotate}if
      tile stroke
   grestore
}for}for
end}def %envelope

%%EndProlog
envelope showpage
%%EOF
```



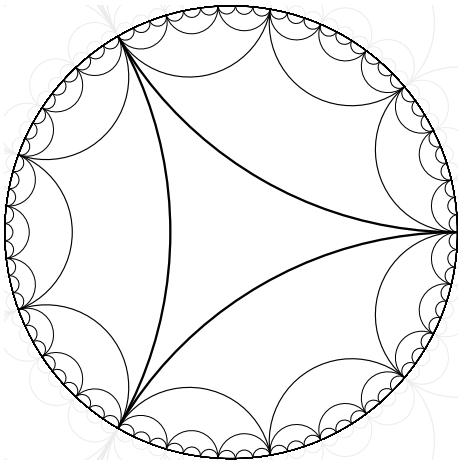
Circle grid

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Circle grid. cgl 1997
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/circlegriddict 8 dict def
/circlegrid{circlegriddict begin %tilxx
0 setlinejoin 1 setlinecap
/R 100 def /tR 2 R mul def
/a .38 R mul def
/m .5 R R a div mul a add mul def
/r m a sub def /tr 2 r mul def
/circle{%x y r on stack
   /rad exch def
   /y exch def /x exch def
   x y translate
   rad 0 moveto
   0 0 rad 0 360 arc
}def
0 0 R circle clip
%
2{8{gsave m 0 r circle stroke grestore
         45 rotate
   }repeat
   /r r 3 div def
   /m R R mul r r mul add sqrt def
 }repeat
0 0 R circle 2 setlinewidth stroke
end}def
%%EndProlog
circlegrid showpage
%%EOF
```

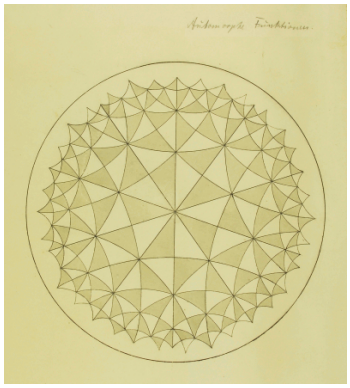


Circle limit grid

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: circlelimitsgrid, cgl 1997
%%BoundingBox: -100 -100 100 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run % contains growthspiral
/circlelimitsgriddict 17 dict def
/circlelimitsgrid{circlelimitsgriddict begin %tilLxiv
0 setlinejoin 1 setlinecap
/R 100 def /levels 4 def
/circle{%x y r on stack
   /radius exch def /y exch def /x exch def
   x y translate
   radius 0 moveto 0 0 radius 0 360 arc
}def
/ring{/n exch def %n on stack
 /ang 360 n div def
 /r R .5 ang mul sin .5 ang mul cos div mul def
 /m R R mul r r mul add sqrt def
 gsave -.5 ang mul rotate
 n{gsave m 0 r circle stroke grestore
   ang rotate}repeat
 grestore
}def
gsave .925 setgray
/n 3 def /wl 1 def
levels{wl setlinewidth
       n ring /n 3 n mul def /wl .5 wl mul def}repeat
grestore
%
0 0 R circle clip
/n 3 def /wl 1 def
levels{wl setlinewidth
       n ring /n 3 n mul def /wl .5 wl mul def}repeat
0 0 R circle 1 setlinewidth stroke
end}def
%%EndProlog
circlelimitsgrid showpage
%%EOF
```

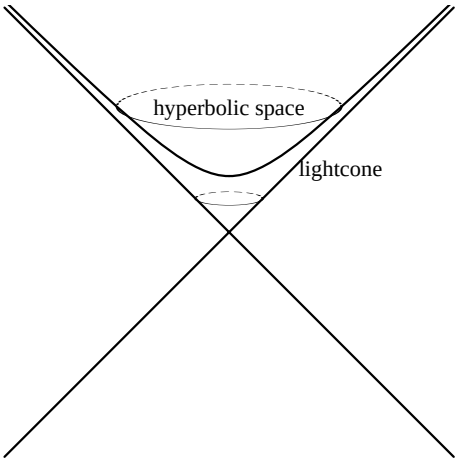


Even Klein already worked on it



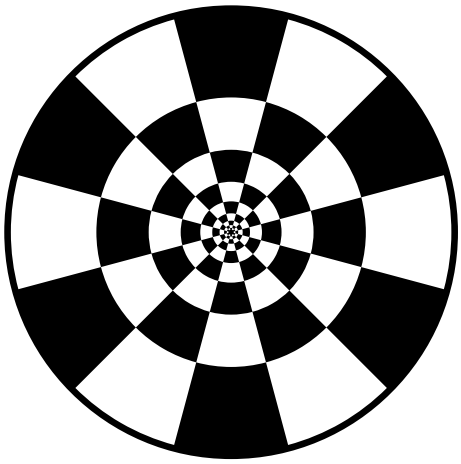
Minkowski model of hyperbolic space

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Minkovski model hyperbolic space. cgl 2014
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/minkowskidict 5 dict def
/minkowski{minkowskidict begin /Times-Roman 10 selectfont
-100 -100 moveto 100 100 lineto
100 -100 moveto -100 100 lineto
/a 25 def
a 6 add  a moveto (lightcone) show
0 52 moveto (hyperbolic space) centershow
0 a moveto
1 1 100{/x exch def  x a a mul x x mul add sqrt lineto}for
0 a moveto
1 1 100{/x exch def  x neg a a mul x x mul add sqrt lineto}for
stroke
gsave 0 15 translate
newpath 1 .2 scale
0 0 15 180 360 arc stroke
[3]0 setdash
0 0 15 0 -180 arc stroke
grestore
0 50 dup mul a a mul add sqrt translate
newpath 1 .2 scale
0 0 50 180 360 arc stroke
gsave [3]0 setdash
0 0 50 0 -180 arc stroke
grestore
end}def
%%EndProlog
minkowski showpage
%%EOF
```



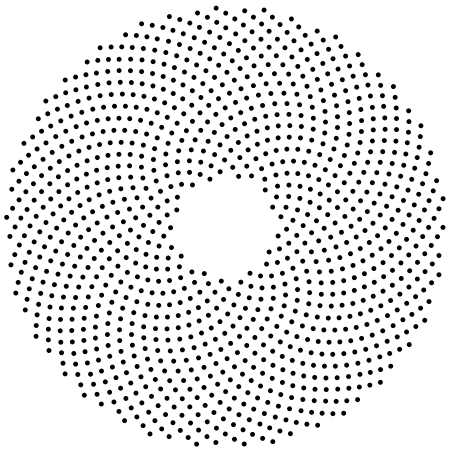
Disk

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Disk. cgl 1997
%%BoundingBox: -103 -103 103 103
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/diskdict 7 dict def
/disk{diskdict begin%tilxL
0 setlinejoin 1 setlinecap
/r 100 def
/square{rin 15 cos mul rin -15 sin mul moveto
%rout 15 cos mul rout -15 sin mul lineto
   0 0 rout -15 15 arc
%   rin 15 cos mul rin 15 sin mul lineto
   0 0 rin 15 -15 arcn
   closepath fill
}def
/ring{/rin rout 1 1.5 15 sin mul sub mul def
      6{gsave square grestore 60 rotate
       }repeat /rout rin def
}def
/frame{rf 0 moveto
   0 0 rf 0 360 arc
   closepath
}def
%
/rout r def
/rf rout 3 add def
frame 0 setgray fill
1 setgray
9{ring 30 rotate}repeat
end}def
%%EndProlog
disk showpage
%%EOF
```



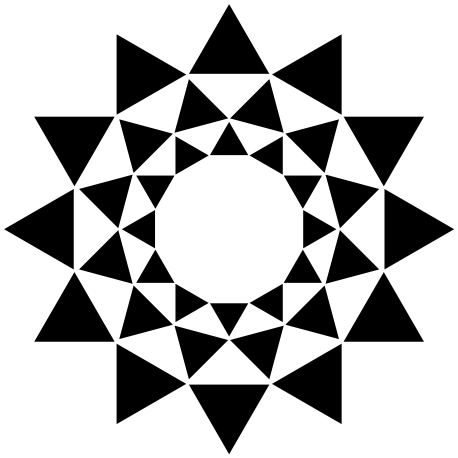
Sunflower pits

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Sunflowerpits. cgl 2009
%%BoundingBox: -95 -92 92 92
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/sunflowerpitsdict 10 dict def
/sunflowerpits{sunflowerpitsdict begin %tilLv
/golden 137.50776 def % super dooper magic number.
/rate 18 360 div def % base spiral expansion rate
/divergence 0.5 def % exponent - 1.0 is linear, 0.5 is best
/sweep 160000 def % total number of degrees in spiral
/inside 60 golden mul def % size of hole in middle
/dotrad 0.2 def % size of seeds
/dotrad 1 def % size of seeds

/dot1 {newpath dotrad 0 360 arc fill} def % dot utility routine
/fixpack {dup 0 ge {divergence exp}
                   {neg % packing utility routine
                    divergence exp neg} ifelse} def
/drawsunflower {inside golden sweep
   { % start for loop
    dup dup rate mul fixpack % copy vector
    exch cos mul exch dup % find & save x position
    rate mul fixpack exch sin mul % find y position
    dot1 } for
} def % draw dot; end loop & proc
drawsunflower % draw the sunflower
end}def
%%EndProlog
sunflowerpits showpage
%%EOF
```



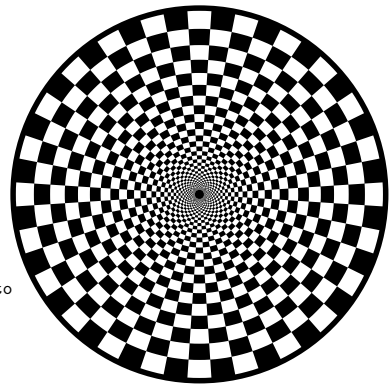
Corona

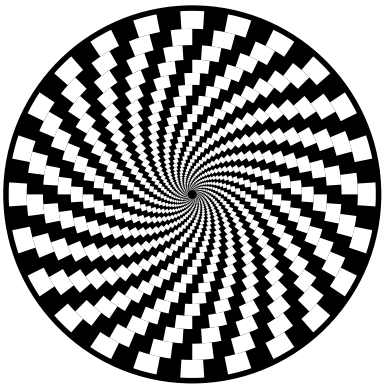
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Corona. cgl 1997
%%BoundingBox: -72.5 -72.5 72.5 72.5
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/coronadict 6 dict def
/corona{coronadict begin %tilxLii
0 setlinejoin 1 setlinecap
/r 50 def %BB: r(1+1.732 sin 15)
/hs r 15 sin mul def  /mhs hs neg def
/hss 1.732 hs mul def
/f r r hss add div def
/tri{hss 0 moveto
     0 hs lineto 0 mhs lineto
     closepath fill
}def
3{12{gsave r 0 translate tri grestore
     30 rotate
    }repeat
  f f scale
  15 rotate
 }repeat
end}def
%%EndProlog
corona showpage
%%EOF
```



Disks

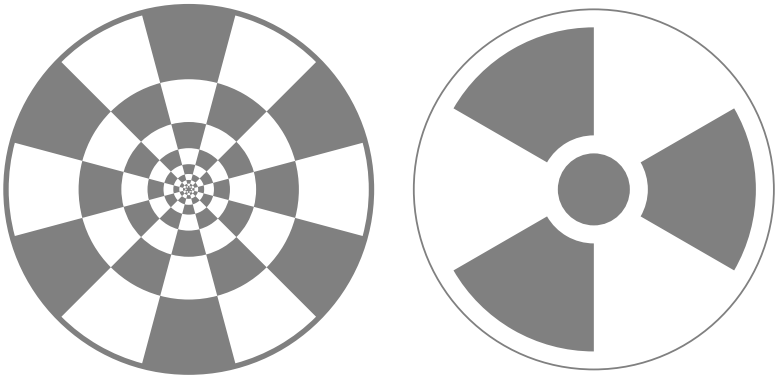
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Disks. cgl 1997
%%BoundingBox: -105 -103 355 103
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/disksdict 7 dict def
/disks{disksdict begin %tilxLi
0 setlinejoin
1 setlinecap
/r 100 def
/square{rin 3.75 cos mul rin -3.75 sin mul moveto
   0 0 rout -3.75 3.75 arc
   0 0 rin 3.75 -3.75 arcn fill
}def
/ring{/rin rout 1 1.5 3.75 sin mul sub mul def
      24{gsave square grestore 15 rotate
        }repeat /rout rin def
}def
/frame{rf 0 moveto
   0 0 rf 0 360 arc
}def
%
/rout r def
/rf rout 3 add def
gsave
frame 0 setgray fill
1 setgray
36{ring 7.5 rotate}repeat
grestore

250 0 translate
/rout r def
/rf rout 3 add def
frame 0 setgray fill
gsave
1 setgray
36 {ring 3.75 rotate}repeat
grestore
end}def
%%EndProlog
disks showpage
%%EOF
```





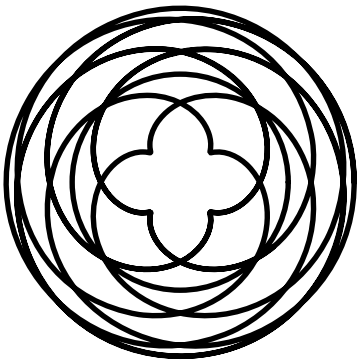
Disks ii

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Disksii. cgl 1997
%%BoundingBox: -105 -105 330 105
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/disksiidict 12 dict def
/disksii{disksiidict begin%tilL
0 setlinejoin 1 setlinecap
/r 100 def
/sector{rin 15 cos mul rin -15 sin mul moveto
   0 0 rout -15 15 arc
   0 0 rin 15 -15 arcn
   closepath fill
}def
/ring{/rin rout 1 1.5 15 sin mul sub mul def
      6{gsave sector grestore 60 rotate
       }repeat /rout rin def
}def
/frame{rf 0 moveto
   0 0 rf 0 360 arc
   closepath
}def
%
gsave
/rout r def
/rf rout 3 add def
frame 0.5 setgray fill
1 setgray
9{ring 30 rotate}repeat
grestore
225 0 translate
/rout r 10 sub def /rin rout 3 div def
/rf r def
frame .5 setgray stroke
/sector{rin 30 cos mul rin -30 sin mul moveto
   0 0 rout -30 30 arc
   0 0 rin 30 -30 arcn
   closepath .5 setgray fill
}def
3{gsave sector grestore 120 rotate}repeat
rin 10 sub 0 moveto
0 0 rin 10 sub 0 360 arc .5 setgray fill
end}def
%%EndProlog
disksii showpage
%%EOF
```

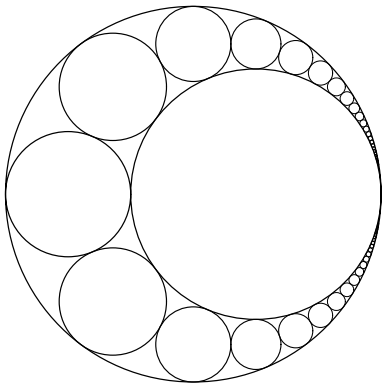


Epicycle

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Epicycle. cgl 2009
%%BoundingBox: -35 -35 35 35
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/epicycledict 3 dict def
/epicycle%a (rational) r (radius smaller circle) ==> epicycle
{epicycledict begin /r exch def/a exch def
 1 r add s 0 moveto
 1 dup 3600{/t exch def
   t cos r a t mul cos mul add s
   t sin r a t mul sin mul add s lineto}for
end}bind def
%
% Program ---the script---
%
/s {20 mul} def
0 0 1 s 0 360 arc stroke
11 7 div  .61 epicycle stroke showpage
%%EOF
```

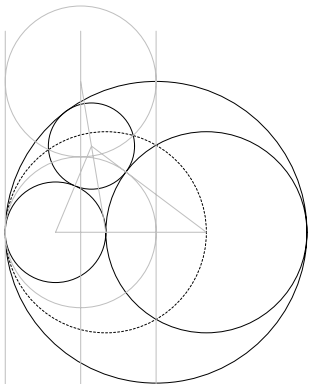


Circle covered by circles



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: circlecoveredbycircles, cgl 1997
%%BoundingBox: -151 -151 151 151
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/circlecoveredbycirclesdict 20 dict def
/circlecoveredbycircles{circlecoveredbycirclesdict begin%tillxix
0 setlinejoin 1 setlinecap
%Initial
/R 100 def /r 50 def /Rr R r add def
/xi R neg def /yi 0 def /ri r def
%Circles
 Rr 0 moveto          r 0  R 0 360 arc
 Rr 0 moveto          0 0 Rr 0 360 arc
  r neg 0 moveto    xi yi  r 0 360 arc
 stroke
%tangent circle: d, x, y
/x{Rr d 2 R mul r div 1 add mul sub}def
/y{Rr d sub dup mul x dup mul sub sqrt}def
/fd{ri d add
    x xi sub dup mul
    y yi sub dup mul add sqrt sub}def
/solveit{%invariant: l fd > 0 and u fd <= 0
  /d .5 l u add mul def
  fd 0 lt{/l d def}
         {/u d def}ifelse
  u l sub eps gt{solveit}
                {/d .5 l u add mul def}ifelse
}def
/eps .01 def /lw 1 def
25{/l .25 ri mul def /u ri def solveit
   x d add y     moveto x y     d 0 360 arc
   x d add y neg moveto x y neg d 0 360 arc
   /lw lw .035 sub def lw setlinewidth stroke
   /ri d def /xi x def /yi y def
}repeat
end}def
%%EndProlog
circlecoveredbycircles showpage
%%EOF
```

Circle covered by circles schema

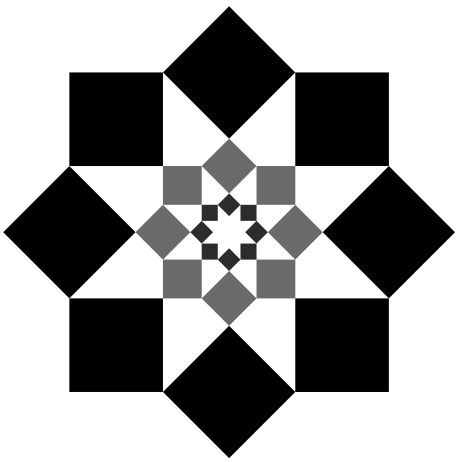


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: circlecoveredbycirclesschema, cgl 1997
%%BoundingBox: -101 -150 201 226
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/circlecoveredbycirclesschemadict 20 dict def
/circlecoveredbycirclesschema{circlecoveredbycirclesschemadict begin %tilLxx
%can be solved more elegantly by using Apollonius def
/r  50 def /R 100 def
/Rr R r add def /Rmr R r sub def
%Circles
 0 0 moveto     r neg 0  r   0 360 arc
 0 0 moveto         R 0  R 180 540 arc
 R 2 mul 0 moveto Rmr 0 Rr   0 360 arc
 stroke
gsave [2] 1 setdash
 R 0 moveto         0 0  R   0 360 arc
stroke
grestore
```

```
%tangent circle
/q Rmr neg Rr div def
/x{d q mul}def
/y{d r mul Rr div
   d R mul Rr div R add mul sqrt 2 mul}def
/f{Rr d sub Rmr x sub dup mul
            y y mul add sqrt sub
}def
/solveit{%invariant: l f > 0 and u f <= 0
  /d .5 l u add mul def
  f 0 gt{/l d def}
        {/u d def}ifelse
  u l sub eps gt{solveit}
                {/d .5 l u add mul def}ifelse
}def
%touching circle
/eps .01 def
/l .5 r mul def /u r def solveit
x d add y moveto x y d 0 360 arc stroke
%inversions
r neg 0 moveto currentpoint x y lineto
R 0 lineto lineto
 r 0 moveto -.5 r mul 0 1.5 r mul 0 360 arc
-2 r mul 2 R mul moveto -2 r mul -2 R mul lineto
   r     2 R mul moveto    r     -2 R mul lineto
-.5 r mul 2 R mul moveto -.5 r mul -2 R mul lineto
 r Rr moveto -.5 r mul Rr .5 Rr mul 0 360 arc
0 0 moveto -.5 r mul Rr lineto
1 setlinewidth .75 setgray stroke
end}def
%%EndProlog
circlecoveredbycirclesschema showpage
%%EOF
```

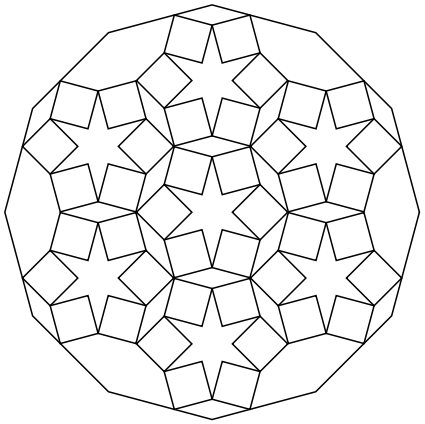
Nontight squares II

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nontight squares II. cgl 1997
%%BoundingBox: -71 -71 71 71
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nontightsquaresIIdict 9 dict def
/nontightsquaresII{nontightsquaresIIdict begin %tilxix
0 setlinejoin 1 setlinecap
/r 50 def
/s r .5858 mul def  /ms s neg def
/sds .707 s mul def /msds sds neg def
/element{gsave r 0 translate
   sds 0 moveto
   0 sds lineto
    0 s  rlineto %square on top
    ms 0 rlineto
    0 ms rlineto
    s 0  rlineto
   msds 0 lineto
   0 msds lineto
   sds 0 lineto fill grestore
}def
/f r sds sub r sds add div def
/gl 1 def
%
3{4{element 90 rotate}repeat
  f f scale /gl f gl mul def
  gl setgray}repeat
end}def
%%EndProlog
nontightsquaresII showpage
%%EOF
```



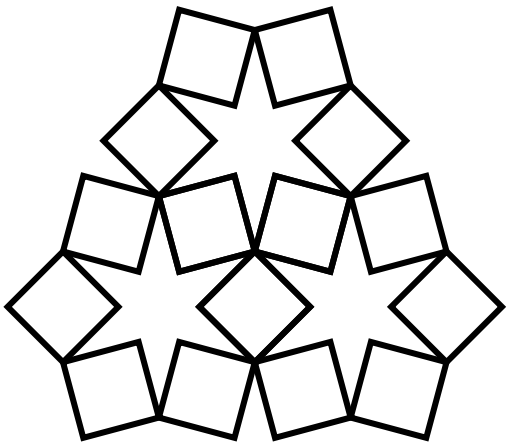
Nontight squares III

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nontightsquares III. cgl 1997
%%BoundingBox: -150 -150 150 150
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nontightsquaresIIIdict 7 dict def
/nontightsquaresIII{nontightsquaresIIIdict begin %tilxxii
0 setlinejoin 1 setlinecap
/r 50 def
/s r 2.8284 75 sin mul div def
/d r 2 mul s .707 mul sub def
/dia{gsave s 0 moveto
   4{0 s lineto 90 rotate}repeat
   closepath stroke grestore
}def
/dodeca{6{gsave
   r s sub 0 translate
   dia
   grestore
   60 rotate}repeat
}def
gsave dodeca grestore
6{gsave
  0 d translate
  dodeca
  grestore
  60 rotate}repeat
gsave
r d add 0 moveto
12{30 rotate
   r d add 0 lineto}repeat
%3 setlinewidth
stroke
grestore
%
0 300 translate
/rstar s s mul r s sub dup mul add sqrt def
3{gsave
  0 rstar translate
  dodeca
  grestore
  120 rotate}repeat
end}def
%%EndProlog
nontightsquaresIII showpage
%%EOF
```



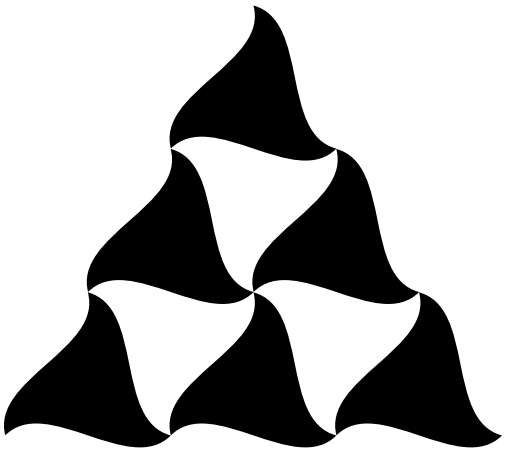
Nontight squares IV

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nontight squares Iv. cgl 1997
%%BoundingBox: -43 -32 43 43
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nontightsquaresivdict 6 dict def
/nontightsquaresiv{nontightsquaresivdict begin %tilxxviii
0 setlinejoin 1 setlinecap
/r 25 def %r dodecahedron
/s r 2.8284 75 sin mul div def
/d r 2 mul s .707 mul sub def
/rotsq{gsave s 0 moveto
   4{0 s lineto 90 rotate}repeat
   closepath stroke grestore
}def
/dodeca{6{gsave r s sub 0 translate
   rotsq grestore 60 rotate}repeat}def
%
/rstar s s mul r s sub dup mul add sqrt def
3{gsave
  0 rstar translate
  dodeca
  grestore
  120 rotate}repeat
end}def
%%EndProlog
nontightsquaresiv showpage
%%EOF
```



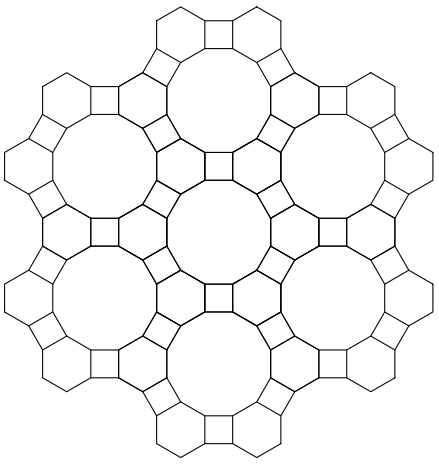
Triangles

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: 6 triangles. cgl 1997
%%BoundingBox: -30 -20 127 117
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/6trianglesdict 10 dict def
/6triangles{6trianglesdict begin %tilxLiii
0 setlinejoin 1 setlinecap
/s 50 def
/hs .5 s mul def  /mhs hs neg def
/hss 1.732 hs mul def
/qs .5 hs mul def /mqs qs neg def
/r 1.15 hs mul def
/hr .5 r mul def /mhr hr neg def
/tri{mhs mhr moveto
     3{mqs mhr qs add
       qs mhr mqs add
       hs mhr curveto
       120 rotate
      }repeat fill
}def
tri
gsave 2{s 0 translate tri}repeat grestore
gsave hs hss translate tri
       s   0 translate tri
grestore
s 2 hss mul translate tri
end}def
%%EndProlog
6triangles showpage
%%EOF
```



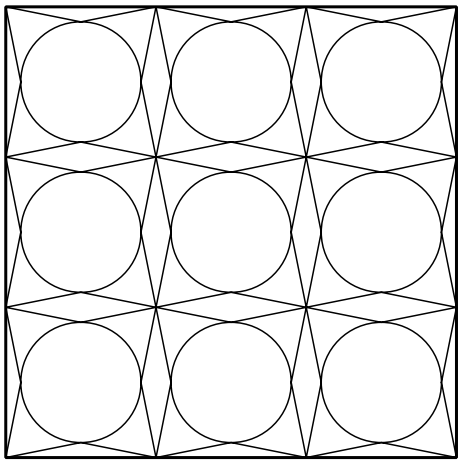
Tile 6-4-12-4

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: tile6-4-12-4. cgl 1997
%%BoundingBox: -201 -211 201 211
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/tile6-4-12-4dict 9 dict def
/tile6-4-12-4{tile6-4-12-4dict begin %tilxxix
/r 50 def /hs r 15 sin mul def  /mhs hs neg def
/hss 1.732 hs mul def
/s 2 hs mul def
/rin  r 15 cos mul def
/shift rin hs add 2 mul def
/hexagon{hss hs moveto
  6{60 rotate hss hs lineto}repeat
stroke}def
/tile{6{gsave
   rin hss add 0 translate
   hexagon
   grestore 30 rotate gsave
   rin 0 translate %‘square’
   0 mhs moveto 0 s rlineto
   s mhs moveto 0 s rlineto stroke
   grestore
   30 rotate}repeat
}def
tile
gsave 0 shift translate tile grestore
gsave 0 shift neg translate tile grestore
gsave .866 shift mul .5 shift mul translate tile grestore
gsave -.866 shift mul .5 shift mul translate tile grestore
gsave .866 shift mul -.5 shift mul translate tile grestore
gsave -.866 shift mul -.5 shift mul translate tile grestore
end}def
%%EndProlog
tile6-4-12-4 showpage
%%EOF
```



9 circles

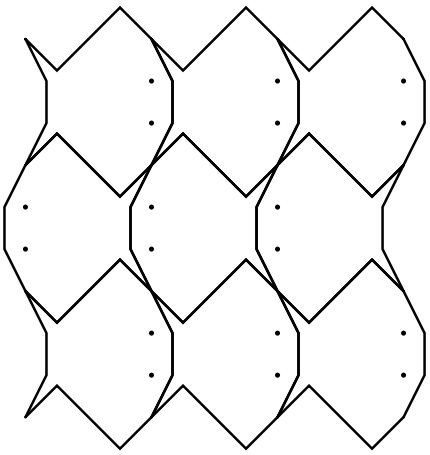
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: 9 circles. cgl 1997
%%BoundingBox: -51 -51 251 251
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/9circlesdict 6 dict def
/9circles{9circlesdict begin %tilxxiv
/s 50 def      /-s s neg def/2s 2 s mul def /5s 5 s mul def
/r .8 s mul def/-r r neg def/2r 2 r mul def
gsave
3{%rows
  gsave
  3{%horizontal tiles
    4{-s -s moveto -r 0 lineto -s s lineto 90 rotate}repeat
    r 0 moveto 0 0 r 0 360 arc
    2s 0 translate
    }repeat
  stroke
  grestore
  0 2s translate
}repeat
grestore
%frame
2 setlinewidth
-s -s moveto 5s -s lineto 5s 5s lineto -s 5s lineto closepath
stroke
end}def
%%EndProlog
9circles showpage
%%EOF
```



Escher fishes

By a little deformation of a 3x3 tile nice artistic results have been obtained by Escher.

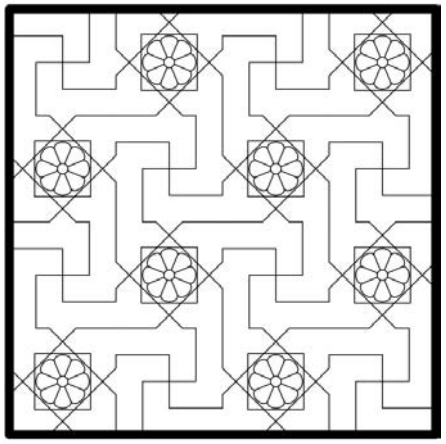
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escherfishes, cgl 1997
%%BoundingBox: -85 -90 85 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/escherfishesdict 12 dict def
/escherfishes{escherfishesdict begin %tilLxxii
0 setlinejoin 1 setlinecap
/a 50 def          /ma a neg def
/ha .5  a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



Escher Alhambra tile

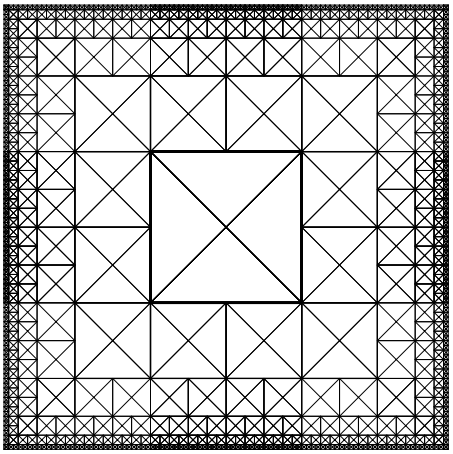
My old Metafont example has been included because it is a nice tile, and it has been processed by Troy Henderson's remote MP Previewer.

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

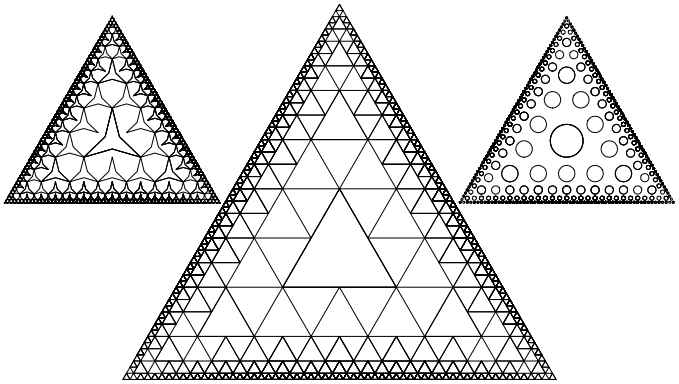


Limit square

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Limit square. cgl 1997
%%BoundingBox: -96 -96 96 96
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/limitsquaredict 17 dict def
/limitsquare{limitsquaredict begin %tilxLiv
0 setlinejoin 1 setlinecap
/gs 64 def /mgs gs neg def
/sq{%lw s x y   on stack
  /y exch def /lx exch def
  /hs exch 2 div def /mhs hs neg def
  /llw exch def
  gsave lx y translate llw setlinewidth
     element
  grestore
  .5 llw mul hs lx 1.5 hs mul add y .5 hs mul add
  .5 llw mul hs lx 1.5 hs mul add y .5 hs mul sub
  hs 1 gt {sq sq}
          {8{pop}repeat} ifelse
}def
/element{hs mhs moveto hs hs lineto
    mhs hs lineto mhs mhs lineto
    closepath stroke
    hs mhs moveto mhs hs lineto
    mhs mhs moveto hs hs lineto
    .25 llw mul setlinewidth stroke
}def
%
/pattern{4{/lw 1 def
  /s gs def /ms s neg def /x 0 def
  gsave
  6{gsave
    2{lw s x x sq
      -1 1 scale 90 rotate
     }repeat
    grestore
    /s .5 s mul def /lw .5 lw mul def
    /x x 1.5 s mul add def
   }repeat
  grestore -90 rotate
 }repeat
}def
%
pattern
end}def
%%EndProlog
limitsquare showpage
%%EOF
```



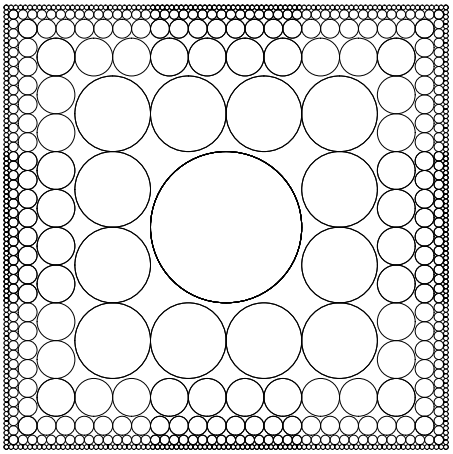
Triangular limit grids



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Triangular limit grids, cgl 1997
%%BoundingBox: -190 -71 190 142
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run % contains growthspiral
/triagularlimgridsdict 10 dict def
/triagularlimgrids{triagularlimgridsdict begin %tilLix
0 setlinejoin 1 setlinecap
/gs 64 def /gr gs 1.732 div def
/tri{%r s x y   on stack
  /y exch def /x exch def /s exch def /r exch def
  gsave x y translate
       /lw .5 lw mul def lw setlinewidth
       element
  .5 r mul .5 s mul     s neg  r neg
  .5 r mul .5 s mul -.5 s mul  r neg
  .5 r mul .5 s mul   0        r neg
  .5 r mul .5 s mul  .5 s mul  r neg
  .5 r mul .5 s mul     s      r neg
  lw .04 gt {tri tri tri tri tri}
          {20{pop}repeat} ifelse
  /lw 2 lw mul def
  grestore
}def
/element{gsave -30 rotate r 0 moveto
     3{120 rotate r 0 lineto}repeat stroke
     grestore
}def
%
gsave
3{/lw 1 def gr gs 0 0 tri 120 rotate}repeat
grestore

gsave
/element{gsave 30 rotate .5 r mul 0 moveto
     0 0 .5 r mul 0 360 arc stroke
     grestore
}def
128 64 translate
/gs 32 def /gr gs 1.732 div def
3{/lw 1 def
  gr gs 0 0 tri 120 rotate}repeat
grestore

/element{gsave 0 r moveto
      3{60 rotate 0 .25 r mul lineto
        60 rotate 0  r     lineto}repeat stroke
     grestore
}def
-128 64 translate
/gs 32 def /gr gs 1.732 div def
3{/lw 1 def
  gr gs 0 0 tri 120 rotate}repeat
end}def
%%EndProlog
triagularlimgrids showpage
%%EOF
%%EOF
```

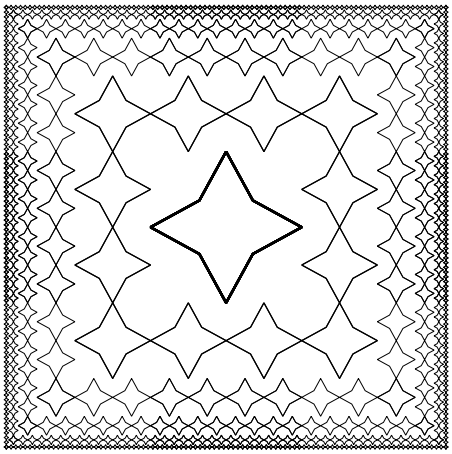
Limit square with circles

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: limitsquareswithcircles. cgl 1997
%%BoundingBox: -96 -96 96 96
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/limitsquareswithcirclesdict 19 dict def
/limitsquareswithcircles{limitsquareswithcirclesdict begin %tilxLviii
0 setlinejoin 1 setlinecap
/gs 64 def /mgs gs neg def
/sq{%lw s x y   on stack
  /y exch def /lx exch def
  /hs exch 2 div def /mhs hs neg def
  /llw exch def
  gsave lx y translate llw setlinewidth
    element
    .25 llw mul setlinewidth stroke
  grestore
  .5 llw mul hs lx 1.5 hs mul add y .5 hs mul add
  .5 llw mul hs lx 1.5 hs mul add y .5 hs mul sub
  hs 1 gt {sq sq}
          {8{pop}repeat} ifelse
}def
/element{hs 0 moveto  0 0 hs 0 360 arc}def
%
/pattern{
4{/lw 1 def /s gs def /ms s neg def /x 0 def
  gsave
  6{gsave
    2{lw s x x sq
      -1 1 scale 90 rotate
     }repeat
    grestore
    /s .5 s mul def /lw .5 lw mul def
    /x x 1.5 s mul add def
   }repeat
  grestore -90 rotate
 }repeat
}def
%
/element{hs 0 moveto  0 0 hs 0 360 arc}def
gsave pattern grestore
%
/element{hs mhs moveto hs hs lineto
    mhs hs lineto mhs mhs lineto
    closepath stroke
    hs mhs moveto mhs hs lineto
    mhs mhs moveto hs hs lineto
}def
end}def
%%EndProlog
limitsquareswithcircles showpage
%%EOF
```



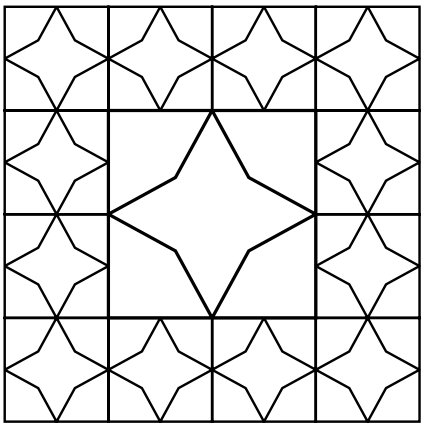
Limit square with stars

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Limit square with stars. cgl 1997
%%BoundingBox: -96 -96 96 96
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/squarewithstarsdict 19 dict def
/squarewithstars{squarewithstarsdict begin %tilLii
0 setlinejoin 1 setlinecap
/gs 64 def /mgs gs neg def
/sq{%lw s x y   on stack
  /y exch def /lx exch def
  /hs exch 2 div def /mhs hs neg def
  /llw exch def
  gsave lx y translate llw setlinewidth
    element
    .25 llw mul setlinewidth stroke
  grestore
  .5 llw mul hs lx 1.5 hs mul add y .5 hs mul add
  .5 llw mul hs lx 1.5 hs mul add y .5 hs mul sub
  hs 1 gt {sq sq}
          {8{pop}repeat} ifelse
}def
/element{hs 0 moveto  0 0 hs 0 360 arc}def
%
/pattern{
4{/lw 1 def /s gs def /ms s neg def /x 0 def
  gsave
  6{gsave
    2{lw s x x sq
      -1 1 scale 90 rotate
     }repeat
    grestore
    /s .5 s mul def /lw .5 lw mul def
    /x x 1.5 s mul add def
   }repeat
  grestore -90 rotate
 }repeat
}def
%
/element{hs 0 moveto
         4{45 rotate .5 hs mul 0 lineto
           45 rotate    hs     0 lineto}repeat
         closepath stroke
}def
gsave pattern grestore
end}def
%%EndProlog
squarewithstars showpage
%%EOF
```



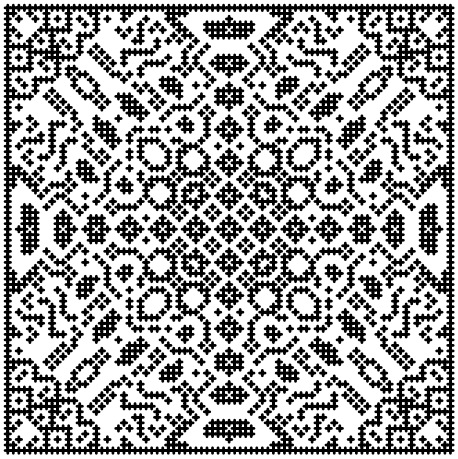
Ikon

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Ikon. cgl 1997
%%BoundingBox: -70 -70 70 70
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/ikondict 12 dict def
/ikon{ikondict begin%tilLi
0 setlinejoin 1 setlinecap
/lw 1 def
/s 64 def /ms s neg def
/hs .5 s mul def /mhs hs neg def
/element{%hs on stack
    /hsloc exch def /mhsloc hsloc neg def
    hsloc mhsloc moveto hsloc hsloc lineto
    mhsloc hsloc lineto mhsloc mhsloc lineto
    closepath stroke
    hsloc 0 moveto
    4{45 rotate .5 hsloc mul 0 lineto
      45 rotate    hsloc     0 lineto}repeat
    closepath stroke
}def
/indices[-1.5 -.5 .5 1.5]def
%
hs element .75 setlinewidth
indices{/r exch def
indices{/c exch def
  c r mul abs .5 gt
     {gsave r hs mul c hs mul translate
            .5 hs mul element
      grestore}if
}forall}forall
end}def
%%EndProlog
ikon showpage
%%EOF
```



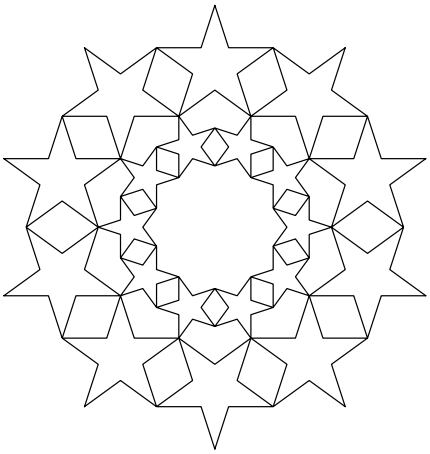
Broidery

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Broidery, cgl 1997
%%BoundingBox: -82 -82 82 82
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/broiderydict 9 dict def
/broidery{broiderydict begin %tilLxxiv
0 setlinejoin 1 setlinecap
/c 400 def /n 40 def /mn n neg def
/s 2 def %scaling of picture
/cross{gsave x n mul s mul y n mul s mul translate
   -.5 s mul 0 moveto .5 s mul 0 lineto
   0 -.5 s mul moveto 0 .5 s mul lineto stroke
   grestore
}def
/f{1 x x mul sub 1 y y mul sub mul}def
mn 1 n{/x exch n div def
mn 1 n{/y exch n div def
  /v f c mul truncate def
  v cvi 3 mod 0 eq{cross}if
}for}for
end}def
%%EndProlog
broidery showpage
%%EOF
```



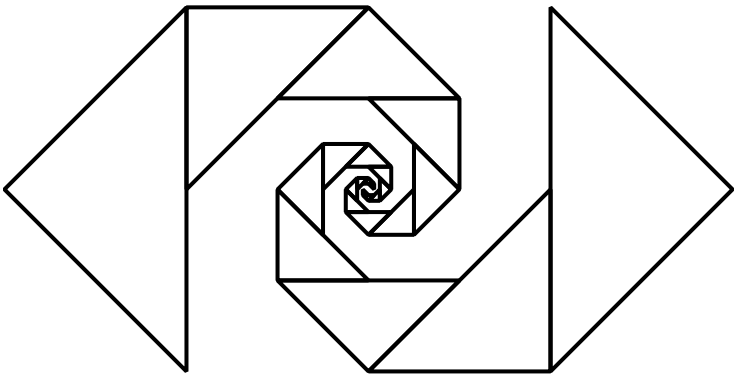
Nontight stars

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Nontight stars. cgl 1997
%%BoundingBox: -175 -185 175 185
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nontightstarsdict 11 dict def
/nontightstars{nontightstarsdict begin %tilxxxvii
0 setlinejoin 1 setlinecap
/r 50 def 2 setlinejoin
/star{rp 0 moveto
  5{rp 72 cos mul
    rp -72 sin div 72 cos dup 1 sub mul mul lineto
    72 rotate rp 0 lineto}repeat
stroke}def
/tile{10{gsave xp 0 translate
         star
   grestore
   36 rotate}repeat
}def
%
2{/hs r 18 sin mul def  /mhs hs neg def
/hss 1.732 hs mul def
/s 2 hs mul def
/rp hs 36 sin div def
/rip rp 36 cos mul def
/xp r 18 cos mul rip add def
tile
18 rotate
/r xp rp add def
}repeat
end}def
%%EndProlog
nontightstars showpage
%%EOF
```



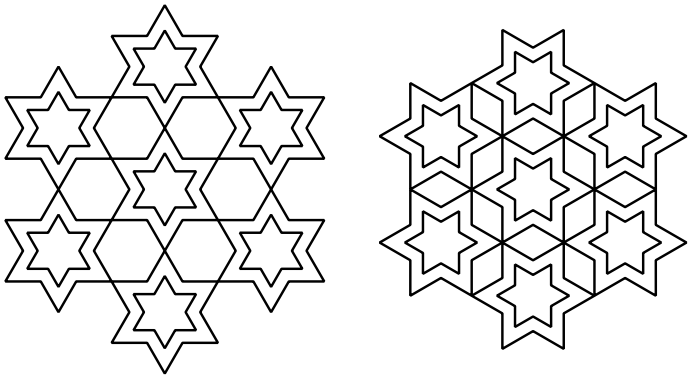
EW rencontre

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: EWrencontre. cgl 1997
%%BoundingBox: -2 -47 182 47
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/ewrencondict 5 dict def
/ewrencontre{ewrencondict begin %tilxLv
0 setlinejoin 1 setlinecap
/s 64 def /ts 2 s mul def
/tri{/locs exch def
     0 locs moveto locs 0 lineto 0 0 lineto
     closepath currentpoint stroke
     translate -45 rotate
     /locs .707 locs mul def
     locs 1 ge {locs tri} if
}def
2 setlinejoin -45 rotate
gsave s tri grestore
ts ts translate 180 rotate
s tri
end}def
%%EndProlog
ewrencontre showpage
%%EOF
```



Tiling stars

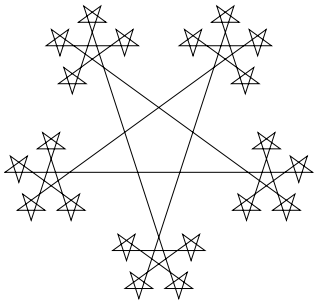
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Tiling stars. cgl 1997
%%BoundingBox: -72 -77 217 77
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/tilingstarsdict 4 dict def
/tilingstars{tilingstarsdict begin %tiliL
0 setlinejoin 1 setlinecap
/r 25 def 2 setlinejoin
/six{%r on stack
  /rloc exch def
  rloc 0 moveto
  6{30 rotate .577 rloc mul 0 lineto
    30 rotate  rloc     0 lineto}repeat
    closepath stroke
}def
gsave
30 rotate
r six .588 r mul six
6{gsave 2 r mul 0 translate
  r six .588 r mul six
  grestore
  60 rotate}repeat
grestore
%
6 r mul 0 translate
r six .588 r mul six
30 rotate
6{gsave 1.732 r mul 0 translate 30 rotate
     r six  .588 r mul six
  grestore 60 rotate}repeat
end}def
%%EndProlog
tilingstars showpage
%%EOF
```



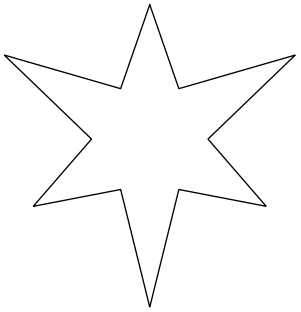
Lauwerier's star fractal

An interesting fractal start picture has been created by Lauwerier in BASIC, which I have translated into PostScript. See *Classical Math fractals in PostScript.*

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lauwerier Star fractal
%%BoundingBox: -5 -130 305 165
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/starfractaldict 8 dict def
/starfractal%stack: reduction angle p (=depth) v(ertices)
{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
%%EndProlog
/s {300 mul}def
.3 144 3 4 starfractal stroke showpage
%%EOF
```



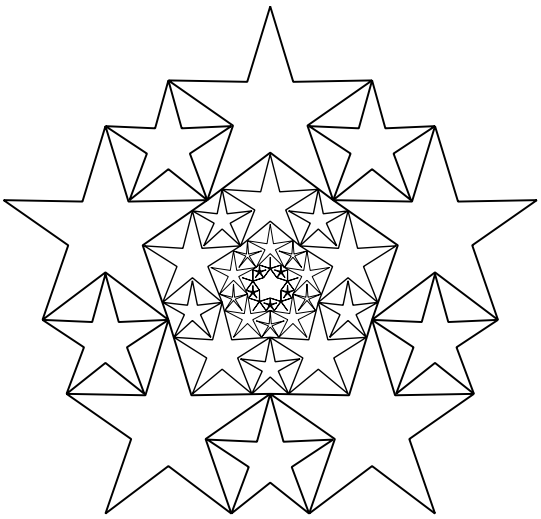
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Staroutline. cgl 1997
%%BoundingBox: -110 -125 110 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/staroutlinedict 4 dict def
/staroutline{staroutlinedict begin %jalxxv
/n 6 def /r 100 def%star data
%auxiliaries
/alfah 180 n div def
/rin r 2 div alfah cos mul def
0 r moveto
1 1 n{alfah rotate
  0 rin lineto
  alfah rotate
  2 mod 0 eq {0 r lineto}
             {0 r 1.25 mul lineto}ifelse
  }for
stroke
end}def
%%EndProlog
staroutline showpage
%%EOF
```



Tiling of stars

In 1996 I tiled this picture in PostScript. See *Tiling in PostScript and Metafont — Escher's wink.*

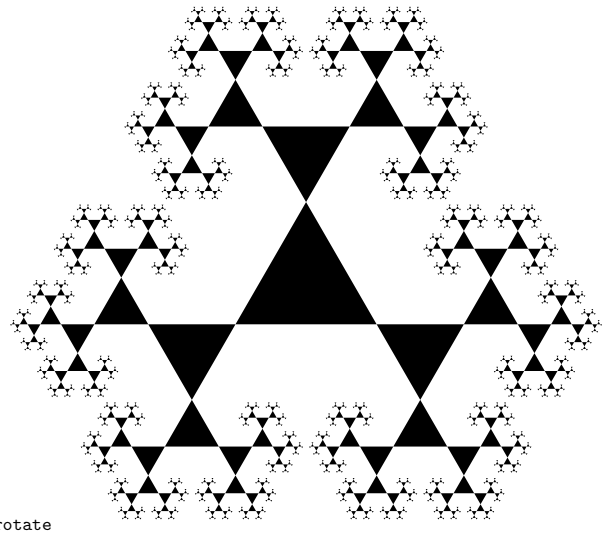
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lauwerier Star fractal
%%BoundingBox: -5 -130 305 165
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%Title: Tiling of pentagrams
%%BoundingBox: -100 -85 100 105
%%Author: Kees van der Laan, kisa1@xs4all.nl, Nov 96
%%BeginSetup
%%EndSetup
%%BeginProlog
1.4 setmiterlimit
/star{%n r on stack, n>=5
/rstar exch def /nstar exch def
/alfahstar 180 nstar div def
/rinstar rstar 2 div alfahstar cos
mul 1.5 sub def
0 rstar moveto
nstar{alfahstar rotate 0 rinstar lineto
alfahstar rotate 0 rstar lineto
}repeat closepath stroke
}def
%%EndProlog
/n 5 def /r 5 def %(r,n)-gon
/alfa 360 n div def
/lw .1 def 2 setlinejoin
%loop over the 'rings'
4{lw setlinewidth
/rh r 2 div def %r half
/rin r 36 cos mul def %r inside
%big stars
/c rin 36 cos mul 2 mul def
n{gsave 0 c translate n rin star
grestore alfa rotate
}repeat
%'half-sized' stars
gsave 36 rotate
/c rin rh add def
n{gsave 0 c translate 36 rotate
n rh star
grestore alfa rotate
}repeat
grestore
/r 36 cos dup 2 mul 1 add mul r mul def
/lw lw .2 add def
}repeat
showpage
%%EOF
```



Variant post-processed by Photoshop.

Triangle fractal

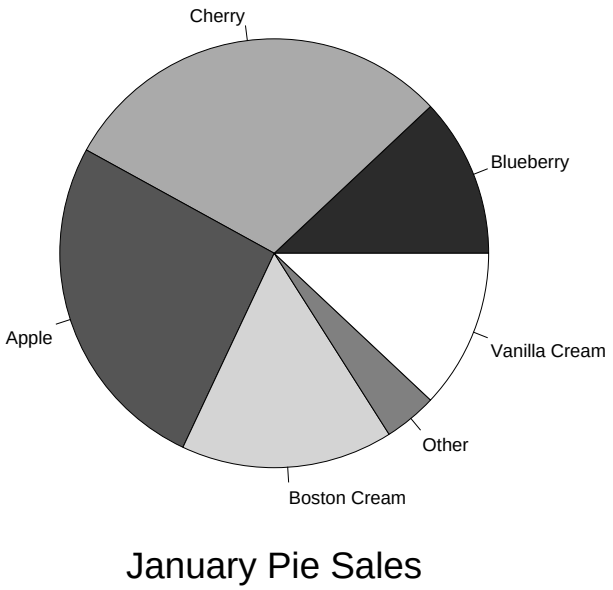
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Phi triangle fractal. CGL febr 2010.
%%BoundingBox: -210 -175  210 200  %size 425X367
%%BeginSetup
%%EndSetup
(c:\\PSlib\\PSlib.eps) run

/trianglefractaldict 7 dict def
trianglefractaldict begin
/s 100 def /sqrt3 3 sqrt def /r s sqrt3 div def
/phi 1.61803 def /phiinv .61803 def
/triangle{% r s global
   gsave s .5 mul r -.5 mul moveto
   3{120 rotate
     s .5 mul r -.5 mul lineto
    }repeat closepath fill
   grestore
}def

/rectriangle{triangle /level level 1 sub def
/r r phi div def /s s phi div def
level 1 gt %two children
 {gsave r phi mul  r add .5 mul dup sqrt3 mul exch neg translate 60 rotate
   rectriangle%recursion
  grestore
   gsave
    r phi mul r add -.5 mul  dup sqrt3 mul exch translate -60 rotate
   rectriangle%recursion
  grestore
  } if
  /level level 1 add def
/r r phi mul def  /s s phi mul def
}def
end

/trianglefractal{trianglefractaldict begin
triangle
 /r r phi div def  /s s phi div def
3{gsave r phi mul  r add .5 mul  dup sqrt3 mul exch neg translate
   60 rotate rectriangle
  grestore
120 rotate}repeat
end
}def

/level 10 def %first triangle has 3 childeren, other levels only 2
trianglefractal showpage
%%EOF
```



Drawing a pie chart

Adobe's Bluebook P14, p183. Not so relevant in view of special chart programs.

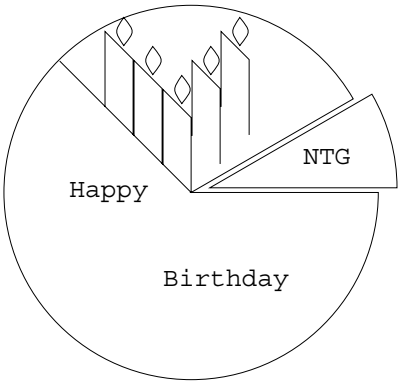
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Adobe's Pie chart
%%BoundingBox: 125 170 530 565
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%%EndProlog
%Actual arguments on the stack
(January Pie Sales)    %Title string
  24 12 %Font size: title, labels
  [ [(Blueberry) .12 ] %Array of data arrays
    [(Cherry) .30 ]
    [(Apple) .26 ]
    [(Boston Cream) .16 ]
    [(Other) .04 ]
    [(Vanilla Cream) .12 ]
  ] 306 396            %translate center to
    140                %size
DrawPieChart showpage %Invoke
%%EOF
```



Voss shows in his *PSTricks—Graphics and PostScript for TEX and LaTEX, p587–590* how to draw pie chart with PSTricks in LaTEX.

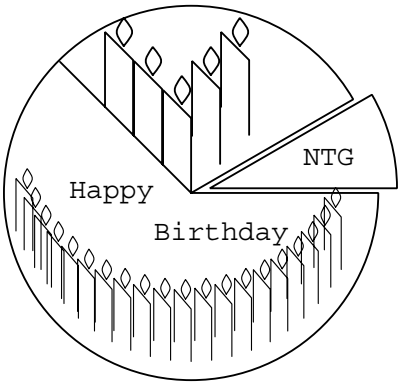
NTGLUSTRUM

Created as PostScript def on the occasion of EuroTEX2012.




```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: NTG lustrum
%%BoundingBox: -101 -101 111 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/ntglustrum{/Courier 15 selectfont
/candle{0 0 moveto 0 8 s lineto -3 s 11 s lineto
        -3 s 3 s lineto stroke
        -1 s 9.5 s moveto
           -2 s 11 s -2 s 11 s -1 s 12.5 s curveto
            1    11 s  1  11 s -1 s 9.5 s curveto stroke}def
/s{ 5 mul} def
0 0 moveto 0   0   20 s 30 360 arc closepath stroke
2 s .5 s moveto 2 s .5 s 20 s 0 30 arc closepath stroke
0 0 moveto 10 s -1.4 mul  dup neg lineto stroke candle
gsave 2{-3.1 s 3.1 s translate candle}repeat grestore
gsave 2{ 3.1 s 3.1 s translate candle}repeat grestore
-13 s -.5 s moveto (Happy) show
-3  s -10 s moveto (Birthday) show
12 s 3 s moveto (NTG) show
}def
%%EndProlog
ntglustrum showpage
%%EOF
```

For the occasion of NTG 25th lustrum

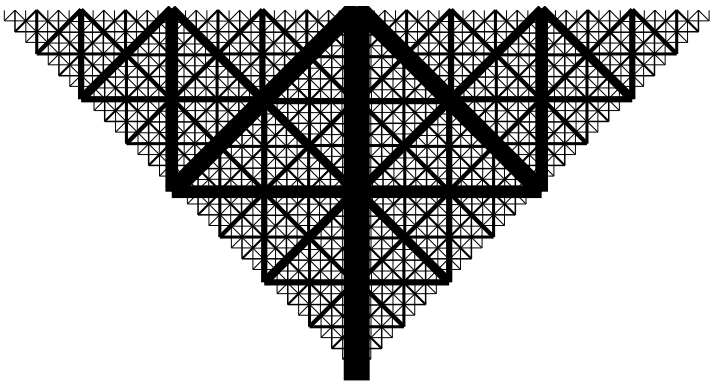



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: NTG25th lustrum. cgl 2014
%%BoundingBox: -101 -101 111 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/ntglustrum{/Courier 15 selectfont
/candle{0 0 moveto 0 8 s lineto -3 s 11 s lineto
        -3 s 3 s lineto stroke
        -1 s 9.5 s moveto
           -2 s 11 s -2 s 11 s -1 s 12.5 s curveto
            1    11 s  1  11 s -1 s 9.5 s curveto
        stroke}def
/s{ 5 mul} def
0 0 moveto 0   0   20 s 30 360 arc closepath stroke
2 s .5 s moveto 2 s .5 s 20 s 0 30 arc closepath stroke
0 0 moveto 10 s -1.4 mul  dup neg lineto stroke
candle
gsave 2{-3.1 s 3.1 s translate candle}repeat grestore
gsave 2{ 3.1 s 3.1 s translate candle}repeat grestore
-13 s -.5 s moveto (Happy) show
-4  s -5 s moveto (Birthday) show
12 s 3 s moveto (NTG) show
/r 18 s def
200 7 339{/angle exch def
 gsave r angle cos mul r angle sin mul translate .6 .7 scale candle grestore}for
}def
%%EndProlog
ntglustrum showpage
%%EOF
```

Fractal arrow

Adobe's Blue book p74 explains recursive graphics in PostScript by means of a fractal arrow. Note the automatic decrease of the line width. Variation of the angle gives interesting results, such as the H-fractal.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Fractal Arrow, Blue book p74.
%%BoundingBox: 200 399 400 501
%%EndSetup
%%BeginProlog
/FracArrowdict 5 dict def
FracArrowdict begin
/depth 0 def /maxdepth 10 def
/down{/depth depth 1 add def}def /up{/depth depth 1 sub def}def
/DoLine%print a vert. line
   {0 144 rlineto currentpoint stroke translate 0 0 moveto}def
end
/FracArrow{FracArrowdict begin
  gsave .7 dup scale 10 setlinewidth
  down DoLine
  depth maxdepth le {135 rotate FracArrow
                     -270 rotate FracArrow}if
  up grestore
end}def
%%EndProlog
300 400 moveto FracArrow stroke
 showpage
%%EOF
```



Adobe explains recursion in stack-oriented PostScript by the program `factorial`, Blue book p71. Interesting is also how to print numbers in PostScript and go to a newline. Rudimentary text processing. Useful when you do calculations in PostScript, and want visualize the results.

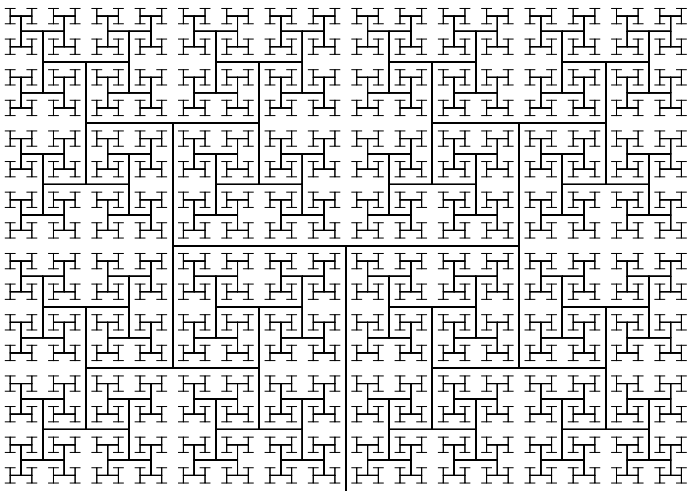
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: n-Factorial. Blue book p72
%%BoundingBox: 71 -150 155 15
%%BeginSetup
%%EndSetup
%%BeginProlog
/LM 72 def /Times-Roman 15 selectfont /nstr 7 string def
/newline{currentpoint 16 sub exch pop LM exch moveto} def
/prt-n %stack: n
   {nstr cvs show}def
/prtFactorial %stack: n
   {dup prt-n (!=) show
    factorial prt-n newline}def
/factorial %stack: n --> n faculty
{dup 1 gt{dup 1 sub factorial mul}if}def
%%EndProlog
LM 0 moveto 1 1 10{prtFactorial}for
showpage
%%EOF
```

1!=1
2!=2
3!=6
4!=24
5!=120
6!=720
7!=5040
8!=40320
9!=362880
10!=3628800

H-fractal

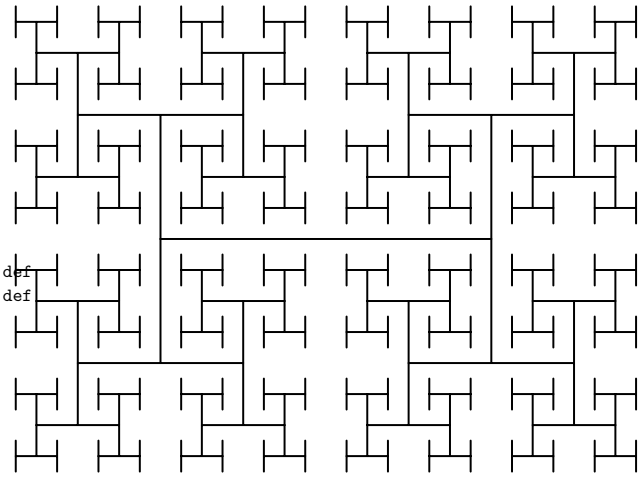
Much similar to Adobe's fractal arrow. It is also shown how to measure performance time.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Hfractal. CGL
%%BoundingBox: -146 -15 146 201
%%BeginSetup
%%EndSetup
%%BeginProlog
/Hf%stack: order
{ 0 100 rlineto
 1 sub dup  0 gt{
       gsave 90 rotate .707 dup scale  Hf
       grestore
            -90 rotate .707 dup scale Hf
      }if 1 add stroke} def
%%EndProlog
0 0 moveto
/time usertime def gsave 12 Hf grestore
/Courier 12 selectfont
0 -12 rmoveto
(Time needed: ) show
usertime time sub (    ) cvs show (ms) show  showpage
%%EOF
```



Lauwerier programmed in BASIC, though I much prefer the recursive version for its conciseness and cla

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Boomh2. cgl 2014
%%BoundingBox: -160 -120 160 120
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/boomh2dict 22 dict def
/boomh2%p (order)==> Binary Tree H-fractal, backtrack method
{boomh2dict begin 150 150 scale .006 setlinewidth
/p exch def /p1 p 1 add def
 /x1 p1 array def /x2 p1 array def /x3 p1 array def /x4 p1 array def
 /y1 p1 array def /y2 p1 array def /y3 p1 array def /y4 p1 array def
 /a .5 def
 x1 0 0 put y1 0 0 put /d 1 def
 draw
 1 1 4 p 1 sub exp 1 sub
  {/m exch def /d p def
   {m 4 mod 0 eq{/m m 4 idiv def /d d 1 sub def}
                {exit}ifelse
   }loop
   /dm1 d 1 sub def
   x1 dm1 x2 dm1 get put  x2 dm1 x3 dm1 get put  x3 dm1 x4 dm1 get put
   y1 dm1 y2 dm1 get put y2 dm1 y3 dm1 get put  y3 dm1 y4 dm1 get put   draw
  }for%m
stroke end}bind def
/draw{d 1 p{/j exch def
   /x x1 j 1 sub get def /y y1 j 1 sub get def
   /b a j exp def /c 1.5 a mul b mul def
   x1 j x b add put y1 j y c add put  x2 j x b add put y2 j y c sub put  x3 j x b sub put y3 j y c add put
   x4 j x b sub put y4 j y c sub put
   x b sub  s y        s moveto x b add  s y        s lineto
   x1 j get s y1 j get s moveto x2 j get s y2 j get s lineto x3 j get s y3 j get s moveto x4 j get s y4 j get s lineto
  }for%j
}bind def
4 boomh2 showpage
%%EOF
```



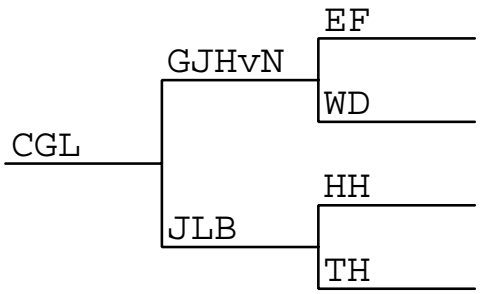
Binary tree

In the TEXbook in the tables chapter, exc22.14, a binary tree has been used, implicitly. The production rule à la Lindenmayer for the balanced tree reads

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

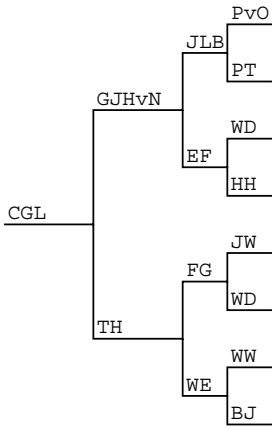
Bt_n the Binary tree of order n,
E_n, N_n, S_n means draw East, North, South with step-size n
$\oplus$ means splice operator, i.e. concatenate properly,
[means store graphics state on the GS-stack and open a new one,
] means remove current graphics state off the Graphics State stack and recall previous.[18]

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Binary tree and names added
%%BoundingBox: -1 -50 185 66
%%BeginSetup
%%EndSetup
%%BeginProlog
/Bintree{%stack: n
E  /kk exch 2 div def
 kk 8 gt {currentpoint N kk Bintree
          moveto S kk Bintree  }if
   /kk kk 2 mul def
 }def %end Bintree
/N{0 kk rlineto }def
/E{gsave  ntg k get 2 3 rmoveto show grestore
    /k k 1 add def
     60 0 rlineto }def
/W{kk neg 0 rlineto}def
/S{0 kk neg rlineto}def
%%EndProlog
/ntg{[(CGL) (GJHvN) (EF) (WD) (JLB) (HH) (TH)]}def
/k 0 def  /Courier 15 selectfont
0 0 moveto 64 Bintree stroke showpage
%%EOF
```



My variant in TEX, where I used Papert's Turtle Graphics.

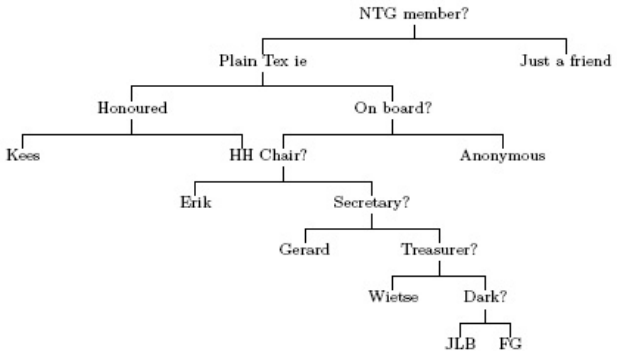
```
\def\bintree{\E{\the\n}%
   \ifnum\n=2 \eertnib\fi
   \divide\n2 {\N{\the\n}\bintree}%
              \S{\the\n}\bintree}%
\def\eertnib#1\bintree{\fi}%terminator
\let\Eold\E                %Text at leaves
\def\E#1{\global\advance\k1
  \xytxt{ \csname\the\k\endcsname}%leaf
  \Eold8}}
\def\1{CGL}\def\2{GJHvN}\def\3{JLB}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



[18] The addition of the graphics state concept to the Lindenmayer production rules is an enrichment.

An unbalanced tree in TEX

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



Below from `bttool` which comes with `BLUe.tex`, the backtrack macro has been copied. The nodes are binary numbers, and when the `node<binary number>` is undefined the end of the branch has been reached, and backtracking is performed. `\whiteS` draws South and the leaves. Pretty advanced macro writing. BLUe in full glory.

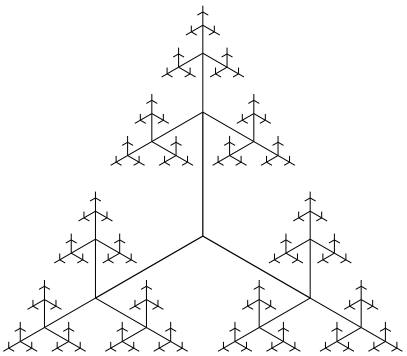
```
\beginbt \def\drawbt{\whiteS{120}%
   \ea\ifx\csname\node0\endcsname\relax
          \tbward\fi%Backtrack
   \S{80}\advance\k-125
   {\W{\the\k}\S{80}\edef\node{\node0}%
    \drawbt}%
    \E{\the\k}\S{80}\edef\node{\node1}%
    \drawbt\relax}
\def\tbward#1\relax{\fi}
```

Note the minimal, necessary data specifications: just the binary 'addresses' of the nodes next to their contents. TEX will handle all that is needed.

Trinary Tree

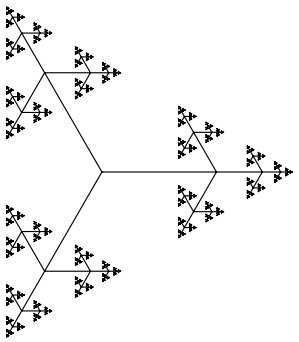
One might add another branch to a binary tree and create trinary trees.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Trinary Tree. cgl, kisa1@xs4all.nl, 2011
%%BoundingBox: -175 -105 175 200
%%BeginSetup
%%EndSetup
%%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
/s 100 def 6 trinarytree
showpage
```



Lauwerier programmed

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Lauwerier Boom3. cgl 2014
%%BoundingBox: -100 -155 180 1565
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/boom3dict 12 dict def
/boom3%p (order)==> Trinary Tree fractal
{boom3dict begin gsave 100 100 scale .01 setlinewidth
/p exch def /t p 1 add array def /a .4 def
 0 1 p{/m exch def
  0 1 3 m exp 1 sub{/n exch def
    /n1 n def
    1 1 m {/l exch def
      t l n1 3 mod put /n1 n1 3 idiv def
      }for%l
    /x 0 def /y 0 def
    1 1 m {/k exch cvi def
      /f 120 t k get mul def
      /x x f cos a k 1 sub exp mul add def
      /y y f sin a k 1 sub exp mul add def
      }for%k
      x s y s moveto x    a m exp add     s  y                          s lineto
      x s y s moveto x .5 a m exp mul sub s  y .8660254 a m exp mul add s lineto
      x s y s moveto x .5 a m exp mul sub s  y .8660254 a m exp mul sub s lineto
    }for%n
}for%m
stroke grestore end}bind def
6 boom3 showpage
%%EOF
```

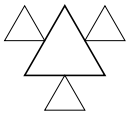


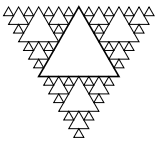
Sierpiński's sieve

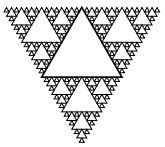
is related to a trinary tree, and constructed by deleting the inner 'half-triangles', which are obtained by connecting the midpoints of the sides, recursively ad infinitum.

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. cgl 2012, kisa1@xs4all.nl
%%BoundingBox:-26 -425 76 46
%%BeginSetup
%%EndSetup
%%BeginProlog
/SierTridict 1 dict def
SierTridict begin /y{3 sqrt 4 div s mul}def end
/SierTri{%stack: order >=1 ==> Sierpinski triangle fractal ( s global)
SierTridict begin
1 sub  dup 0 ge
{gsave 0 0 moveto s 0 lineto s 2 div .869 s mul lineto closepath stroke
grestore%triangle order 1
 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
end} def
%%EndProlog
/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
```



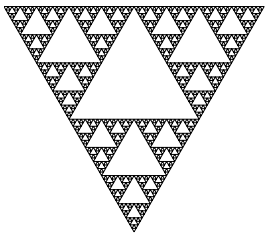




Lauwerier's approach

The idea behind the code is to associate each triangle with a trinary number.

```
%%Title: Sierpinski Sieve
%%Author: H.A. Lauwerier (in BASIC)
%%Transcriptor: cgl, kisa1@xs4all.nl, 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
/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
```

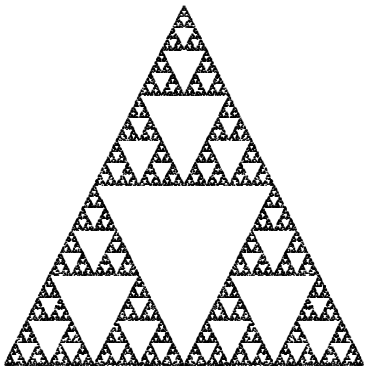


Sierpiński triangle by Iterated Function System

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).

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

```
%!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 $(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k, n \geq 0$. In PWT(1995) for low n Pascal's triangle was obtained by TEX alone simply by.

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

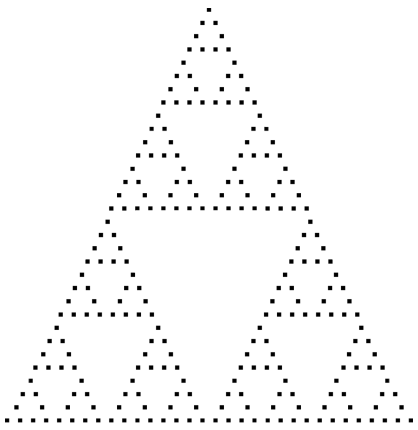
$$\begin{array}{c} 1 \\ 1 \quad 1 \\ 1 \quad 2 \quad 1 \\ 1 \quad 3 \quad 3 \quad 1 \end{array}$$

For general order n the macro for the Pascal triangle were the entries are calculated by the recursion $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}, n \geq k \geq 1$, 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
\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}
$$\pascal{32}$$
```

Pythagoras Tree

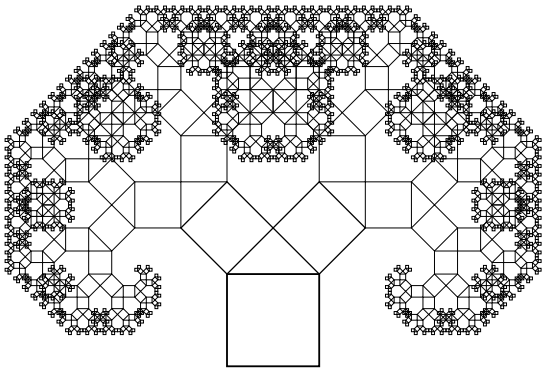
Much similar to Adobe's fractal arrow and my H-fractal. It is all about printing squares at the appropriate place and appropriately scaled.

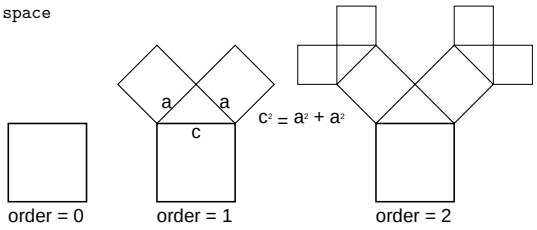
The production rule for a Pythagoras Tree of order n reads

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

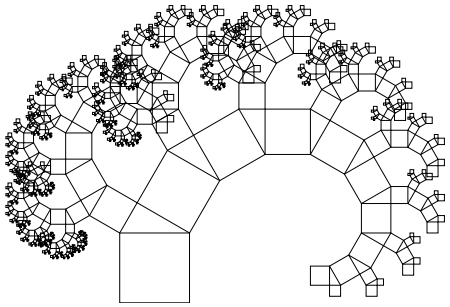
with P_n the Pythagoras Tree of order n, $\oplus$ denotes splice operator, means add properly, [means store graphics state and open a new one,] means close current graphics state and recall previous, R^{45} means rotate US 45° in the PS sense, $S_{a,b}$ means scale US by a and b, $T_{a,b}$ means translate US by a and b, $\square_s$ 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.

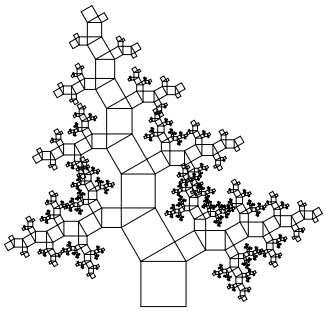
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pythagorean Tree of squares. cgl 2011
%%BoundingBox: -125 -5 175 200
%%BeginSetup
%%EndSetup
%%BeginProlog
%Ultimate example of translating and rotating user space
%The square is the basic element; the triangles are spurious.
%An asymmetrical variant can be obtained after slight modification.
%The asymmetric scaling might introduce different linewidths,
%which can be overcome in 2 ways:
% 1 save and reuse the transformation matrix instead
%   of the gsave grestore pair
% 2 scale the linewidths back to the same thickness
/square{0 0 moveto 0 0 s s rectstroke}def
/pythagoreantree{square                                 %draw the square
1 sub  dup 0 gt
 {gsave 0 s translate 45 rotate .7071 dup scale %transformed user space
  pythagoreantree                                       %do it again, Sam
  grestore
  .5 s mul 1.5 s mul translate -45 rotate .7071 dup scale %transformed user space
  pythagoreantree                                       %do it again, Sam
  }if 1 add } def
%%Endprolog
/s 50 def s 0 moveto                                    %s=size of side
11 pythagoreantree  pop grestore showpage               %depth 11
 %%EOF
```

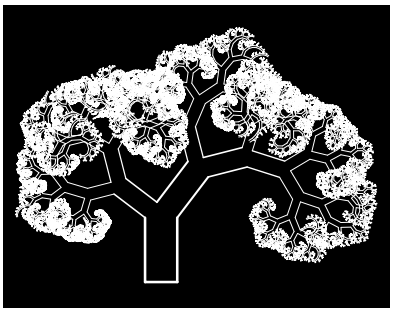




Variants







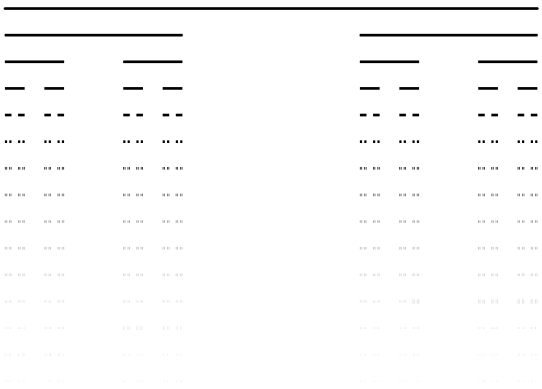
Cantor dust

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

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

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.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Cantor dust. cgl 2012, kisa1@xs4all.nl
%%BoundingBox: -1 -141 201 1
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/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
/s 200 def 0 1 14{ cd pop 0 -10 translate}for showpage
%%EOF
```



In TEX borrowed from `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 into 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).

Cantor Dust as Iterated Function System by Lauwerier(1989, ch8)

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

His tiny BASIC program and my PostScript 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
%%BoundingBox: -1 39 1010 70
%%BeginSetup
%%EndSetup
/Courier 10 selectfont 22121943 srand /x .3 def
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
```

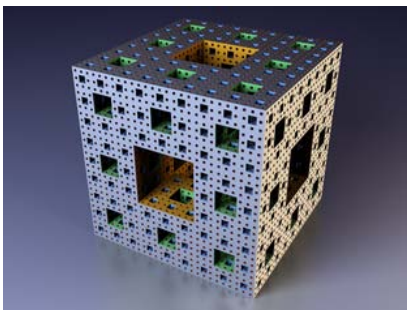
Cantor dust 2D, also called Sierpiński carpet

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 Bogusław Jackowski showed his variant, probably in connection with his `mftoeps` program, which transforms Metafont code into PostScript.

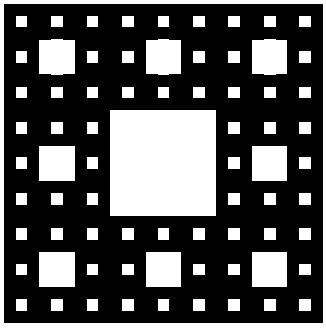
In Metafont my program of old

```
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
```



In TEX my coding of the 90-ies, which is included in `pic.dat`

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

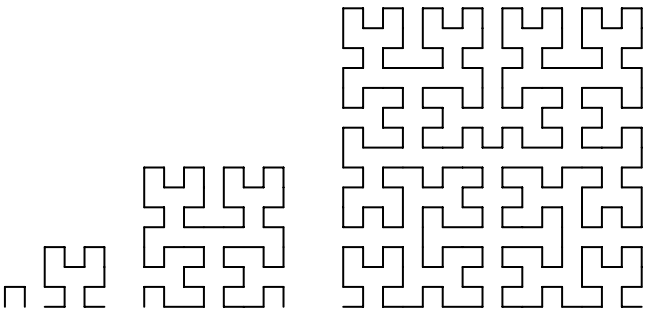


Hilbert curve

Wirth(1975) I consider a starting point for programming a Hilbert curve. The production rule of the Hilbert curve[19]

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

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, 2012. cgl, kisa1@xs4all.nl
%%BoundingBox: -1 -1 321 151
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/Hilbertdict 4 dict def
Hilbertdict begin  /-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
end
%
/Hilbert{%on stack order >=1 ==> Hilbert curve
         %s size of line segment (global),
Hilbertdict begin 1 sub dup 0 ge
{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
end} def
%%EndProlog
/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
```

Lauwerier(1990) provides a BASIC program, `Peanol`, for the Hilbert curves, restricted to orders ≤ 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.) For Wirth's(1975) program see *Classical Math fractals in PostScript*.

[19] The 'fixed-point' of the production rule is the real fractal.

Hilbert curve in Metafont and Joseph Romanovski's (1995) PostScript 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
```

Sierpiński islands

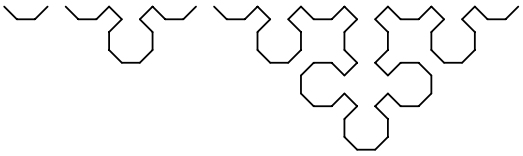
Wirth(1975) I consider a starting point for programming a Sierpiński island.

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.

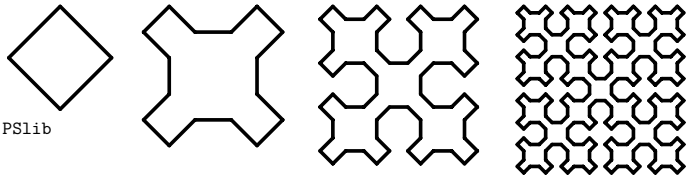
The production rule for a side

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

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.



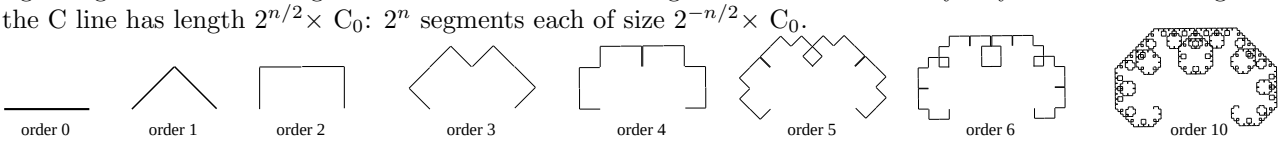
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski island Side, 2012. cgl, kisa1@xs4all.nl
%%BoundingBox:-1 -82 287 2
%%BeginSetup
%%EndSetup
%%BeginProlog
/sf {s 1.4142 div} def
/NE{0 0 moveto sf dup     lineto currentpoint stroke translate}def
/SE{0 0 moveto sf dup neg lineto currentpoint stroke translate}def
/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
```



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski island, 2012. cgl kisa1@xs4all.nl
%%BoundingBox:-10 -185 307 2
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %loads /SideS and auxiliaries from PSlib
%%EndProlog
/Sierpinskiislanddict 3 dict def
/Sierpinskiisland{Sierpinskiislanddict begin
/s 20 def /sf {s 1.4142 div} def
/SE{0 0 moveto sf dup neg lineto currentpoint stroke translate}def
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
end}def
%%EndProlog
Sierpinskiisland showpage
%%EOF
```

Lévy fractal

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

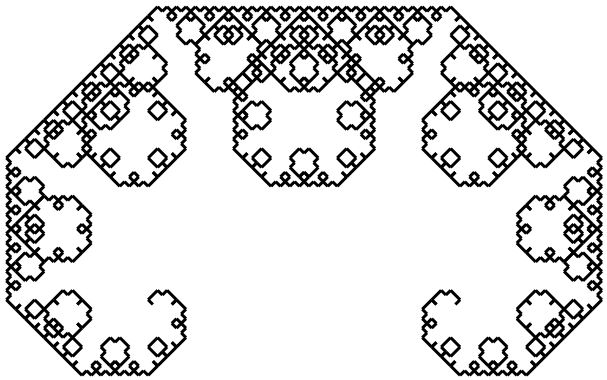


Production rule

$$C_n = [R_{45}S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})}C_{n-1}] \oplus T_{\frac{s}{2}\frac{s}{2}}\,[R_{-45}S_{(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})}C_{n-1}], \quad \text{with} \quad C_0 = \text{initial line, and}$$

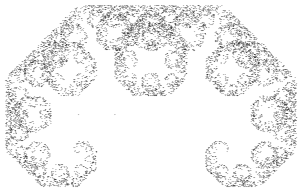
C_n the Lévy C curve of order n, $\oplus$ splice operator, meaning add properly, i.e. $T_{(\frac{s}{2},\frac{s}{2})}$, [means store graphics state on the GS stack and open a new one,] means remove current graphics state off the GS stack and recall previous, R_{45} means rotate US 45° in the PS sense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

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



Lévy fractal as Iterated Function System

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

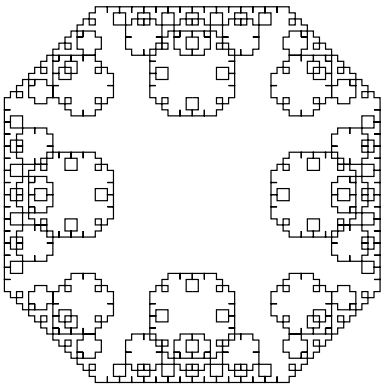


Associated with the Lévy fractal are 2 rotations with rotation centres for L: (-1,0) and for R: (1,0) and contraction $\sqrt{.5} \approx .7$. Amazing, isn't it! Laurier's BASIC program `FRACMC2` and my conversion are given below.

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

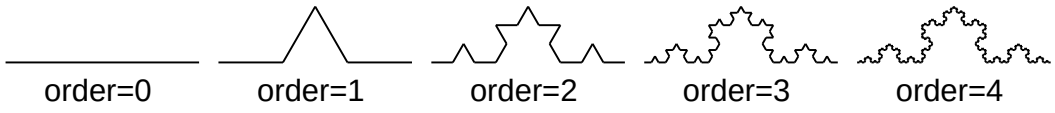
```
%!PS-Adobe-3.0 EPSF-3.0
%%Author: H.A. Lauwerier(1994): Spelen met Graphics en Fractals
%%BoundingBox: -203 -54 205 206
%%BeginSetup
%%EndSetup
%%BeginProlog
/Courier 7 selectfont
/x 0 def  /y 0 def /halfmaxint 2 30 exp  def
/a .5 def  /b a def /c a def /d a neg def
/det1 a dup mul b dup mul add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

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



von Koch fractal

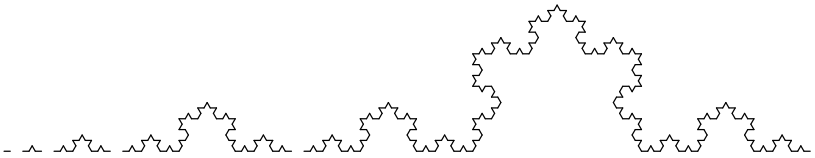
A von Koch broken line and a Lévy C line are related to a Lindenmayer system, also called a rewrite system. For the von Koch broken line the rewrite is: divide a line in 3 pieces and replace the middle piece by an equilateral triangle, with the base omitted. Repeat the process on the 4 line pieces to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



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

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

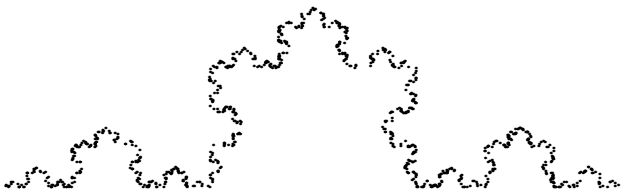
with, K_0 the initial line, K_n the von Koch curve of order n, $\oplus$ splice operator, meaning add properly, i.e. translate, [open a new GS on the GS stack,] remove current graphics state from the GS stack and recall previous, R_{60} means rotate US 60° in the PS sense, $S_{a,b}$ means scale US by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



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

Von Koch-like fractal as Iterated Function System

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



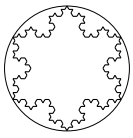
$$\begin{pmatrix} x' \\ y' \end{pmatrix} \overset{\text{L}}{=} \begin{pmatrix} .5 & .289 \\ .289 & -.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -.5 \\ .289 \end{pmatrix} \text{ and } \begin{pmatrix} x' \\ y' \end{pmatrix} \overset{\text{R}}{=} \begin{pmatrix} .5 & -.289 \\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

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

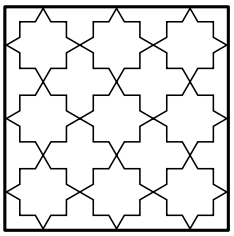
```
REM ***iteratief systeem, 2 spiegelingen, FRACMC4***
REM ***coefficienten***
A=.5 : B=..289 : C=A : D=-B
DET1=A*A+B*B : DET2=C*C+D*D : Q=DET1/(DET1+DET2)
X=1 : Y=0 : K=0 : KMAX=1000
DO WHILE K<KMAX AND INKEY$=" "
   R=RND
   IF R<Q THEN
      X1=A*X+B*Y-1+A : Y1=B*X-A*Y+B 'spiegeling L
   ELSE
 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

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

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

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



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

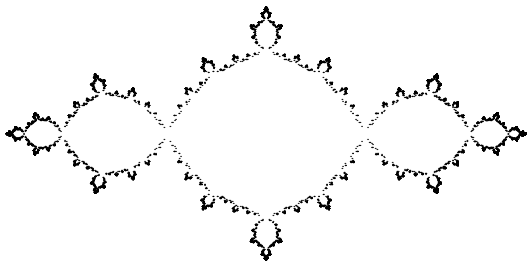


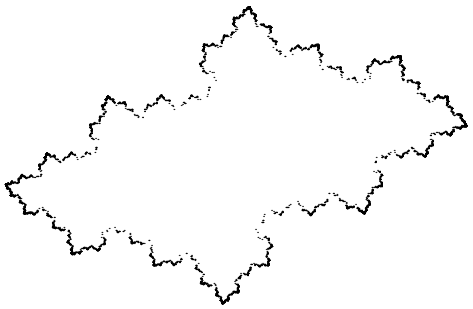
Julia fractals

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

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

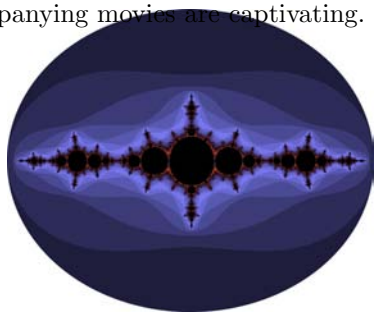
```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -185 -85 185 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/JULIAMCdict 11 dict def                 %local dictionary
/JULIAMC{%stack: a, b, maxk.
         %a+ib complex constant c,  maxk maximal iterations
         % ==> Julia set of z^2 + a + ib
JULIAMCdict begin%open the local dictionary
/kmax exch def /b exch def /a exch def  /Courier 7 selectfont
/nextpoint{/x1 x a sub 2 div def /y1 y b sub 2 div def
            x1 dup mul y1 dup mul add sqrt dup%R
           /y exch x1 sub sqrt y1 0 lt{neg}if def
           /x exch x1 add sqrt def
}bind def
/printxy{x s y s        moveto (.) centershow
    x s neg y s neg     moveto (.) centershow   %point symmetry
    b 0 eq y 0 ne and
       {x s y s neg moveto (.) centershow
        x s neg y s moveto (.) centershow}if%symmetry x- and y-axes
}bind def
/s {100 mul} def                         %scaling
/nrand rand 2147483647 div def           %random number in [0,1]
/x nrand def /y nrand def                %start values in [0,1]
10{nextpoint}repeat                      %discard 10 iterations
kmax{nextpoint
     rand 1073741823 gt {/x x neg def /y y neg def}if
     printxy
    }repeat
end}bind def
%%EndProlog
-1    0   5000 JULIAMC showpage  %SanMarco
-.59 -.34 5000 JULIAMC showpage  %cloud
%%EOF
```





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

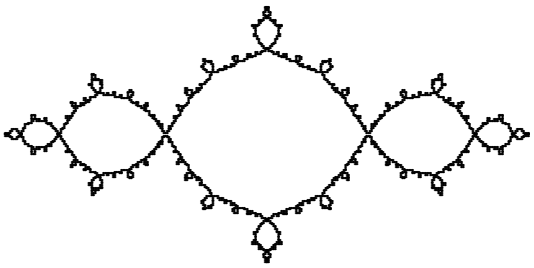
`Chaos` and `Fractalus` are packages, in the Public Domain, which create fractals with rich colour possibilities. The aqccompanying movies are captivating.



Julia fractals II: San Marco

The Boundary Scan method yields a more complete fractal. Implicit is that the BoundingBox is a factor 100 the domain of the fractal.

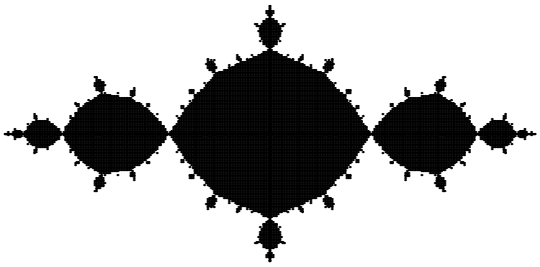
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaBoundaryScan. cgl 2012
%%BoundingBox: -165 -85 165 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-1 0 1.85 .85 80 JULIABS
stroke showpage
%%EOF
```



The San Marco fractal has no contour so filling up is special

The forward method of iterarion is used and the iteration is neglected when sequence flies to ∞.

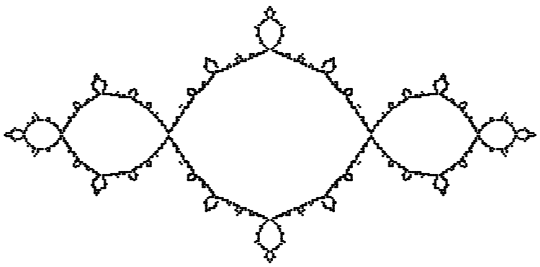
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaFilled. cgl 2012
%%BoundingBox: -185 -85 185 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-1 0   1.85 .85  50 JULIAF
stroke showpage
%%EOF
```



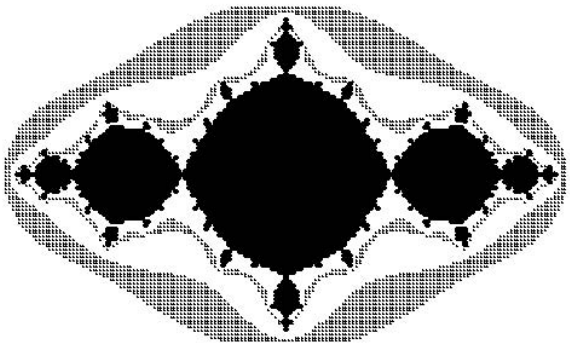
The San Marco fractal via the distance formula

The method is rumoured to yield when other methods fail.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaDistanceFormula. cgl 2012
%%BoundingBox: -185 -85 185 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-1 0 1.85 .85 90 .005 JULIAD
stroke showpage
%%EOF
```



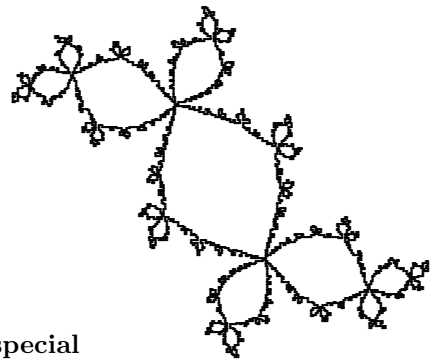
Captivating pictures can be obtained by colouring places with equal escape numbers.



Julia fractals III: Rabit of Gouady

The Boundary Scan method yields a more complete fractal. Implicit is that the BoundingBox is a factor 100 the domain of the fractal.

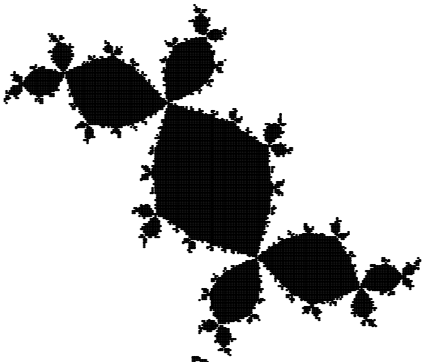
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaBoundaryScanDouady. cgl 2012
%%BoundingBox: -140 -110 140 110
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.12 .74  1.4 1.1 80 JULIABS stroke showpage
%%EOF
```



The Douady fractal has no contour so filling up is special

The forward method of iterarion is used and the iteration is neglected when the sequence flies to ∞.

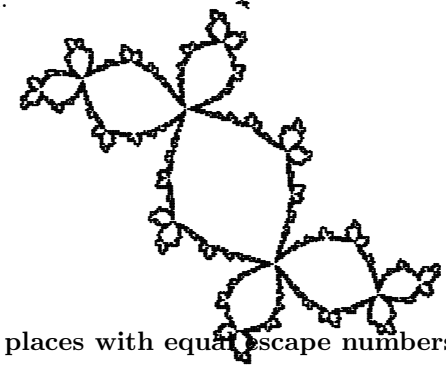
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaFDouady. cgl 2012
%%BoundingBox: -140 -110 140 110
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.12 .74  1.4 1.1 80 JULIAF stroke showpage
%%EOF
```



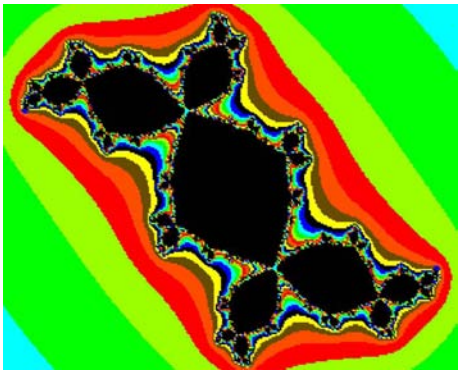
The Douady fractal via the distance formula

The method is rumoured to yield when other methods fail.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaDDouady. cgl 2012
%%BoundingBox: -140 -110 140 115
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.12 .74  1.4 1.15 50 .01 JULIAD stroke showpage
%%EOF
```



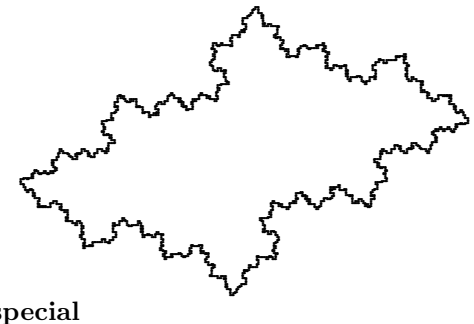
Captivating pictures can be obtained by colouring places with equal escape numbers.



Julia fractals IV: Cloud

The Boundary Scan method yields a more complete fractal. Implicit is that the BoundingBox is a factor 100 the domain of the fractal.

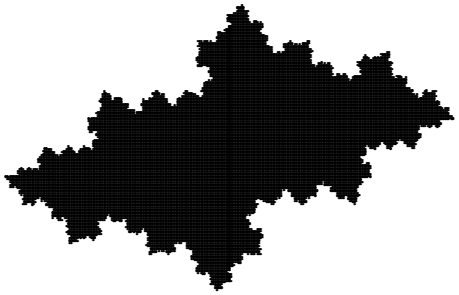
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaBSCloud. cgl 2012
%%BoundingBox: -145 -100 145 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.59 -.34  1.45 1 80 JULIABS stroke showpage
%%EOF
```



The Cloud fractal has no contour so filling up is special

The forward method of iterarion is used and the iteration is neglected when the sequence flies to ∞.

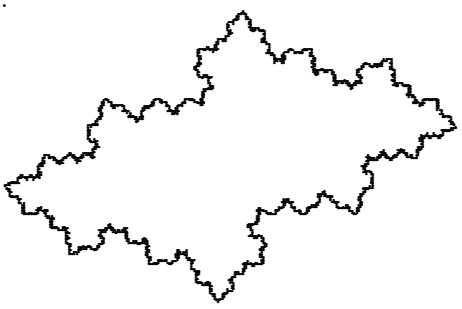
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaFCloud. cgl 2012
%%BoundingBox: -145 -100 145 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.59 -.34  1.45 1 80 JULIAF stroke showpage
%%EOF
```



The cloud fractal via the distance formula

The method is rumoured to yield when other methods fail.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaDCloudy. cgl 2012
%%BoundingBox: -175 -120 175 120
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.12 .74  1.4 1.15 50 .01 JULIAP stroke showpage
%%EOF
```



Captivating pictures can be obtained by colouring places with equal escape numbers.

Acrobat Pro 7 did not terminate?!? (Program error?) A PrtScrn of result via PSV was obtained as kludge.

Julia fractals V: Dragon

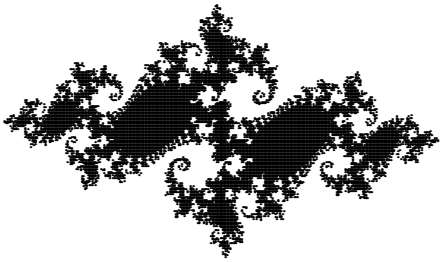
The Boundary Scan method yields a more complete fractal. Implicit is that the BoundingBox is a factor 100 the domain of the fractal.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaBoundaryScanDragon. cgl 2012
%%BoundingBox: -155 -90 155 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.7454 .11  1.55 .9 90 JULIABS stroke showpage
%%EOF
```

The Dragond fractal has no contour so filling up is special

The forward method of iterarion is used and the iteration is neglected when the sequence flies to ∞.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaFDragon. cgl 2012
%%BoundingBox: -155 -90 155 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.7454 .11  1.55 .9 90 JULIAF stroke showpage
%%EOF
```



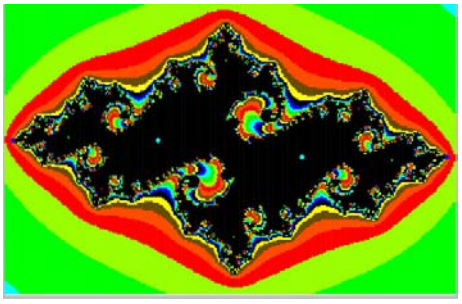
The Dragon fractal via the distance formula

The method is rumoured to yield when other methods fail.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaDDragon. cgl 2012
%%BoundingBox: -155 -90 155 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
-.7454 .11  1.55 .9 90 .01 JULIAD stroke showpage
%%EOF
```

Captivating pictures can be obtained by colouring places with equal escape numbers.

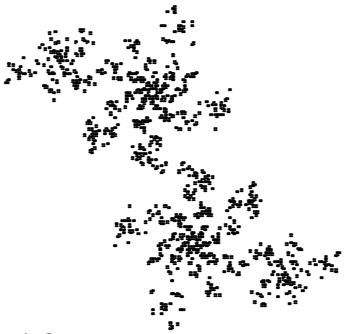
Acrobat Pro 7 did not terminate?!? (Program error?) A PrtScrn of result via PSV was obtained as kludge. `PSlib.eps` contains a black-and-white variant.



Julia fractals VI: Leaves

The Boundary Scan method yields a more complete fractal. Implicit is that the BoundingBox is a factor 100 the domain of the fractal.

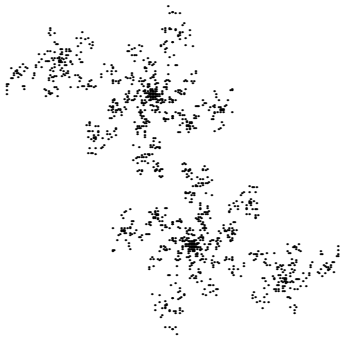
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaBSLeaves. cgl 2012
%%BoundingBox: -130 -130 130 130
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%Endprolog
.11 .66  1.3 1.3 50  JULIABS stroke showpage
%%EOF
```



The Leaves fractal has no contour so filling up is special

The forward method of iterarion is used and the iteration is neglected when the sequence flies to ∞.

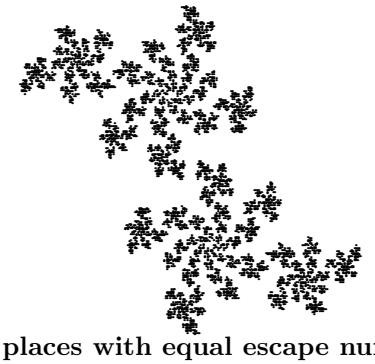
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaFLeaves. cgl 2012
%%BoundingBox: -130 -130 130 130
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
.11 .66  1.3 1.3 50  JULIAF stroke showpage
%%EOF
```



The Leaves fractal via the distance formula

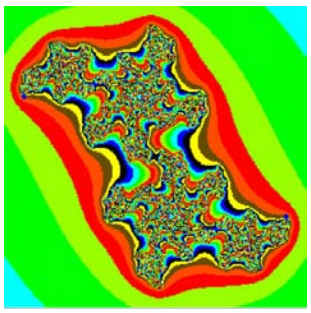
The method is rumoured to yield when other methods fail.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaDLeaves. cgl 2012
%%BoundingBox: -130 -130 130 130
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
.11 .66  1.3 1.3 50  .001 JULIAD stroke showpage
%%EOF
```



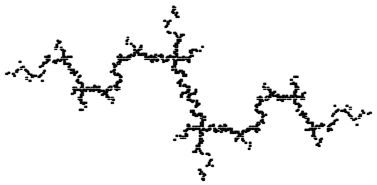
Captivating pictures can be obtained by colouring places with equal escape numbers.

Acrobat Pro 7 did not terminate?!? (Program error?) A PrtScrn of result via PSV was obtained as kludge. `PSlib.eps` contains a black-and-white variant.



Julia fractals VII: Lightning

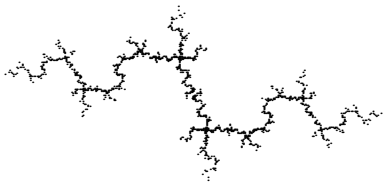
The Boundary Scan method yields a more complete fractal. Implicit is that the BoundingBox is a factor 100 the domain of the fractal.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaBSLightning. cgl 2012
%%BoundingBox: -170 -85 170 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
-1.03 .386  1.7 .85 20  JULIABS stroke showpage
%%EOF
```

The Leaves fractal has no contour so filling up is special

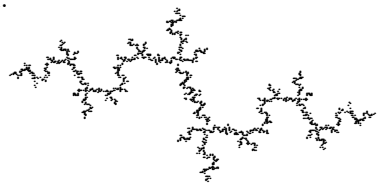
The forward method of iterarion is used and the iteration is neglected when the sequence flies to ∞.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaFLightning. cgl 2012
%%BoundingBox: -165 -85 165 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
-1.03 .386  1.65 .85 20  JULIAF stroke showpage
%%EOF
```

The Leaves fractal via the distance formula

The method is rumoured to yield when other methods fail.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: JuliaDLightning. cgl 2012
%%BoundingBox: -170 -85 170 85
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%Endprolog
-1.03 .386  1.7 .85 50 .001 JULIAD stroke showpage
%%EOF
```

Captivating pictures can be obtained by colouring places with equal escape numbers.

Acrobat Pro 7 did not terminate?!? (Program error?) A PrtScrn of result via PSV was obtained as kludge. `PSlib.eps` contains a black-and-white variant.

Circle symmetric Julia fractals I

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

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

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

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

Circle symmetric Julia fractals II

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

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

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

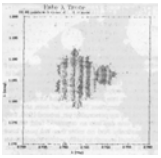
```
%PostScript-Adobe-3.0 EPSF-3.0
%%BoundingBox: 125 40 515 440
%%BeginSetup
%%EndSetup
%%DocumentFonts: Courier
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
-2.08 1 -.1 .167 150 7 5000 JULIASYMm%Mayan bracelet
showpage
%%EOF
```

More example are provided in van der Laan(2012).

Mandelbrot's fractal

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

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

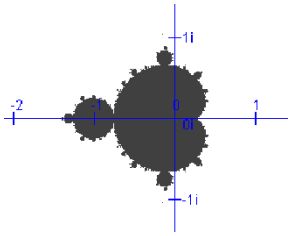


Mandelbrot's first M-fractal

Mandelbrot in 1980 answered the question:

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

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

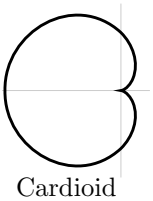


Form and size of M-fractal

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

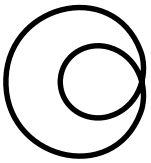
M-cardiod

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



Cardioid

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



Limaçon

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

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

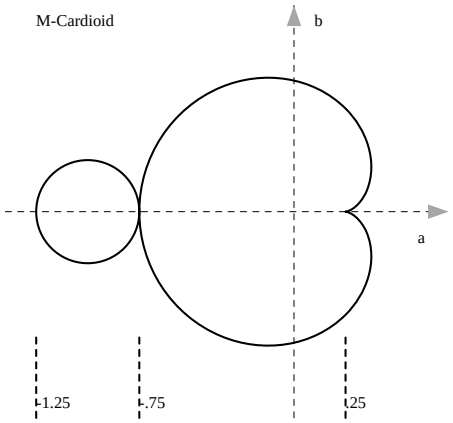
The cardioid has been draw, with the main circle centre at (-1,0) and radius .25, next to it; see accompanying picture. The cardioid is of the same size as the M-fractal cardioid.

In order to have an idea of the scale in the M-fractal picture the relevant numbers are shown underneath the drawing. The a- and b-axis have been drawn dashed.

If $|4a^2 + 8a + b^2| < 3.75$ then (a,b) lies inside the cardioid.

If $|a + ib + 1| < .25$ then (a,b) lies inside the circle.

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




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

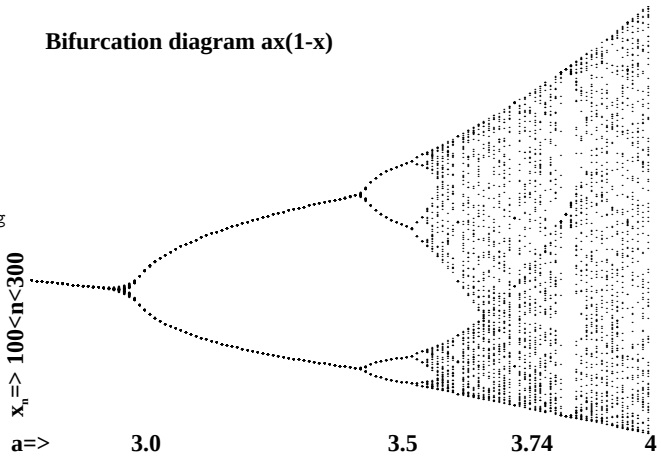
Wim W. Wilhelm communicated his compact specification for drawing the cardioid in Mathematica, using Cartesian coordinates.

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

Bifurcation diagram

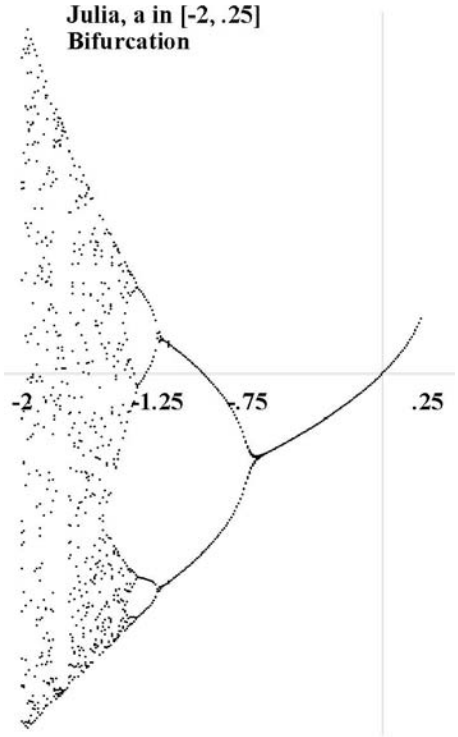
The values of the nonlinear growth model $ax(1-x)$ with starting value x=.7, converge, bifurcate in various peroids or ends up in chaos. The picture and codes has been included in orde rthat users can redo the experiment.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Author: Lauwerier(1987):
%%Title:  Fractals, meetkundige figuren in eindeloze herhaling
%%Transcriptor: Kees van der Laan, May 2012
%%BoundingBox: 825 35 1210 300
%%BeginSetup
%%EndSetup
%%DocumentFonts: Times-Bold
%%BeginProlog
%(c:\\PSlib\\Pslib.eps) run
/bifurcationdiagram{/Times-Bold 14 selectfont /sc{300 mul} def%scaling
850 275 moveto (Bifurcation diagram ax\(1-x\) )show
gsave%text along y-axis
2.8 sc 3 sub 60 moveto
90 rotate (x) show 0 -3 rmoveto (n)/Times-Bold 7 selectfont show
    0 3  rmoveto (=> 100<n<300) /Times-Bold 14 selectfont show
grestore /Times-Bold 14 selectfont
2.8 sc 10 sub 40 moveto (a=>)  show%x-axis
2.8 .01 4{/a exch def  /x .7 def%starting value
 a 3.0  sub abs .001 lt{a sc 40 moveto (3.0) show}if
 a 3.5  sub abs .001 lt{a sc 40 moveto (3.5) show}if
 a 3.74 sub abs .001 lt{a sc 40 moveto (3.74)show}if
 a 4    sub abs .001 lt{a sc 40 moveto (4)   show}if
1 2 300{/k exch def /x a x mul 1 x sub mul def%restricted growth model
         k 100 gt{a sc 300 250 x mul sub moveto (.)
                  /Courier 7 selectfont show}if}for
 }for}def
%%EndProlog
bifurcationdiagram showpage
%%EOF
```



Julia model $x_{k+1} = x_k^2 + a, k = 0, 1, \ldots$

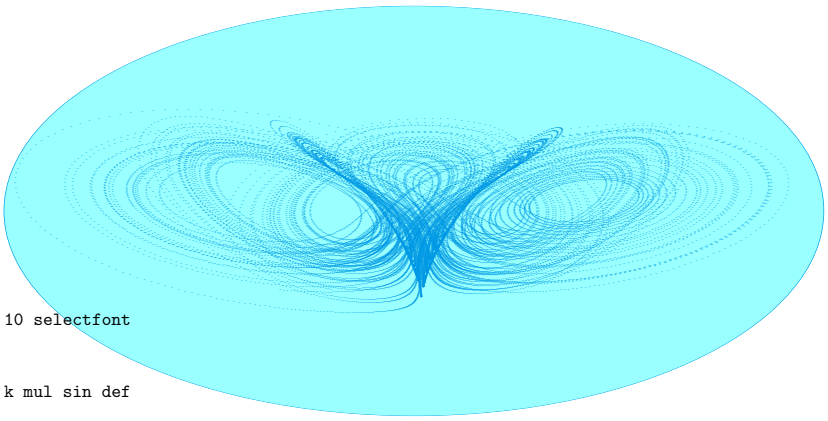
```
%!PS-Adobe-3.0  EPSF-3.0
%%Name: CollectJulia: Julia bifurcatie diagram for real c
%%Author: Kees van der Laan, June 2012
%%BoundingBox: -210 -210 40 210
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/TB14 /Times-Bold findfont 14 scalefont def
/TB7 /Times-Bold findfont 7 scalefont def
/bifurcationdiagramjulia{TB14 setfont
/sc{100 mul} def%scaling
-175 195 moveto (Julia, a in [-2, .25])show
-175 180 moveto (Bifurcation) show
gsave .9 setgray
-210 0 moveto 40 0 lineto     %x-axis
0 -200 moveto 0 200 lineto stroke%y-axis
grestore
-2 .01 .25{/a exch def  TB14 setfont
 a -2    sub abs .001 lt{a sc -20 moveto (-2)centershow}if%legend
 a -1.25 sub abs .001 lt{a sc -20 moveto (-1.25)centershow}if
 a -.75  sub abs .001 lt{a sc -20 moveto (-.75) centershow}if
 a .25   sub abs .001 lt{a sc -20 moveto (.25)  centershow}if
  /x .7 def%starting value
  1 2 125{/k exch def
        x 100000000 lt{/x x x mul a add def%Julia model
         k 100 gt{a sc x sc moveto (.)  TB7 setfont centershow}if
        }if }for%k
 }for%a
}bind def
%%EndProlog
 bifurcationdiagramjulia showpage
%%EOF
```



Lorenz attractor

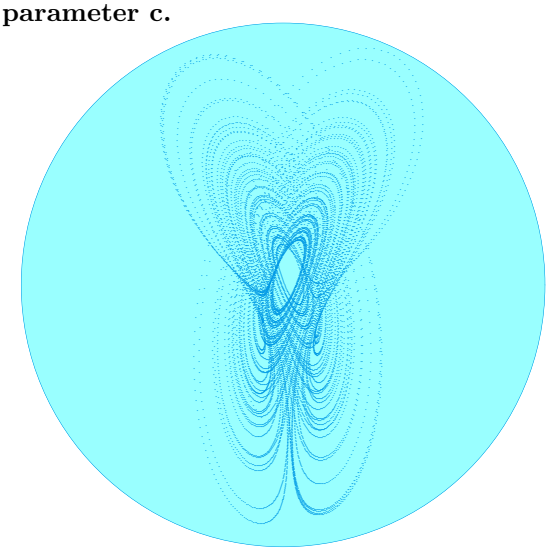
Lorenz simplified the differential equations for weather forecasting, and stumbled upun the strange attractor for the difference equations. Interactive BASIC shows the growth of the data, while in batch PostScript we only see the results, alas. What we want to know from a given sequence whether they come from an attractor, and if so, we also need to quatify this attractor in terms of Lyapunov exponents and dimension so that we may speak of a chaotic and strange attractor.

The shape of the Lorentz attractor gave rise to the name butterfly-effect.



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lorenz attractorer. cgl 2009
%%BoundingBox: -435 -110 425 360
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/lorenzdict  24 dict def
/lorenzclipped{lorenzdict begin
1 .5 scale  -5 250 425 0 360 arc
gsave .6 1 1 setrgbcolor fill grestore clip
/scale{10 mul}def 0 .6 .9 setrgbcolor /Courier 10 selectfont
/a 10 def /b 8 3 div def /c 28 def /d .003 def
/x -12.5 def /y -17.4 def /z 26 def
0 1 3 {/k exch def /cs 36 k mul cos def /ss 36 k mul sin def
10000{/u x def
      /v x cs mul y ss mul sub def /w x ss mul y cs mul add def  v w add scale z scale moveto (.) show
      /x1 x d a mul y x sub mul add def /y1 y d x c z sub mul y sub mul add def  /z1 z d x y mul b z mul sub mul add def
      /x2 x1 d a mul y1 x1 sub mul add def  /y2 y1 d x1 c z1 sub mul y1 sub mul add def /z2 z1 d x1 y1 mul b z1 mul sub mul add def
      /x x x2 add 2 div def  /y y y2 add 2 div def /z z z2 add 2 div def}repeat
}for%k
end}def
lorenzclipped stroke showpage
%%EOF
```

Variant with ptp projection and different chaos parameter c.

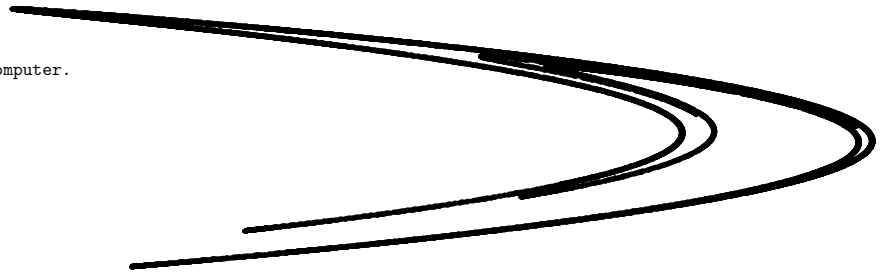


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lorenz attractor variant. cgl 2014
%%BoundingBox: -355 0 345 750
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/lorenzdict  24 dict def
/lorenzvar{lorenzdict begin
-5 425 325 0 360 arc
gsave .6 1 1 setrgbcolor fill grestore
clip
/scale{10 mul}def
0 .6 .9 setrgbcolor
/Courier 10 selectfont
/a 10 def /b 8 3 div def /c 45 def /d .003 def
/x -12.5 def /y -17.4 def /z 26 def
/theta 0 def /phi 45 def
15000{x y z ptp /v exch scale def /u exch scale def
      u v moveto (.) show
      /x1 x d a mul y x sub mul add def  /y1 y d x c z sub mul y sub mul add def  /z1 z d x y mul b z mul sub mul add def
      /x2 x1 d a mul y1 x1 sub mul add def /y2 y1 d x1 c z1 sub mul y1 sub mul add def /z2 z1 d x1 y1 mul b z1 mul sub mul add def
      /x x x2 add 2 div def  /y y y2 add 2 div def /z z z2 add 2 div def}repeat
end}def
lorenzvar stroke showpage
%%EOF
```

Hénon attractor

Hénon in 1976 found a simple 2D strange attractor

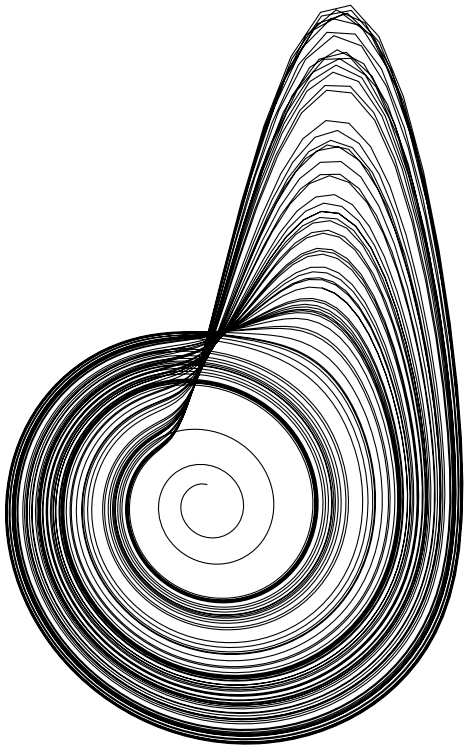
$$\begin{aligned} x_{k+1} &= y_k + 1 - ax_k^2 \\ y_{k+1} &= bx_k \end{aligned} \qquad k = 1 \ldots$$



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Henon attractor
%%Author: H A Lauwerier(1996): Chaos met de computer.
%%BoundingBox: -130 -45 130 45
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/henondict 10 dict def
/henon{henondict begin
/Courier 12 selectfont
/a 1.4 def /b .3 def /x .6 def /y .6 def
/s{ 100 mul}def
1 1 20000{/k exch def
/z x def /x 1 y add a x x mul mul sub def/y b z mul def
k 32 gt {x s y s moveto (.) centershow}if
}for
end}def
henon stroke showpage
%%EOF
```

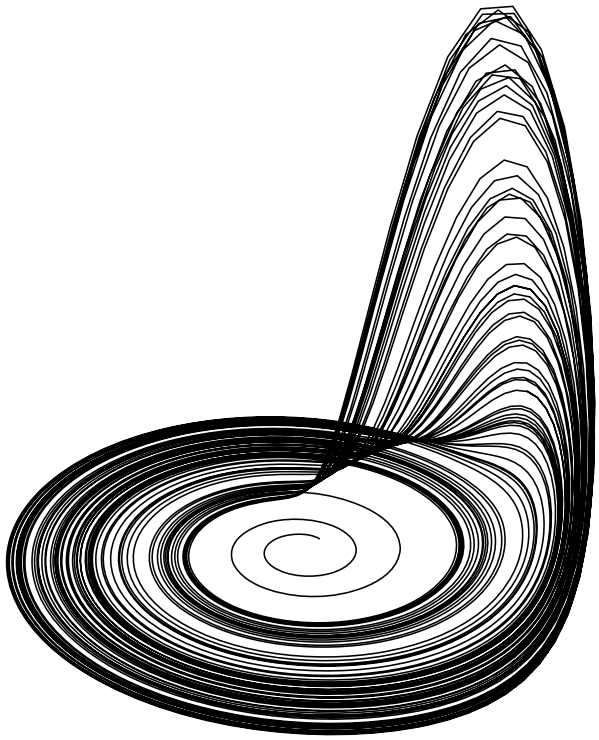
Rössler attractor

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Roessler (strange) attractor, cgl May 97
%%Author: Peitgen e.a. (1992): Chaos and fractals. Springer-Verlag
%%BoundingBox: -205 -230 245 460
%%BeginSetup
%%EndSetup
/roesslerdict 15 dict def
/roessler{roesslerdict begin /s 20 def %Scaling
%Equations: d(x,y,z)/dt=(fx,fy,fz)
/fx{y z add neg}def
/fy{x a y mul add}def
/fz{b x z mul add c z mul sub}def
/a .2 def /b .2 def /c 5.7 def %Parameters Integration
/x 0 def /y 1 def /z 0 def     %Initial values
/dt .06 def                    %Stepsize ofintegration
/newxyz{%Euler's method
/x x fx dt mul add def
/y y fy dt mul add def
/z z fz dt mul add def
}def
x s mul y s mul moveto
7500{newxyz x s mul y z add s mul lineto}repeat
stroke
end}def
roessler showpage
%%EOF
```



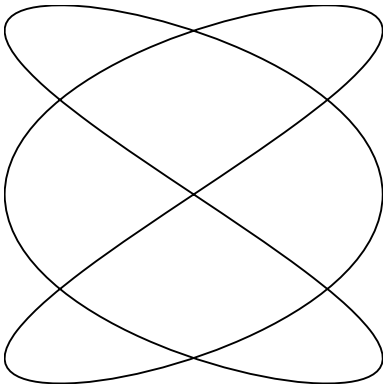
With ptp projection

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Roessler (strange) attractor, cgl May 97
%%Author: Peitgen e.a. (1992): Chaos and fractals. Springer-Verlag
%%BoundingBox: -205 -130 200 350
%%BeginSetup
%%EndSetup
(c:\\PSlib\\PSlib.eps) run
/roesslerdict 15 dict def
/roessler{roesslerdict begin /s 20 def %Scaling
/fx{y z add neg}def%Equations: d(x,y,z)/dt=(fx,fy,fz)
/fy{x a y mul add}def
/fz{b x z mul add c z mul sub}def
/a .2 def /b .2 def /c 5.7 def %Parameters Integration
/x 0 def /y 1 def /z 0 def     %Initial values
/dt .06 def                    %Stepsize of integration
/newxyz{%Euler's method
/x x fx dt mul add def
/y y fy dt mul add def
/z z fz dt mul add def}def
/theta 30 def /phi -60 def x s mul y s mul z ptp moveto
7500{newxyz x s mul y s mul z s mul ptp lineto}repeat
stroke  end}def
roessler showpage
%%EOF
```



Lissajous

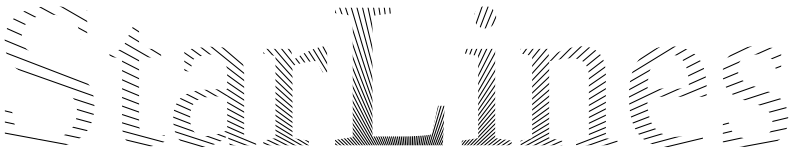
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lissajous. cgl 1997
%%BoundingBox: -100 -100 100 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/lissajousdict 4 dict def
/lissajous%n m ==> lissajous figure
{lissajousdict begin
 /m exch def /n exch def
 0 0 moveto /s {100 mul} def
 0 1 360{/t exch def
   n t mul sin s  m t mul sin s lineto}for stroke
end}bind def
%%EndProlog
3 2 lissajous showpage
%%EOF
```



Text as clipping path

Adobe's Blue book p103.

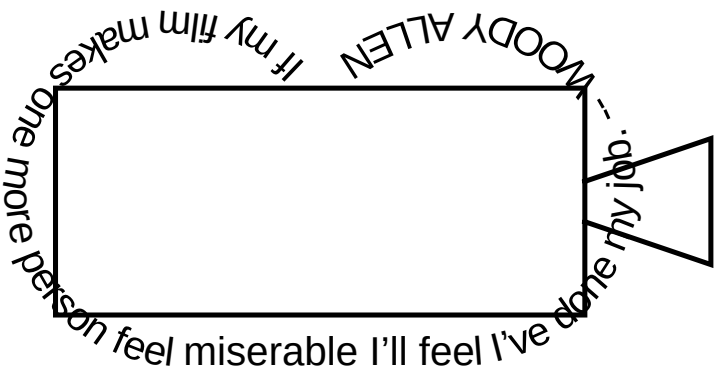
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Starlines. Blue book p103
%%BoundingBox: -1 -1 195 35
%%BeginSetup
%%EndSetup
/starlines{/Times-Roman 50 selectfont
/rays{120{0 0 moveto 108 0 lineto 1.5 rotate}repeat
      stroke}def
.25 setlinewidth
0 0  moveto (StarLines) true charpath clip
newpath 100 -15 translate rays}def
starlines
 showpage
%%EOF
```



Placing text along an explicit arbitrary path

Adobe's Blue book P11 p167. An explicit path consisting of 2 circle parts, the reels, connected by a straight line has been used. The path is approximated by `flattenpath` as a a broken line. The points of the broken line are available via the use of `pathforall`, point by point. Moreover, it is remembered what the original path was: a straight line, a curve or a spline, next to the beginning or the closing.

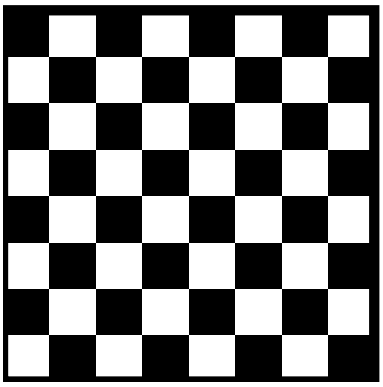
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Blue Book Program 11, p171
%%BoundingBox: 126 285 412 435
%%BeginSetup
%%EndSetup
(c:\\PSlib\\PSlib.eps) run
%Used from the library: pathtext, pathtextdict, linetoproc,
%  curvetoproc, closepathproc setchar
/Helvetica 16 selectfont 2 setlinewidth
150 310 moveto 360 310 lineto 360 400 lineto 150 400 lineto
closepath%camera
360 347 moveto 410 330 lineto 410 380 lineto 360 363 lineto
stroke%lens
%path for the text, the reels
200 360 70 0 270 arc  200 110 add 360 70 270 180 arc
(If my film makes one more person feel\
 miserable I'll feel I've done my job.\
 -- WOODY ALLEN) 55 pathtext %55 is offset
 showpage
%%EOF
```



Printing images

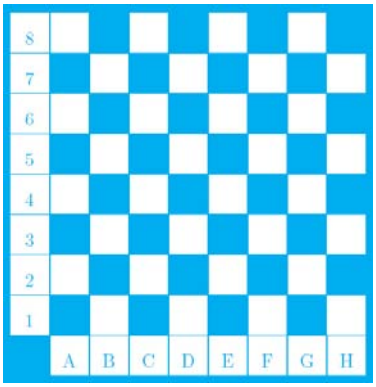
Adobe's Bluebook P6, p145. I modified the image of Adobe into a chessboard.

```
%!PS-Adobe-3.0  EPSF-3.0
%Title:  Chessboard by image(mask)
%%BoundingBox: 0 0 620 790
216 216 scale
8 8 1 [8 0 0 8 0 0] {<aa55>} image    %B&W
%blue 8 8 1 [8 0 0 8 0 0] {<aa55>} imagemask
%frame
0 0 moveto 1 0 lineto 1 1 lineto 0 1 lineto
                                  closepath
2 72 div setlinewidth blue stroke showpage
```



My programming of the chessboard goes via my crosswords macros.

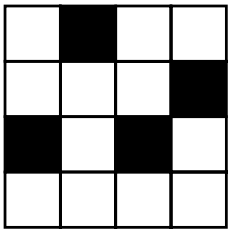
```
\begincrosswords
\obeyspaces\let =\space\csize=3ex
\bdata
8 * * * *
7* * * *
6 * * * *
5* * * *
4 * * * *
3* * * *
2 * * * *
1* * * *
*ABCDEFGH
\edata
\sol
\endcrosswords
```



Of course it is easier just to put black and white boxes next to each other. I had the `crosswords` environment at hand and misused it for the purpose.

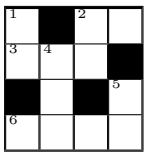
Crosswords

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Crossword. cgl 2012.
%%BoundingBox: -1 -61 81 21
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/crossworddict 3 dict def
/crossword{crossworddict begin
/x{0 0 20 20 rectstroke 20 0 translate} def
/X{0 0 20 20 rectstroke 0 0 20 20 rectfill 20 0 translate} def
/crl{-80 -20 translate 0 0 moveto} def
x X x x crl              %X denotes black and x denotes white
x x x X crl
X x X x crl
x x x x crl
end} def
%%EndProlog
crossword showpage
%%EOF
```



Typesetting Crosswords via TEX, MAPS 8, 1992

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



Across
2 Switch mode
3 Knuth
6 Prior to TEX

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


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

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

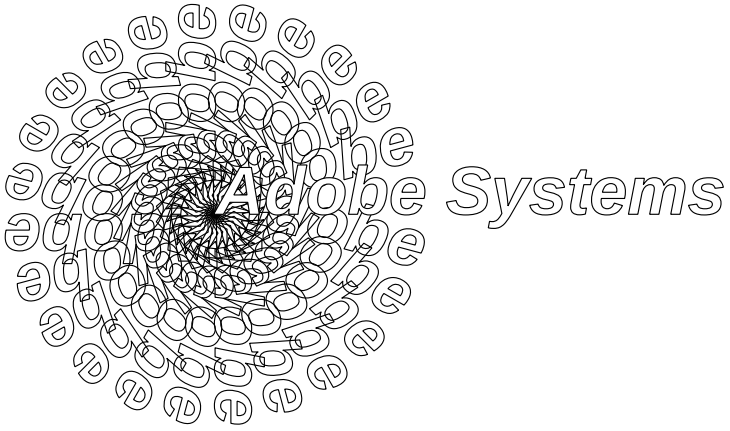
The above variant via PostScript inspired by David Byram–Wigfield,[20] who created a special font `QuadFont`, unnecessary, but interesting in itself, for the black and white squares. But ... without numbers for the clues and no toggling of solution and puzzle. In my PostScript version I simplified, without creating opr using `QuadFont`. TEX's version I consider superior.

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

Adobe Systems

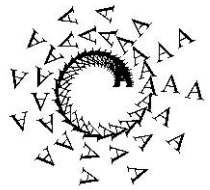
Interesting use of `oshow`.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Circle of Adobe. Blue book p98
%%BoundingBox: -100 -100 230 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/AdobeSystems{/Helvetica-BoldOblique 30 selectfont
%/oshow%stack: (string)
%{true charpath stroke}def
/circleofAdobe{ 15 15 345
{gsave rotate 0 0 moveto (Adobe) oshow grestore}for}def
.5 setlinewidth circleofAdobe
0 0 moveto (Adobe Systems) true charpath
gsave 1 setgray fill grestore}def
%%EndProlog
AdobeSystems stroke showpage
%%EOF
```



I imitated Voß' example of rotated `A`'s, more-or-less, as variant of Adobe's rotated word `Adobe`, Blue book p98. Vo31' picture has also been supplied in the LaTeX Graphics Companion p357.[21]

In `PSTricks`' code is too much one has to remember to my taste, too many and too varied braces, {...} , (...), and [...] ... moreover, the data `A` has to be supplied three times. Personally, I abhor the (curly) braces mania, and favour minimal mark-up; providing the data `A` three times is not minimal.



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

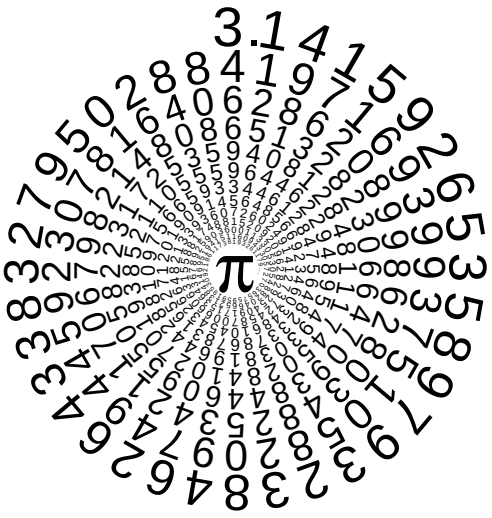
```
\begin{pspicture}(4.5, 3.5)
   \cnode*(2,2){4pt}{A}
   \multido{\nA=0+10. \rB=)+0.5}{110}{%
      \nput[rot=\nA, labelsep\rB pt]%
           {\nA}{A}{A}}
\end{pspicture}
```

[21] Another example of text along a spiral, explicit, is on p451 of the Graphics Companion, which comes close to typesetting along an implicit spiral.

Pi decimals

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

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pi-decimals along a  Spiral cgl 2010, 2012
%%BoundingBox: -80 -100 100 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%Endprolog
/Symbol 26 selectfont
1 -18 moveto (p) show%p denotes pi in the symbol font
/Helvetica 20 selectfont
0 70 moveto (3) show 1 0 rmoveto (.) show -2 0 rmoveto
-10 rotate .995 dup scale
%
{pop pop -10 rotate 3 0 rmoveto .995 dup scale}
(3.1415926535897932384626433832795028841971693993751\
05820974944592307816406286208998628034825342117067\
98214808651328230664709384460955058223172535940\
81284811174502841027019385211055596446229489549303\
81964428810975665933446128475648233786783165271201\
90914564856692346034861045432664821339360726024\
91412737245870066063155881748815209209628292540917\
15364367892590360011330530548820466521384146951941\
51160943305727036575959195309218611738193261179...) kshow
showpage
%%EOF
```

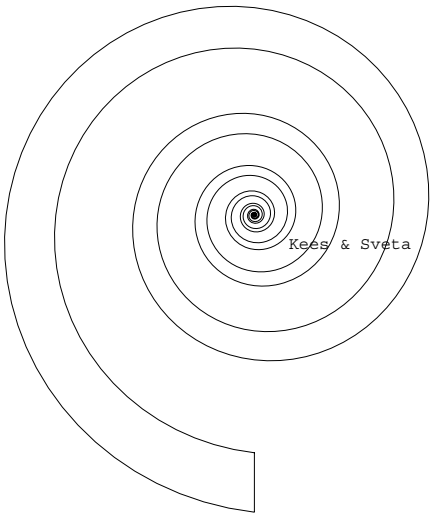


Voß in his *PSTricks — Graphics and PostScript for TEX and LaTEX, p294* shows a variant π-decimals in `PSTricks` in LaTEX.

The original picture from the CWI calendar of 1972

Log spiral

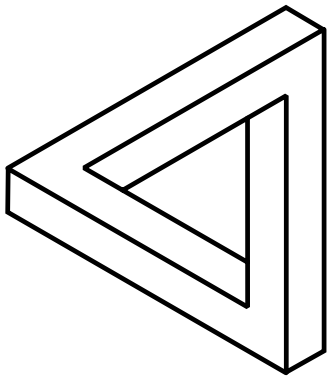
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Log spiral. cgl 2009
%%BoundingBox: -315 -350 235 270
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/logspiral load 0 10 dict put
/logspiral%cgl  logarithmic or grow spiral
{0 begin gsave
/Courier findfont 20 scalefont setfont
/a .00019 def /b .002 def /e 2.714 def
a 0 moveto
100 5 7110{/t exch def
   /r a e b t mul exp mul def
   r t cos mul    % x on stack
   r t sin mul    % y on stack
   lineto
}for
currentpoint /yc exch def /xc exch def
a 0 moveto
/a a 1.25 mul def
100 5 7110{/t exch def
   /r a e b t mul exp mul def
   r t cos mul    % x on stack
   r t sin mul    % y on stack
   lineto
}for
xc yc lineto
stroke
40 -40 moveto (Kees & Sveta) show
grestore end}def
logspiral stroke showpage
%%EOF
```



Escher impossible triangle

A triviality to program once the symmetry has been discovered.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escher's impossible triangle
%%BoundingBox: -45 -40 31 40
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/eschertriangle{1.43 setmiterlimit
3{25  34 moveto
  25 -34 lineto
  17 -38.5 lineto
  17  20 lineto
 -17.6 0 lineto
  120 rotate}repeat
stroke}def
%%EndProlog
eschertriangle showpage
%%EOF
```

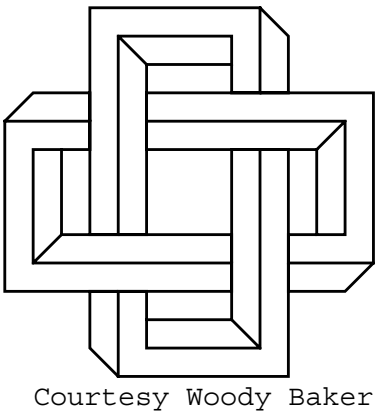


Joseph Romanovski communicated his impossible impression

```
%!PS-Adobe-3.0  EPSF-3.0
%%Creator: J.V. Romanovsky, 1996
%%BoundingBox: -130 -201 250 201
%%Beginsetup
%%EndSetup
%%BeginProlog
/romanovskidict 9 dict def
romanovskidict begin
/rl{rlineto}def  %  to reduce the code
/FS {gsave setgray fill grestore stroke}def
/s3 0.75 sqrt def /unit 40 def 1.43 setmiterlimit
   % Two constants, 'unit'  defines the size of construction
/U { unit mul 0 exch rl} def
   % Verical line for several units
/R { unit mul  % Transfer the number of units to length
     dup       % Copy it
     s3 mul    % Calculate X-offset
     exch      % Hide it
     0.5 mul   % Calculate Y-offset
     rl} def   % Draw the line with the given offset pair
/L { unit mul dup s3 mul neg  % neg is the only modificaion
     exch 0.5 mul rl}def
   % Right and left lines for several lines
/M{ s3 unit mul 0.5 unit mul translate -120 rotate} def
   % Rotation of the picture to scan the vertices of the central triangle
/P1{ 0 0 moveto 2 R -1 L -2 U 4 R -1 U -9 R 4 U -1 L -2 U
     3 R 4 U -4 R 3 U -1 L -2 U -1 L} def
end
/romanovski{romanovskidict begin
% Three parts of the picture are made with the same procedure
P1 0.6 FS M P1 0.8 FS M P1 0.95 FS
end}def
%%EndProlog
%mark origin %0  10 moveto 0 -10 lineto 10  0 moveto stroke
romanovski showpage
%%EOF
```

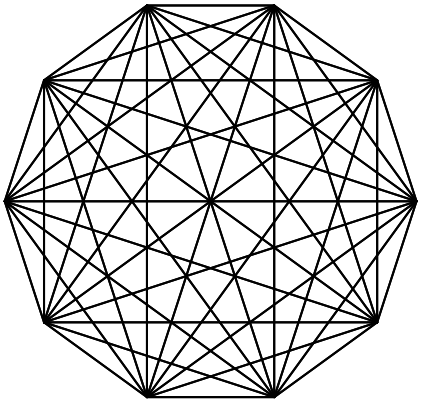
Baker's knot

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escher impossible triangle
%%BoundingBox: -45 -40 31 40
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/bakerknot{/Courier 10 selectfont
4{-15 25 moveto  0 -10 rlineto 60 0 rlineto   0 -30 rlineto
   10 0 rlineto  0 40 rlineto   -70 0 rlineto    0 10 rlineto
   80 0 rlineto  0 -60 rlineto  -30 0 rlineto   0 10 rlineto 10 0 rlineto
   35 -25 moveto  0 -10 rlineto    20 0 rlineto    10 10 rlineto
   45 15 moveto  10 10 rlineto 90 rotate
}repeat stroke}def
%%EndProlog
bakerknot
-55 -75 moveto (Courtesy Woody Baker)show  showpage
%%EOF
```



Bentley's double loop

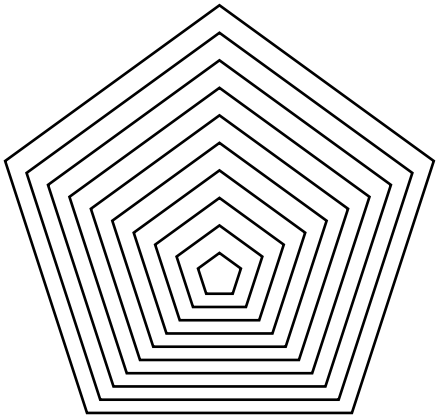
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Bentley's double loop
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
10{ 1 1 9{100 0 moveto gsave
          36 mul rotate
          100 0 lineto stroke grestore
         }for 36 rotate
  }repeat
%%EndProlog
showpage
%%EOF
```



Nested pentagons, another double loop

As compared with *Juat a little bit of PostScript* the program has been changed a little, the library `gonstar` is invoked.

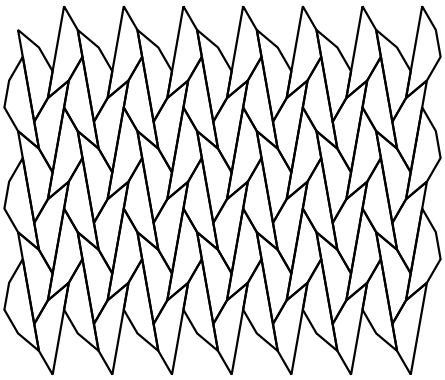
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: nested pentagons
%%BoundingBox: -86 -51 86 106
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/nestedpentagons{% l global for gonstar
10 10 100{/l exch def 5 -5 translate
          72 5 gonstar stroke}for
}def
%%EndProlog
nestedpentagons showpage
%%EOF
```



Stained-glass

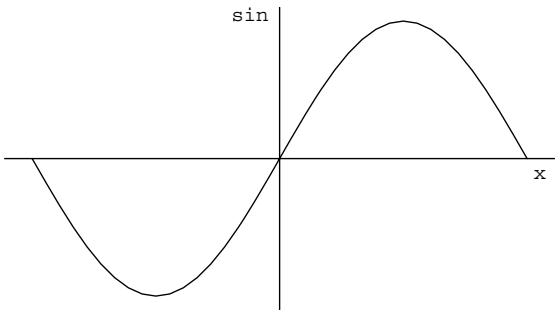
An intriguing pattern which invites for colouring.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Stained-glass. cgl 2009
%%BoundingBox: -17.5 -12.5 190 160
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/stainedglassdict 7 dict def
/stainedglass{stainedglassdict begin %tilxxv
/s 12.5 def %parameter of the figure
            %smallest side monohedra
/hs s 2 div def /mhs hs neg def
/hss 1.723 hs mul def
/long 40 cos 20 cos add s mul 2 mul def
/element{gsave 0 0 moveto
   40 rotate  0 s lineto currentpoint translate
  -20 rotate  0 s lineto currentpoint translate
  -40 rotate  0 s lineto currentpoint translate
  -20 rotate  0 s lineto closepath stroke
grestore}def
/reflectedelement{gsave
   -1 1 scale element
grestore}def
%
0 setlinejoin 1 setlinecap
10 rotate
7{gsave
  3{element
  gsave 0 s translate reflectedelement grestore
  gsave 20 sin s mul 20 cos s mul neg translate
        -20 rotate
        element
  grestore
  gsave s 40 sin mul s 40 cos 1 add mul translate
        -20 rotate
        reflectedelement
  grestore
  s 40 sin mul long s 1 40 cos sub mul add translate
  }repeat grestore
  2 40 sin mul .866 add s mul
  s 1.5 40 cos 2 mul add mul long sub translate
}repeat
end}def
%%EndProlog
stainedglass  showpage
%%EOF
```



Sine function

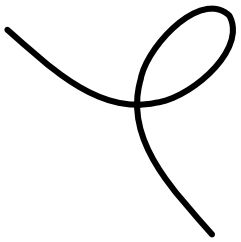
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Sine function
%%BoundingBox: -200 -110  200 110
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/sine{/Courier 15 selectfont
%x-axes with label
-200 0 moveto 200 0 lineto -15 -15 rmoveto (x) show
%y-axes with label
0 -110 moveto 0 110 lineto -35 -10 rmoveto (sin) show
%function
-180 0 moveto -180 10 180{dup sin 100 mul lineto}for
stroke}def
%%EOF
```



Folium of Descartes

In Cartesian coordinates: $x^3 + y^3 + 3axy = 0$. In Polar coordinates: $\frac{3a \sin\theta \cos\theta}{\sin^3\theta + \cos^3\theta}$.

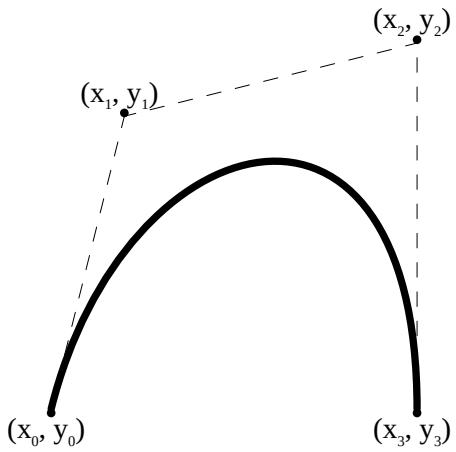
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Folium of Descartes. cgl 2012
%%BoundingBox: -30 -30 30 20
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/foliumDescartesdict 5 dict def
/foliumDescartes{foliumDescartesdict begin /3a 30 def
/r {3a t sin t cos  mul mul
       t sin dup dup mul mul
       t cos dup dup mul mul add div} def %radius vector
/t -30 def r t cos mul  r t sin mul moveto%start left
-30  2 120{/t exch def
    r t cos mul r t sin mul lineto}for stroke
end} bind def
foliumDescartes stroke showpage
%%EOF
```



Curveto from PostScript Red book p140

The picture illustrates how to add texts, with subscripts, to a graph, as well as how to obtain dashed lines. Metapost allows the inclusion of TEX marked up texts.

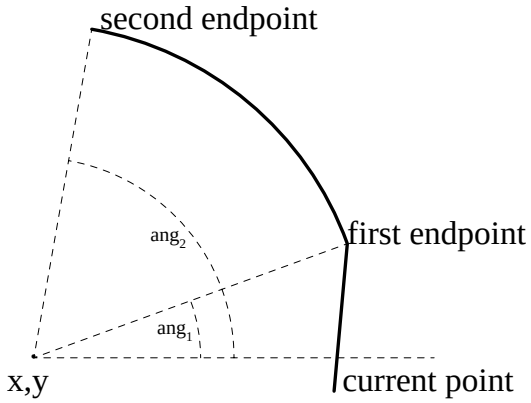
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: curveto Redbook p140
%%BoundingBox: -6 -6 56 56
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/curvetopicturedict 9 dict def
/curvetopicture{curvetopicturedict begin
/x0  0 def /y0 0 def
/x1 10 def /y1 40 def
/x2 50 def /y2 50 def
/x3 50 def /y3 0 def

x0 y0 moveto x1 y1 x2 y2 x3 y3 curveto stroke
.1 setlinewidth
x0 y0 moveto x1 y1 lineto [2] centerdash stroke
x1 y1 moveto x2 y2 lineto [2] centerdash stroke
x2 y2 moveto x3 y3 lineto [2] centerdash stroke
/Times-Roman 10 selectfont
x0 y0 1 sub moveto (.) centershow
x1 y1       moveto (.) centershow
x2 y2       moveto (.) centershow
x3 y3 1 sub moveto (.) centershow
-6 -4 moveto (\(x) /Times-Roman 4 selectfont show
  0 -1 rmoveto (0) /Times-Roman 2 selectfont show
       0 1 rmoveto (, y) /Times-Roman 4 selectfont show
  0 -1 rmoveto (0) /Times-Roman 2 selectfont show
       0 1 rmoveto (\)) /Times-Roman 4 selectfont show
44 -4 moveto (\(x) /Times-Roman 4 selectfont show
  0 -1 rmoveto (3) /Times-Roman 2 selectfont show
       0 1 rmoveto (, y) /Times-Roman 4 selectfont show
  0 -1 rmoveto (3) /Times-Roman 2 selectfont show
       0 1 rmoveto (\)) /Times-Roman 4 selectfont show
x1 6 sub y1 2 add moveto (\(x) /Times-Roman 4 selectfont show
  0 -1 rmoveto (1) /Times-Roman 2 selectfont show
       0 1 rmoveto (, y) /Times-Roman 4 selectfont show
  0 -1 rmoveto (1) /Times-Roman 2 selectfont show
      0 1 rmoveto (\)) /Times-Roman 4 selectfont show
x2 6 sub y2 2 add moveto (\(x) /Times-Roman 4 selectfont show
  0 -1 rmoveto (2) /Times-Roman 2 selectfont show
       0 1 rmoveto (, y) /Times-Roman 4 selectfont show
  0 -1 rmoveto (2) /Times-Roman 2 selectfont show
       0 1 rmoveto (\)) /Times-Roman 4 selectfont show
end} def
%%EndProlog
curvetopicture showpage
%%EOF
```



Arc from PostScript Red book p117

The picture illustrates how to add texts, with subscripts, to a graph, as well as how to obtain dashed lines. Metapost allows the inclusion of TEX marked up texts.

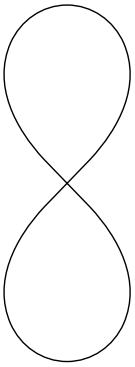
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Cantor dust

%%BoundingBox: -16 -21 150 115
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/arcpicturedict 6 dict def
/arcpicture{arcpicturedict begin
/Times-Roman 10 selectfont
/r 100 def
/arg1 20 def /arg2 80 def
origin moveto (.) centershow -8 -10 moveto (x,y) show
r 10 sub -10 moveto gsave ( current point) show grestore%currentpoint
0 0 r arg1 arg2 arc stroke
/first{ r arg1 cos mul r arg1 sin mul} def
/second{r arg2 cos mul r arg2 sin mul} def
first moveto (first endpoint) show
second moveto ( second endpoint) show
37 7 moveto (ang) /Times-Roman 6 selectfont show
     0 -2 rmoveto (1) /Times-Roman 4 selectfont show
35 35 moveto (ang) /Times-Roman 6 selectfont show
     0 -2 rmoveto (2) /Times-Roman 4 selectfont show
.1 setlinewidth%dashed lines
origin moveto r 20 add 0 lineto [2] centerdash stroke
origin moveto second lineto [2] centerdash stroke
origin moveto first lineto [2] centerdash stroke
first  r 0 origin 50 anglemark
second  r 0 origin 60 anglemark
end}def
%%EndProlog
arcpicture showpage
%%EOF
```



Lemniscate

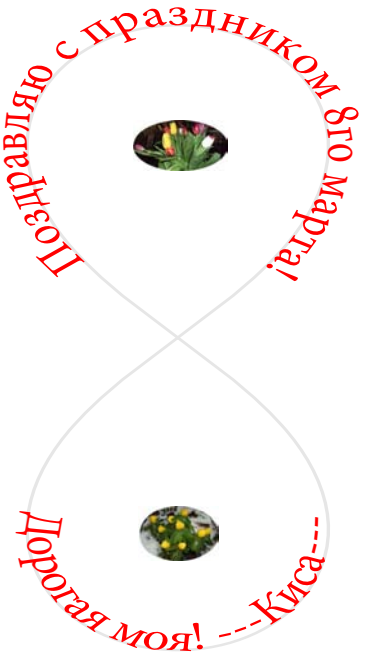
The equation for the lemniscate in polar coordinates (ϕ, r) reads $r^2 = 2a^2 \cos 2\phi$, a a constant.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Lemniscate
%%BoundingBox: -200 -110  200 110
%%BeginSetup
%%EndSetup
%%BeginProlog
/lemniscaatdict 7 dict def
/lemniscaat{lemniscaatdict begin
/a 90 def 90 rotate 0 0 moveto
-45 1 45{/t exch def
/r sqrt2 a mul 2 t mul cos sqrt mul def
/x r t cos mul def
/y r t sin mul def
x y lineto
}for
-45 1 45{/t exch def
/r sqrt2 a mul 2 t mul cos sqrt mul def
/x r t cos mul neg def
/y r t sin mul def
x y lineto
}for stroke
end}def
%%EndProlog
lemniscaat showpage
%%EOF
```



A present on occasion of the 8^{th} March

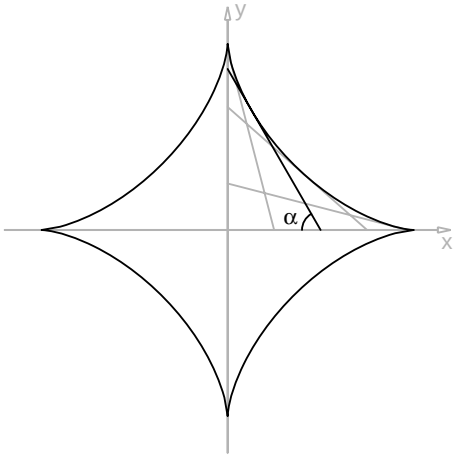
Interesting is how to include `.jpg` pictures in your `.eps`, next to printing text (Cyrillic) along an explicit path.



Astroid

Interesting are the definitions of x-axis and y-axis in the dictionary with the use of the library operator `arrow`. The use of anglemark is nice.`H12p` is used and activated by `setfont`. α is used from `S12pt`.

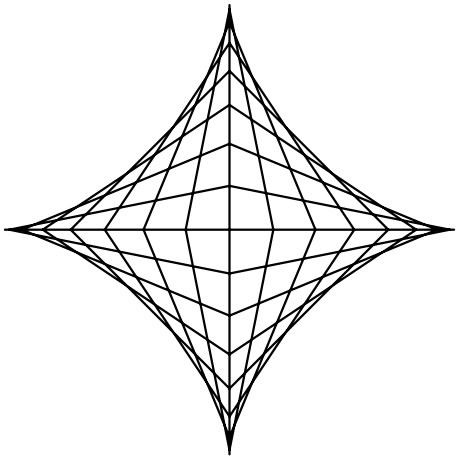
```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -121 -122 121 122
%%BeginSetup
%%EndSetup
(c:\\PSlib\\PSlib.eps) run
/astroiddict 5 dict def
astroiddict begin /origin{ 0 0 } def
/x { 1.2 r mul 0   } def
/y { 0         1.2 r mul} def
/x-axis {-1.2 r mul 0          x .1 3 7 arrow} def
/y-axis { 0         -1.2 r mul y .1 3 7 arrow} def
end
/astroid{astroiddict begin
/r 100 def%scale
.7 setgray x-axis y-axis stroke
x moveto -5 -10 rmoveto (x) H12pt setfont show
y moveto  4  -5 rmoveto (y)               show
.25 r mul 0         moveto
0         .25 r mul 15 sqrt mul lineto
.75 r mul 0         moveto
0         .25 r mul 7 sqrt mul lineto
.25 r mul 15 sqrt mul 0         moveto
0         .25 r mul lineto
stroke
%
0 setgray
4{r 0 moveto
  1 1 90{/t exch def
    /x t cos r mul def
    r log .6667 mul 10 exch exp %r^(2/3)
    x log .6667 mul 10 exch exp %x^(2/3}
    sub sqrt dup dup mul mul
    x exch lineto
    }for
  90 rotate}bind repeat
%
.5 r mul 0 moveto 0 .5 r mul sqrt3 mul lineto stroke

0 0 0 .5 r mul sqrt3 mul .5 r mul 0 10 anglemark
.5 r mul  20 sub 3 moveto (a) S12pt setfont show % alpha
end}def
%%EndProlog
astroid showpage
%%EOF
```



Astroid as spurious envelope

No formula for the Astroid has been used.

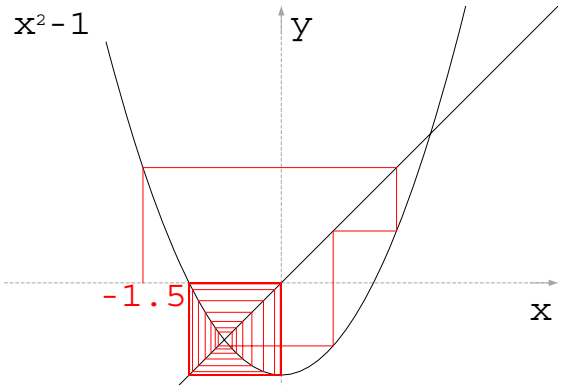
```
%!PS-Adobe-3.0  EPSF-3.0
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/astroidhull%n (>=32) ==> astroide
{/n exch def
 0 1 n 1 sub{/t exch 360 mul n div def
   t cos s mul 0 moveto   0 t sin s mul lineto}for stroke
}bind def
%%EndProlog
/s 100 def 32 astroidhull showpage
%%EOF
```

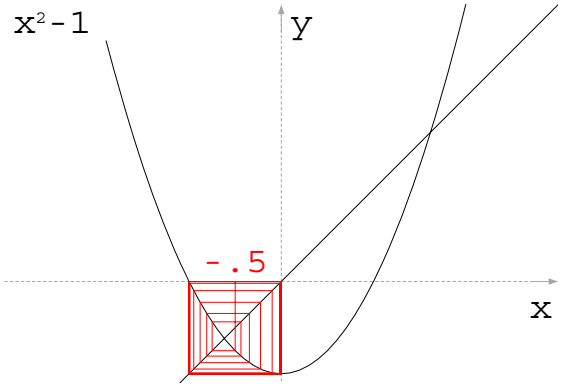


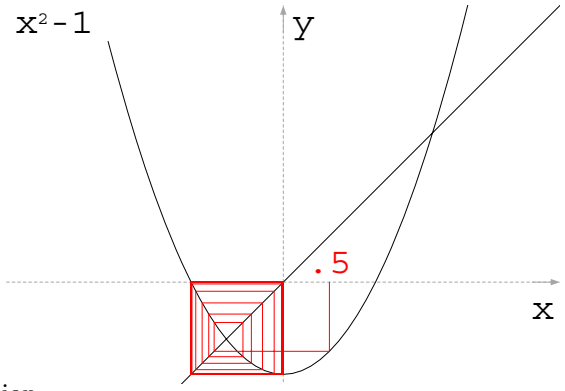
Julia model iteration

This picture is*not* about the calculation of the intersection points of the straight line and parabola. It is about behaviour of the sequence of iterates $\{x_k^2 - 1\}, k = k_0, \ldots$. The iterates assemble at the strange attractors 0, and -1.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: bifurcation of parabola and straight line. cgl 2012
%%BoundingBox: -300 -110 300 300
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/xy-diagram{
%coordinate axes
.65 setgray [3] 0 setdash
xmin 0 moveto xmax 0 lineto
0 xmin moveto 0 xmax lineto stroke
1 setlinewidth [] 0 setdash 0 setgray
xmax .9 mul -40 moveto (x) show
10 xmax .9 mul moveto (y) show
%arrowheads coordinate axes
.65 setgray
0 xmax .9 mul 0 xmax 1 7 10 arrow fill %y-axis
xmax .9 mul 0 xmax 0 1 7 10 arrow fill %x-axis
%
0 setgray%black
[] 0 setdash %normal
xmin dup moveto xmax dup lineto stroke%diagonal y=x
-1.9 s  dup -1.9 mul 100 sub moveto
-1.8 .1 2.2 {/x exch def
    x s dup dup x mul 100 sub lineto
   }for stroke
}bind def
/C40 /Courier findfont 40 scalefont def
/C20 /Courier findfont 20 scalefont def
%%EndProlog
%
% Program
%
/s {100 mul} def
/xmin -3 s def /xmax 3 s def C40 setfont
xy-diagram
%iteration
10 xmin add xmax .9 mul moveto (x) show
    0 10 rmoveto (2) C20 setfont show
    0 -10 rmoveto (-1) C40 setfont show %<---
red
-150 -25 moveto (-1.5) centershow            %<---
    /n 25 def
/x -1.5 def /f{x dup mul 1 sub}def%x^2-1  %<---
x s 0 moveto
.5 setlinewidth
n{/y f def
   x s y s lineto
   y s y s lineto
   /x y def
}repeat stroke
showpage
%%EOF
```







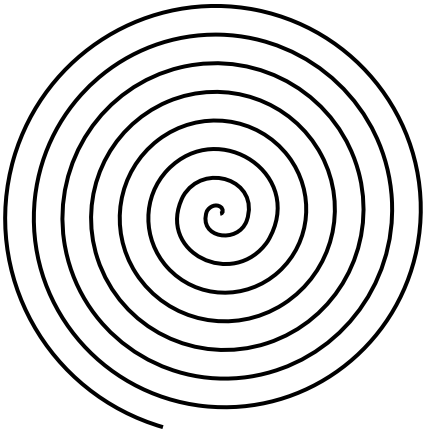
xy-diagram is contained in `PSlib.eps` for experimentation.

Spirals: Archimedian and Growth

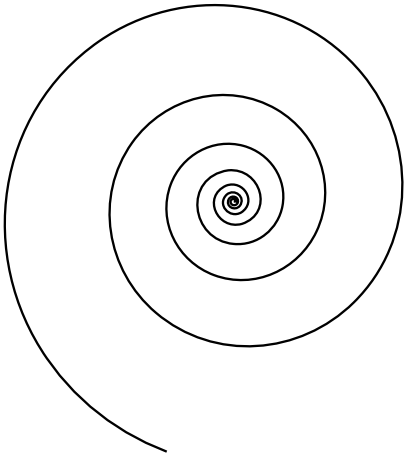
The Archimedes Spiral is in polar coordinates defined by $r_\theta = k\theta, \quad 0 \leq \theta < \infty$.

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

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

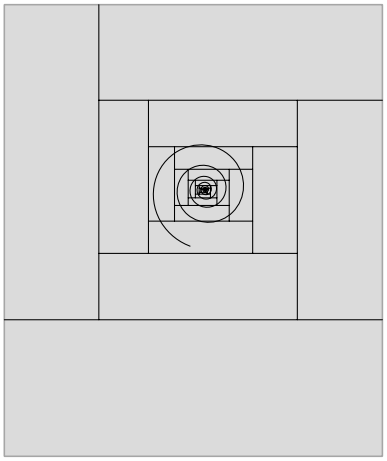


```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth Spiral
%%BoundingBox: -100 -110 75 86
%%BeginSetup
%%EndSetup
/growthspiralpc{1 0 moveto%off the 0
/f 2.718 .0085 exp def /r 1 def
555{5 rotate /r r f mul def
    r 0 lineto}repeat stroke
}bind def
%%EndProlog
growthspirapc
```



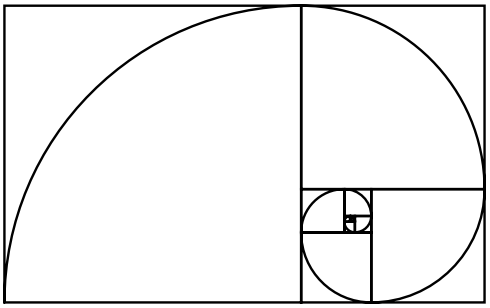
The Gyre Open font Type activity has logo

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



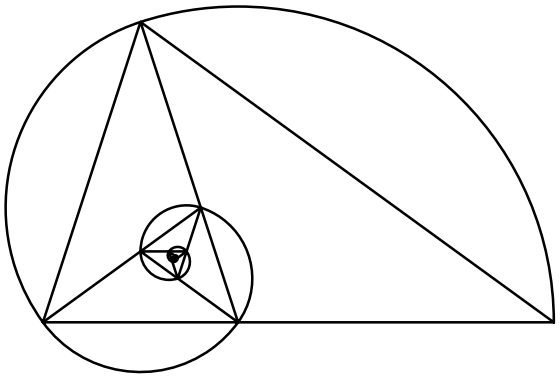
Variant growth spiral

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

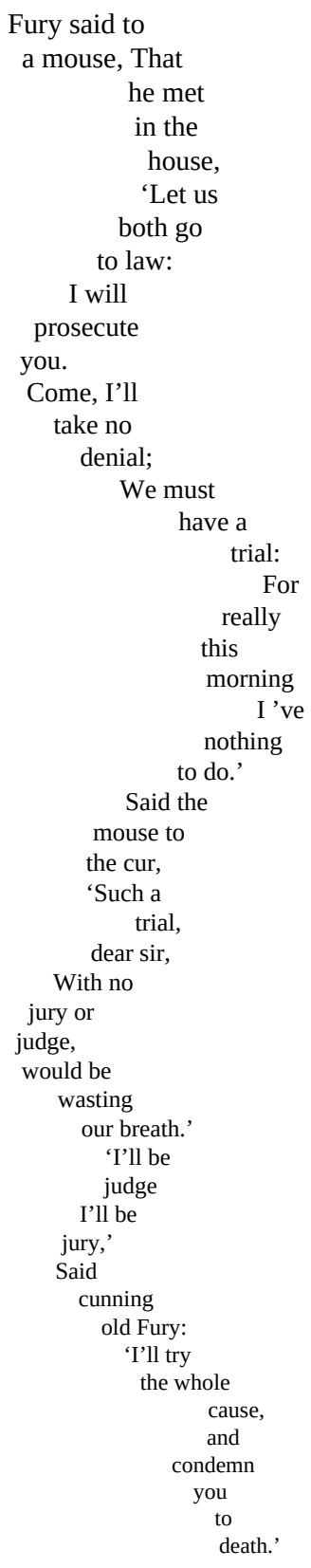


Variant growth spiral

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



The mouse's tail and Alice's tale



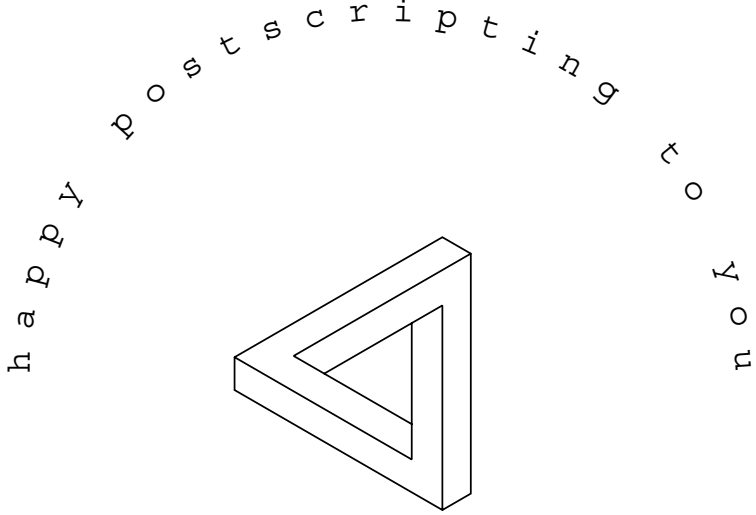
This emblematic proza by Lewis Carroll is typeset in PostScript by the use of `forall`, which expects an array, enclosed by [ ], and a procedure enclosed by { } on the stack. The array contains a necklage of strings, each enclosed by ( ), which holds the WYSIWYG data. The procedure scales and typesets the lines. No explicit positioning by coordinates on the page nor controlling of the loop is needed.

```
%!PS-Adobe EPSF-3.0
%%BoundingBox: 0 0 144 770
/crlf{ .995 dup scale currentpoint 12 sub exch pop 10 exch moveto } def
10 770 moveto /Times-Bold 12 selectfont
[ (Fury said to)
  (  a mouse, That)
  (                 he met)
  (                  in the )
  (                    house,)
  (                   'Let us)
  (                both go)
  (             to law:)
  (         I will)
  (    prosecute)
  (  you.)
  (   Come, I'll)
  (       take no)
  (           denial;)
  (                 We must)
  (                          have a)
  (                                 trial:)
  (                                      For)
  (                                   really)
  (                                this)
  (                                 morning)
  (                                         I 've)
  (                                 nothing)
  (                              to do.')
  (                    Said the)
  (              mouse to)
  (             the cur,)
  (             'Such a)
  (                     trial,)
  (              dear sir,)
  (        With no)
  (    jury or)
  (  judge,)
  (   would be)
  (         wasting)
  (             our breath.')
  (                 'I'll be)
  (                 judge)
  (             I'll be)
  (          jury,')
  (         Said)
  (             cunning)
  (                 old Fury:)
  (                     'I'll try)
  (                        the whole)
  (                                  cause,)
  (                                  and)
  (                              condemn)
  (                                  you)
  (                                      to)
  (                                       death.')]
{show crlf}
forall
showpage
%%EOF
```

Seal

A nice use of `kshow`.[22]

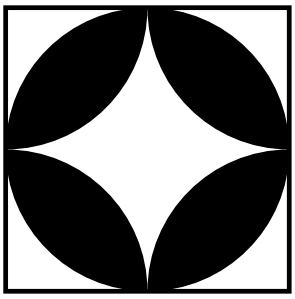
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Seal
%%BoundingBox: -110 -60 110 110
%%Beginsetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/sealdict 3 dict def
/seal{sealdict begin
/Courier 10 selectfont /r 100 def
/text (happy postscripting to you) def
gsave 89.9 rotate
0 r moveto
{-7.04 rotate 0 r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

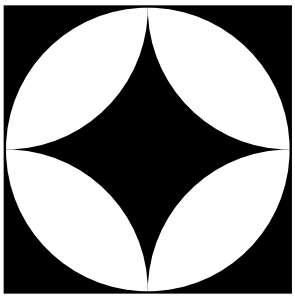


[22] What puzzles me is that rotation over 89.9 and 90 differ so much in appearance???

Reverse video

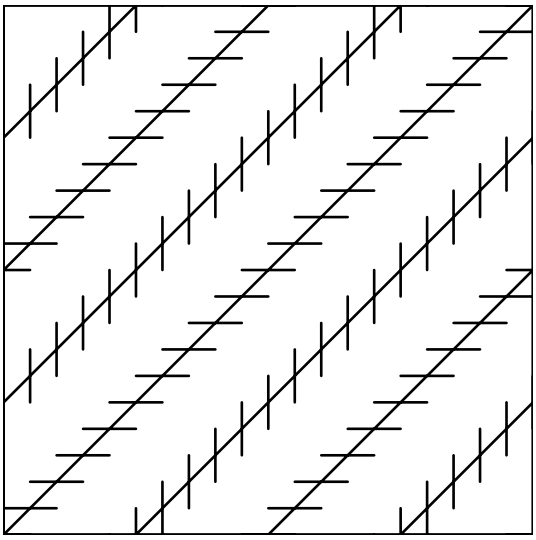
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Reverse video, cgl 96, 2009
%%BoundingBox: -32 -32 122 32
%%Beginsetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/reversevideodict 5 dict def
/reversevideo{reversevideodict begin
/r 30 def /2r r r add def /-r r neg def
/tile {4{r 0 moveto  0 0 r 0  90 arc
                    r r r 180 270 arc
            fill 90 rotate
        }repeat}def
/frame {-r -r 2r 2r rectstroke}def
tile frame
%reverse video tile
r 3 mul 0 translate frame
-r -r 2r 2r rectfill% black background
1 setgray tile stroke
end}def
reversevideo showpage
%%EOF
```





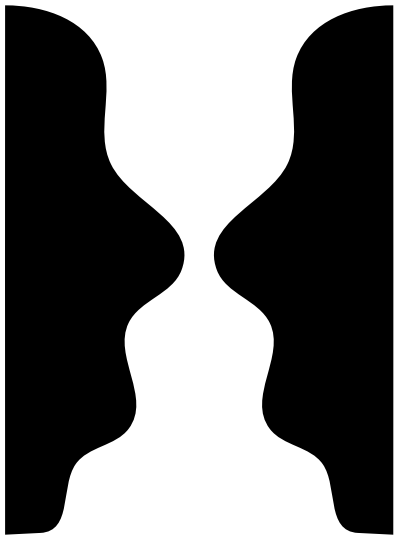
Optical Illusion

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Optical Illusion, cgl 1996
%%BoundingBox: -100 -100 100 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/opticalillusiondict 10 dict def
/opticalillusion{opticalillusiondict begin
/r 100 def      /mr r neg def
/hr .5 r mul def
/tr 2 r mul def /mtr tr neg def
/d r 10 div def
/line{mr mr moveto r r lineto}def
/ver{mr mr moveto
   mr d r{dup moveto
          0 d rmoveto
          0 d -2 mul rlineto}for
}def
/hor{mr mr moveto
   mr d r{dup moveto
          d 0 rmoveto
          d -2 mul 0 rlineto}for
}def
/frame{mr mr moveto tr 0 rlineto
       0 tr rlineto tr neg 0 rlineto
       closepath
}def
gsave frame clip
0 mtr translate
4{line hor 0 hr translate
  line ver 0 hr translate}repeat
stroke
grestore
frame stroke
end}def
%%EndProlog
opticalillusion
showpage
%%EOF
```



Candle or faces

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



Metaposttcode adapted from the old Metafont code.

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

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

Malevich's White cross on a white background

Malevich Suprematism:
White cross on a White background
Emulation →

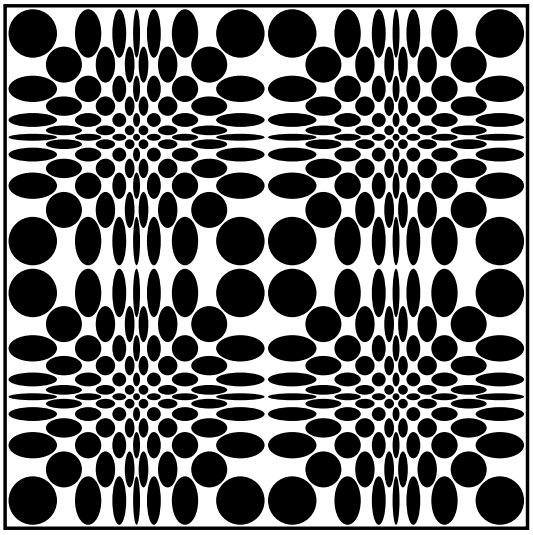
I have included a picture, and its emulation, of Malevich's[23] 'White Cross on a white background,' because he is the father of suprematism, which deletes the superfluous, which I associate with Minimal Mark-up. But ... sometimes redundancy is beneficial.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title:  Malevich White cross. cgl 2009
%%BoundingBox: -75 -100 75 100
%%BeginSetup
%%EndSetup
%%BeginProlog
/malevichdict 1 dict def
/malevich{malevichdict begin
/cross{-30 -100 moveto
      30 -100 lineto
      32 -20 lineto
      74 -20 lineto
      75  40 lineto
      33  40 lineto
      35  100 lineto
     -35  100 lineto
     -33  40 lineto
     -75  40 lineto
     -74 -20 lineto
     -32 -20 lineto
     -30 -100 lineto closepath
}def
gsave .91 setgray -75 -100 150 200 rectfill
      cross .95 setgray fill grestore
end}def
%%EndProlog
malevich
```

[23] Kazimir Malevich, 1878–1935. Russian painter, born in Kiev.

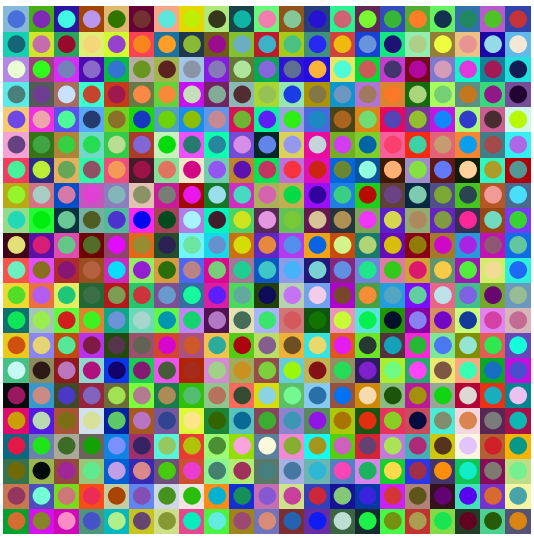
Schrofer's Op Art

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



Vasarely's Op Art: random coloured squares with circles

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Vasarely random coloured Squares. CGL 2007
%%BoundingBox: -211 -211 211 211
%%BeginSetup
%%EndSetup
%%BeginProlog
/vasarelyrandomdict 6 dict def
vasarelyrandomdict begin
/r 20 def              %side square
/R r 10 mul def        %bound loops
/r2 r 2 div def        %r/2
/ri r2 2 sqrt div def %radius inner circle
/s 2 31 exp 1 sub def %scaling random  numbers, reus
/square
{rand s div rand s div rand s div setrgbcolor
 r2 neg dup r dup rectfill

 rand s div rand s div rand s div setrgbcolor
 0 0 ri 0 360 arc fill} def
end%dict
/vasarelyrandom{vasarelyrandomdict begin 1951 srand
R neg r R{/i exch def
R neg r R{/j exch def
          gsave i j translate square grestore
         }for
         }for
end}def
%%EndProlog
vasarelyrandom showpage
%%EOF
```

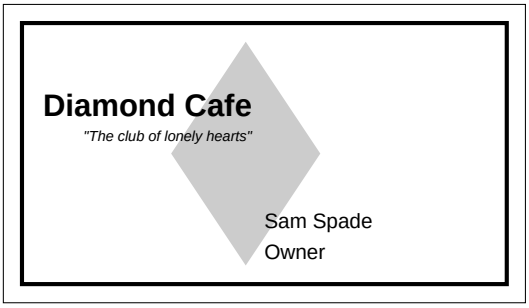


In Metafont the following Vasarely's I programmed 20 years ago. Interesting is the use of `interpath`, similar as used in the heart figure in the Metafont book p134. `interpath` is not available in PostScript, because we don't have explicit paths. Below I have imitated `interpath` functionality in PostScript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

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

Diamond Cafe

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Diamond cafe. Blue book. cgl 2014
%%BoundingBox: 89 89 343 235
%%BeginSetup
%%EndSetup
%%BeginProlog
%(c:\\PSlib\\PSlib.eps) run
/diamondcafedict 8 dict def
/diamondcafe{diamondcafedict begin
/MainFont /Helvetica-Bold findfont 15 scalefont def
/SloganFont /Helvetica-Oblique findfont 7 scalefont def
/OwnerFont  /Helvetica findfont 10 scalefont def
/rightshow%stack: string
{dup stringwidth pop
 120 exch sub 0 rmoveto show} def
/CardOutline{newpath
90 90 moveto 0 144 rlineto 252 0 rlineto 0 -144 rlineto closepath
.5 setlinewidth stroke} def
/doBorder{ 99 99 moveto 0 126 rlineto 234 0 rlineto 0 -126 rlineto closepath
2 setlinewidth stroke } def
/Diamond{
 newpath 207 216 moveto 36 -54 rlineto -36 -54 rlineto -36 54 rlineto
 closepath .8 setgray fill}def
/doTeXt{0 setgray
 90 180 moveto MainFont setfont (Diamond Cafe) rightshow
 90 168 moveto SloganFont setfont ("The club of lonely hearts") rightshow
 216 126 moveto OwnerFont setfont (Sam Spade) show
 216 111 moveto (Owner) show}def
%
CardOutline
doBorder
Diamond
doTeXt end}def
%%EndProlog
diamondcafe showpage
%%EOF
```

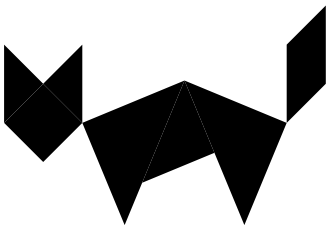


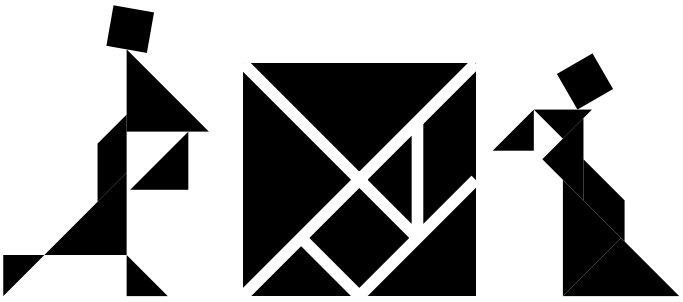
Omaha logo

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Omaha logo. Blue book p58. cgl 2014
%%BoundingBox: -1 -1 109 73
%%BeginSetup
%%EndSetup
%%BeginProlog
%(c:\\PSlib\\PSlib.eps) run
/omahadict 5 dict def
/omaha{omahadict begin
/Helvetica-Bold findfont 27 scalefont setfont
/fourpops{4{pop}repeat}def
/background{0 18 moveto 0 72 108 72 18 arcto fourpops
                      108 72 108 0 18 arcto fourpops
                      108 0 0 0 18 arcto fourpops
                      0 0 0 72 18 arcto fourpops
 fill}def
/moon{.6 setgray 81 45 18 0 360 arc fill}def
/omaha{1 setgray 0 -1 moveto 1 2 scale (Omaha) stringwidth pop
 108 exch sub 2 div 0 rmoveto (Omaha) show }def
background
moon
omaha
end}def
%%EndProlog
omaha showpage
%%EOF
```

Tangrams: little fox

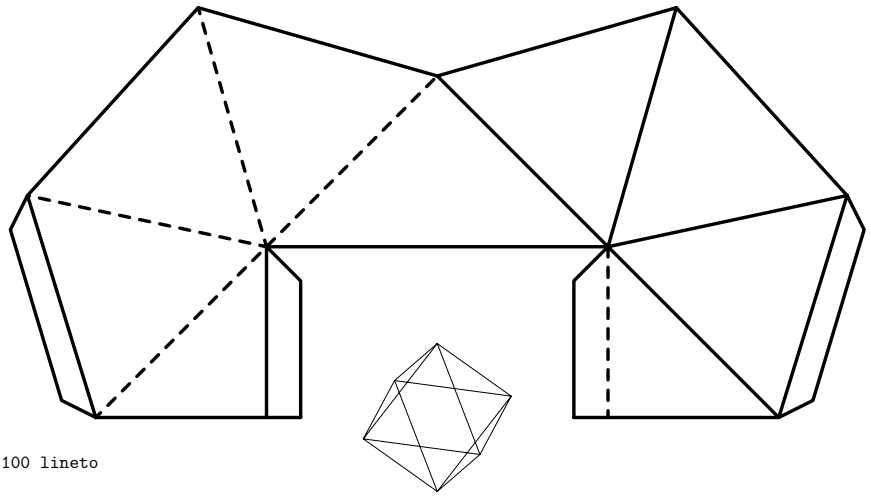
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Tangram Vosje. cgl 97
%%BoundingBox: -11 -16 32 18
%%BeginSetup
%%EndSetup
%%BeginProlog
%(c:\\PSlib\\PSlib.eps) run
/vosjedict 13 dict def
vosjedict begin
/s 20 def  2 setlinejoin
/reflectxy{-1 1 scale 90 rotate}def
/hs .5 s mul def /qs .5 hs mul def
/sds .707 s mul def
/hsds .5 sds mul def
/tri{/locs exch def
   0 locs moveto 0 0 lineto
   locs 0 lineto closepath fors}def
/trib{sds tri}def
/trim{hs tri}def
/tris{hsds tri}def
/sq{0 0 moveto hsds 0 lineto
   hsds hsds lineto 0 hsds lineto
   closepath fors}def
/dia{0 0 moveto hs 0 lineto
   3 qs mul qs lineto qs qs lineto
   closepath fors}def
/fors{stroke}def
end
%
/cat{vosjedict begin /fors{fill}def
gsave -67.5 rotate trib grestore%brest
gsave 135 rotate sq grestore%head
gsave
-.25 s mul .25 s mul translate -45 rotate
  tris 180 rotate tris
grestore%tail
gsave
22.5 cos sds mul dup add 0 translate
gsave reflectxy dia grestore
gsave 157.5 rotate
  trib
grestore
grestore
gsave 22.5 cos sds mul 22.5 sin sds mul translate
  -67.5 rotate hs 0 translate 180 rotate
  trim
grestore
end}def
%%EndProlog
cat
showpage
%%EOF
```





More complicated is the composed worshipped tangram, of which the code has been supplied in `PSlib.eps`. The square tangram is not complicated at all. It has always been fun to compose figures out of the pieces.

Tuckerman

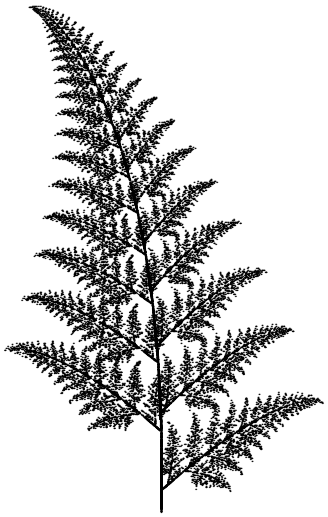


```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Cube-Oktaeder. cgl 2014
%%BoundingBox: -140 -25 140 130
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/Tuckerman{
/phi 60 def /theta 30 def
gsave .25 .25 scale
octaeder stroke
grestore
-40 0 moveto -100 0 lineto
0 100 moveto 100 0 lineto 40 0 lineto
-50 0 moveto -50 50 lineto
-50 50 moveto 50 50 lineto
-40 0 moveto -40 40 lineto -50 50 lineto
 40 0 moveto  40 40 lineto  50 50 lineto

100 0 moveto 120 65 lineto 70 120 lineto 0 100 lineto
50 50 moveto 120 65 lineto
50 50 moveto 70 120 lineto
100 0 moveto  110 5 lineto 125 55 lineto 120 65 lineto

-100 0 moveto -120 65 lineto -70 120 lineto 0 100 lineto
-100 0 moveto -110  5 lineto -125 55 lineto -120 65 lineto stroke
%dashed lines
 50 0 moveto  50 50 lineto
-100 0 moveto 0 100 lineto
-50 50 moveto -120 65 lineto
-50 50 moveto -70 120 lineto
[3]1 setdash stroke
stroke}def
Tuckerman showpage
%%EOF
```

Fern

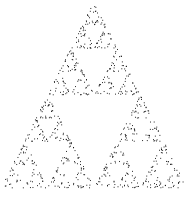
This picture shows what pointillism graphics can be obtained by the miracle of Iterated Function Systems. Barnsley(1988), Peitgen c.s.(2004).

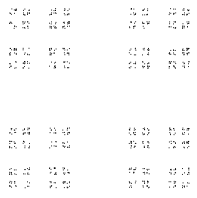
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Fern due to Barnsley., cgl May 97, June 2012
%%BoundingBox: -65 -1 70 208
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/ferndict 13 dict def
/fern%Author: Barnsley (1988): Fractals Everywhere. AP. (In PS by cgl)
{ferndict begin gsave
/Courier findfont 5 scalefont setfont /s 20 def
/reus 2 31 exp 1 sub def
/up 3 def %Array of mapping functions M [0..up]
/M[{  0
     .16 y mul}
   { .85 x mul -.04 y mul add
     .04 x mul  .85 y mul add 1.6 add}
   { .2  x mul -.26 y mul add
     .23 x mul  .22 y mul add 1.6 add}
   {-.15 x mul  .28 y mul add
     .26 x mul  .24 y mul add .44 add}
  ]def
/p[.04 .78 .89 1]def %accumulated probabilities
/thresholdk{reus p k get mul}def
/S{%loop maximum on stack
   {/r rand def %compact 'nested ifs'
    0 1 up{/k exch def
           r thresholdk lt {M k get exec
                            /y exch def /x exch def
                            exit}if
          }for %end nested ifs
   x s mul y s mul moveto (.) show
   }repeat
}def
/x 0 def /y 0 def 10 srand 32768 S stroke grestore end}def
%%EndProlog
fern showpage
%%EOF
```

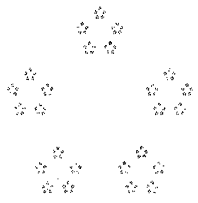


IFS in TEX

In 1989 I attended the TUG conference where Alan Hoenig showed some Iterated Function System fractals, a miracle,[24] which I reproduced in PWT. The idea is that the points within an n-gon are created by: the mean of a random point within the n-gon and one of its corners at random, à la Monte Carlo. A random number generator for plain TEX had to be written. The representation of the corner points is tricky via `\newdimen`-variables, in order to perform the arithmetic. Too many details in order to be presented here. The code for the three Sierpiński carpets below is included in `pic.dat`, which comes with `Blue.tex`







[24] His paper has been published in TUGboat 1989. For Iterated Function Systems and Fractals, see Peitgen c.s. (2004 2nd ed): Chaos and Fractals. Springer, which is more accessible than Barnsley, because . no sophisticated Math is required for reading the Peitgen book.

Escher's Buddhas

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escher Buddhas. cgl Jan 1997
%%BoundingBox: -10 -10 70 70
%%BeginSetup
%%endSetup
%%BeginProlog
/s 10 def /ms s neg def /ts s s add def /pa{s s}def
/pb{.7 s mul 1.7 s mul}def
/finger{.2 s mul 1.5 s mul}def
/pc{-.4 s mul 1.45 s mul}def
/pd{.5 s mul 1.35 s mul}def
/knee{.7 s mul 1.15 s mul}def
/pf{.35 s mul s}def
/ankle{-.1 s mul .6 s mul}def
/toe{0 .3 s mul}def
/heel{-.4 s mul .6 s mul}def
/head{-.9 s mul .9 s mul}def
/center{-.25 s mul 1.8 s mul}def
/cpa{0 1.55 s mul}def
/cpb{.25 s mul 1.55 s mul}def
/cpc{-.6 s mul 1.1 s mul}def
%
/element{gsave
pa moveto pb pb finger curveto
pc moveto cpa cpb pd curveto
knee knee pf curveto %pf lineto
ankle lineto toe lineto heel lineto
center .75 s mul 315 225 arcn
heel moveto cpc cpc head curveto
.5 setlinewidth stroke
grestore}def
%
/square{element
gsave  90 rotate 0 s -2 mul translate element grestore
gsave  1 -1 scale 90 rotate           element grestore
gsave -1 1 scale 0 s -2 mul translate element grestore
gsave ms s moveto s ms lineto
.05 setlinewidth stroke
grestore
}def
%
/tile{gsave square grestore
gsave ts 0  translate  90 rotate square grestore
gsave ts ts translate 180 rotate square grestore
gsave 0  ts translate -90 rotate square grestore
}def
%
/pattern{gsave tile
s 4   mul 0 translate tile
0 s 4 mul   translate tile
s -4  mul 0 translate tile
grestore}def
%
/as 8 s mul def /mas as neg def
/contour{ms ms moveto
 as 0 rlineto 0 as rlineto mas 0 rlineto
 closepath}def
%%EndProlog
%Draw
contour clip pattern
contour 1 setlinewidth stroke showpage
%%EOF
```

Gurari ABC

Nice suggestion of motion. Resembles much the ZIP picture of the Blue book p68.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Guarai's ABC
%%BoundingBox: -20 -52  100 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run

/gurariabcdict 1 dict def
/gurariabc{gurariabcdict begin
/Times-Bold 45 selectfont
-40 rotate
1 -.03 0{setgray
0 0 moveto  (ABC) show 3 rotate
} for
0 0 moveto -4 rotate
1 setgray (ABC) show
end}def
%%Endprolog
gurariabc
showpage
%%EOF
```

Escher‘s sun and moon

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escher zon en maan, cgl, Jan 97
%%BoundingBox: -107 -105 105 95
%%BeginSetup
%%EndSetup
/sunandmoondetail{% "path1"
0 0 moveto 2 12 4 24 6 36 curveto
-4.343 28.129 -17.003 23.909 -30 24 curveto
-36.121 24.043 -42.196 25.056 -48 27 curveto
-46 32 -44 37 -42 42 curveto
-34 48 -26 54 -18 60 curveto
-31.863 62.773 -46.137 62.773 -60 60 curveto
-56.193 67.780 -51.124 74.876 -45 81 curveto
-61.799 81 -75.082 68.916 -87 57 curveto
-81.148 47.163 -74.093 38.093 -66 30 curveto
-75.737 33.246 -86.263 33.246 -96 30 curveto
-99.302 20.093 -102.304 10.0886 -105 0 curveto
-99.283 9.036 -87.995 12.799 -78 9 curveto
-71.677 6.597 -66.694 1.22 -60 0 curveto
-45.345 -2.671 -35.296 13.902 -21 15 curveto
-11.291 15.746 -2.443 9.426 0 0 curveto
closepath .9 setgray fill % path2
0 0 moveto 8.24 -8.24 18.38 -15 30 -15 curveto
35.991 -2.187 48.856 6 63 6 curveto
58 -1 53 -8 48 -15 curveto
66.213 -11.435 83.553 -4.298 99 6 curveto
90.682 15.531 78.65 21 66 21 curveto
75.737 32.421 90 39 105 39 curveto
93.729 39 83.271 45 72 45 curveto
65.509 45 59.193 42.895 54 39 curveto
54 53.909 62.023 67.662 75 75 curveto
59.46 76.549 44.044 71.043 33 60 curveto
33 30.88 25.146 0 0 0 curveto
closepath .9 setgray fill %  path3
0 0 moveto -13.563 -3.391 -26.660 -8.428 -39 -15 curveto
-31 -15 -23 -15 -15 -15 curveto
-16.992 -22.845 -18 -30.907 -18 -39 curveto
-30.618 -35.024 -43.77 -33 -57 -33 curveto
-57 -50.395 -53.954 -67.656 -48 -84 curveto
-42 -71.187 -29.144 -63 -15 -63 curveto
-5.300 -74.848 0 -89.688 0 -105 curveto
8.089 -97.297 18.830 -93 30 -93  curveto
22 -81 14 -69  6 -57 curveto
21.008 -57 35.263 -63.578 45 -75 curveto
45 -58.475 37.22 -42.915 24 -33  curveto
14.93 -30.758 5.357 -31.821 -3 -36 curveto
-3 -36 -3 -36 -3 -36 curveto
-3 -23.930 -2.368 -11.84 0 0 curveto
closepath .9 setgray fill %  path4
-48 -84 moveto -54 -76 -60 -68 -66 -60 curveto
-74.093 -60 -82.156 -61 -90 -63 curveto
-85 -49 -80 -35 -75 -21 curveto
-85 -14 -95 -7 -105 0 curveto
-105 0 -95 -7 -105 0 curveto stroke %  path5
-45 81 moveto -42 84 -39 87 -36 90 curveto
-22 85 -8 80 6 75 curveto
10.940 76.278 15.948 77.28 21 78 curveto
39.022 80.570 57.3741 79.55 75 75 curveto
75 75 57.374 79.55 75 75 curveto stroke %  path6
99 6 moveto 99 -8.846 94.8431 -23.395 87 -36 curveto
84 -33 81 -30 78 -27 curveto
69 -29 60 -31 51 -33 curveto
51 -44.808 58.10 -55.459 69 -60 curveto
58.82 -60.102 49.553 -65.894 45 -75 curveto
45 -75 49.553 -65.894 45 -75 curveto
45 -75 49.553 -65.894 45 -75 curveto stroke
}def
sunandmoondetail showpage
%%EOF
```

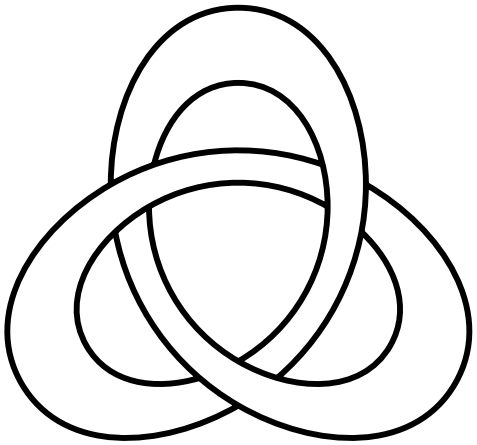

Zoals
mijn dochters
vroeger
analoog
puzzelden,
zo zullen
mijn kleinkinderen
in de toekomst
electronisch
nog plezier
aan Escher
beleven, en
ongetwijfeld
meer

Electronische puzzle

Escher doughnut

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

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

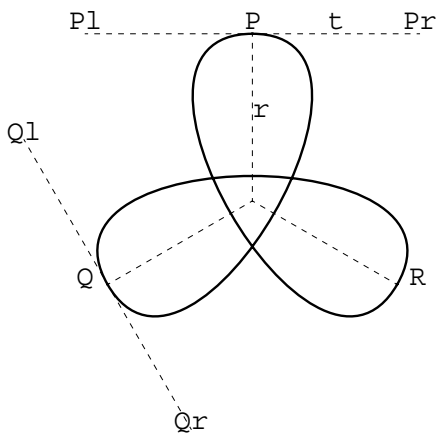


Postprocessed by Photoshop.

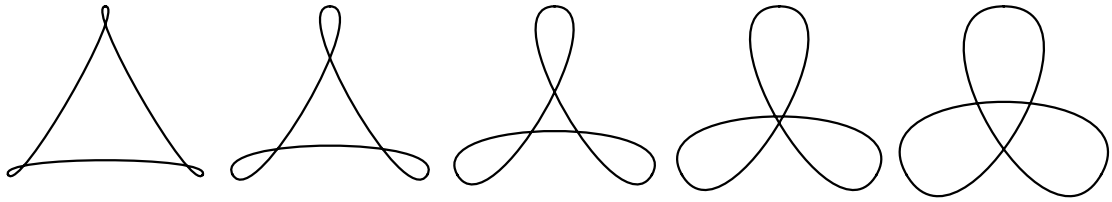
Escher‘s knot

The Escher knot was reprogrammed on the occasion of EuroTEX2012. An essential simpler problem that Escher doughnut. It shows that the tension concept is not needed, just the handles, ie a suitable choice of the control points of the spline is all that matters.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Escher knot
%%BoundingBox: -75 -55 75 85
%%BeginSetup
%%endSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
%%EndProlog
%used from the library: rot, centershow
/Courier 12 selectfont
/r 70 def /t 70 def
/P{0 r}def /Q{P 120 rot}def /R{P -120 rot}def
/Pr{t r}def /Pl{t neg r}def /Qr{Pl 120 rot}def
gsave .1 setlinewidth [2 3] 11 setdash
   3{0 0 moveto P lineto 120 rotate}repeat stroke grestore
P 2 add moveto (P) centershow
Q exch 13 sub exch moveto (Q) show
R exch 5 add exch moveto (R) show
3{P moveto Pr Qr Q curveto 120 rotate} repeat stroke
showpage
%%EOF
```

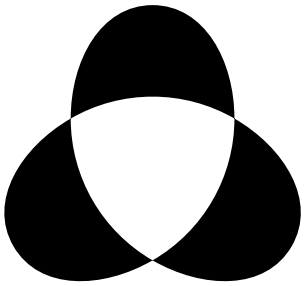


Variants



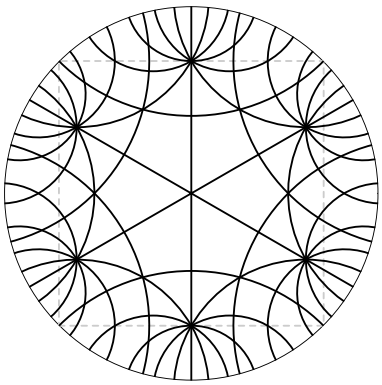
Escher knot eofill

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Escherknoteofill. cgl 1997
%%BoundingBox: -85 -70 85 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/escherknoteofill{%jalxxxix
-43.30 25 moveto
3{-43.30 55.49 -27.56  85     0    85 curveto
   27.56 85     43.30  55.49 43.30 25 curveto
   43.30 -5.94  26.79 -34.53  0   -50 curveto
 120 rotate
 }repeat closepath
eofill
}def
escherknoteofill showpage
%%EOF
```



Escher‘s Circle limit I grid

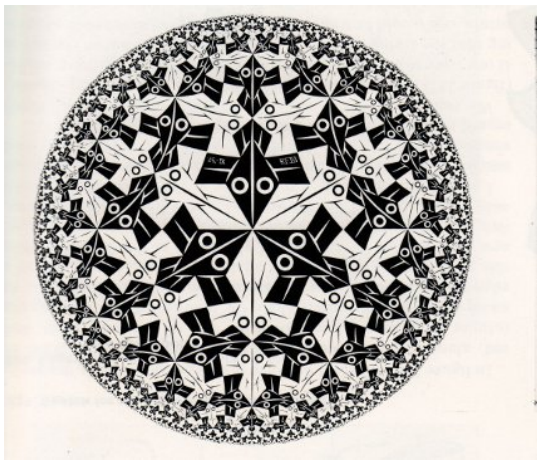
From the AMS in their feature column I borrowed how to draw a la Escher the grid for Limit Circle I. All arcs cut the boundary circle orthogonally. The next step is to create contours and color.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: CirclelimitI. cgl 2014
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
newpath 0 0 100 0 360 arc%central circle
clip
stroke
/Courier 14 selectfont
gsave %.7 setgray
/h1x 0 def /h1y -200 sqrt2 div def h1x 5 sub h1y 20 sub moveto (h1) show
0 h1y moveto 6{60 rotate 0 h1y lineto}repeat%circumscribed hexagon
stroke
gsave -100 sqrt2 div dup moveto 100 sqrt2 div dup neg lineto
 100 sqrt2 div dup lineto -100 sqrt2 div dup neg lineto closepath%square
[3]0 setdash .8 setgray stroke grestore
%[]0 setdash%solid lines
6{0 h1y 100 0 360 arc stroke 60 rotate}repeat%circle1
0 h1y  0 h1y  60 rot mean /y exch def /x exch def
6{x y 70.7 0 360 arc stroke 60 rotate}repeat%Circle2
stroke
%next hexagon
/h2x x def /h2y y def h2x h2y 20 sub moveto (h2) show
6{h2x h2y moveto x y 60 rot lineto 60 rotate}repeat%hexagon
stroke
6{h2x h2y h2x h2y -60 rot mean /h3y exch def /h3x exch def
/r3 h3y abs 100 sqrt2 div sub def
h3x h3y r3 0 360 arc stroke 60 rotate}repeat
% points on one third
6{/m4x h2x 3 div def
%m4x h2y moveto (.) centershow
m4x h2y m4x 2 mul 0 360 arc stroke
m4x neg h2y m4x 2 mul 0 360 arc stroke 60 rotate}repeat
3{0 100 moveto 0 -100 lineto 60 rotate}repeat%line bundel
stroke
grestore
showpage
%%EOF
```

Coxeter cooperated with Escher.

The figure left from Coxeter stimulated Escher to develop his limit circle series. At reight Eschers limcircleI.

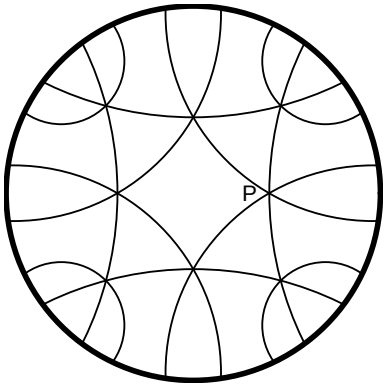


Escher‘s Circle limit III grid

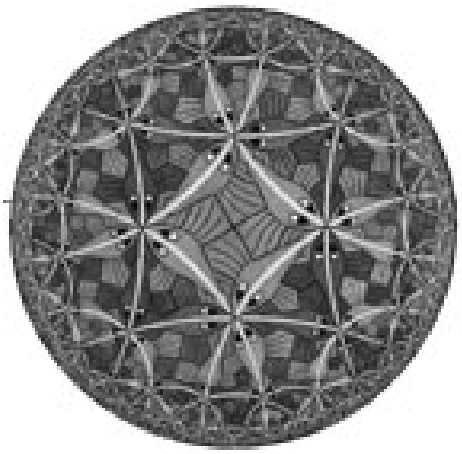
Coxeter’s solution

Coxeter, in *What Escher left unstated*, The Mathematical Intelligences, 18, 4, 1996, analyzed Escher’s Circle Limit III. He started from the rotational symmetry at P and derived (the parameters for) the grid.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Circle cuts scaled unit circle. at 80 degrees. CGL 2010
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run %PS library
/circlelimdict 8 dict def
/circlelim{circlelimdict begin
/r 100 def %equals r1. 1st circle: 0 0 r
/p 14 34 div r mul def  /P {p 0} def
gsave p 15 sub -4 moveto (P) H12pt setfont show grestore
gsave
0 0 r 0 360 arc %unit circle scaled r
gsave 3 setlinewidth stroke grestore
clip
%main arcs
/r1 r 1.1081646 mul def
/d1 r 1.3572189 mul def
4{newpath d1 sqrt2 div dup r1 0 360 arc stroke
   90 rotate}repeat
%parallel arc
/o3 .998189 r mul sqrt2  div def
/r3 .3375915 r mul def
4{newpath o3 o3 r3 0 360 arc  stroke
   90 rotate}repeat
/d2 2.2104 r mul def
/r2 1.8047860 r mul def
4{newpath d2 0 r2 0 360 arc stroke
   90 rotate}repeat
end}def
circlelim showpage
%%EOF
```



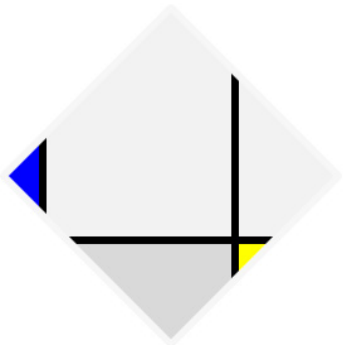
At right Eschers Limit Circle III



Invitation

In the 90-ties I worked in Metafont, and reused the code for processing with Metapost much later. Mondriaan missed the near golden ratio proportions. The pictures included adhere to the golden ratio proportions. Later I programmed `intersection` for the calculation of the intersection points of 2 straight lines directly in PostScript, no need to use Metapost. The PostScript and Metapost programs are equal modulo some syntactic sugar. In rewriting the Metapost code into PostScript I realized that the calculation of the intersection point is superfluous.

```
%Mondriaan-like Lozenge with 3 lines. Fall 2009 kisa1@xs4all.nl
beginfig(0)
s=200;
z1=(0,.5s); z2=(.5s,s); z3=(s,.5s); z4=(.5s,0);
z5=0.618/1.618[z2,z3]; z6=0.618/1.618[z3,z4];
z7= 1/1.618[z3,z4]; z8=1/1.618[z4,z1];
z9=0.382/1.618[z1,z4]; z10=0.382/1.618[z1,z2];
z11= (z5--z7) intersectionpoint (z6--z8);
path p; p = z1--z2--z3--z4--cycle;
pw=4; pickup pencircle scaled pw;
fill p withcolor .95white;
fill z1--z10--z9--cycle withcolor blue;
fill z7--z11--z6--cycle withcolor red+green;
fill z8--z11--z7--z4--cycle withcolor .85white;
draw z5--z7;
draw z6--z8;
draw z9--z10;
clip currentpicture  to p;
draw p withcolor .97white;
endfig
end
```



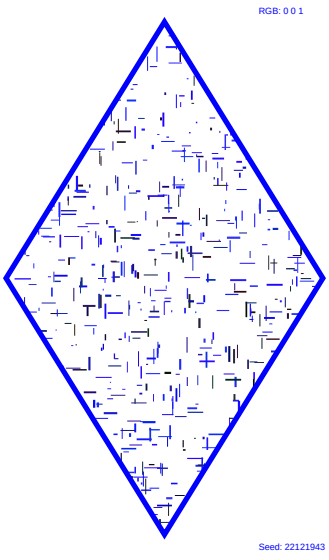
Made into an invitation by Photoshop and the recent developed direct PostScript

Note the use of the library `def intersect`.

```
%PostScript-Adobe-3.0 EPSF-3.0
%%Title: Lozenge 3 lines. Mondriaan
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
(c:\\PSlib\\PSlib.eps) run
/lozenge3dict 10 dict def

lozenge3dict begin%locals
/s 200 def
/p1{.5 s mul 0}def /p2{0 .5 s mul} def /p3{-.5 s mul 0} def /p4{0 -.5 s mul} def
/p8 {.5 s mul .618 1.618 div mul .5 s mul 1.618 div}def
/p7{.5 s mul .618 1.618 div mul  -.5 s mul 1.618 div}def
/p5{ -.5 s mul 1.618 div -.5 s mul .618 1.618 div mul}def
/p6{ .5 s mul 1.618 div -.5 s mul .618 1.618 div mul}def
/p9{ -.4 s mul -.1 s mul}def
/p10{ -.4 s mul .1 s mul}def
end
/lozenge3{lozenge3dict begin gsave
%background
p1 moveto p2 lineto p3 lineto p4 lineto closepath .95 setgray fill
%blue
p3 moveto p10 lineto p9 lineto closepath 0 0 1 setrgbcolor fill
%yellow
p7 moveto p6 lineto
   .5 s mul .618 1.618 div mu 1.618 div mul -.5 s mul .618 1.618 div mu lineto %intersection
   closepath 1 1 .5 setrgbcolor fill
p1 moveto p2 lineto p3 lineto p4 lineto closepath
%lines
2 setlinewidth
p7 moveto p8 lineto p5 moveto p6 lineto p9 moveto p10 lineto 0 setgray stroke
%frame
p1 moveto p2 lineto p3 lineto p4 lineto closepath 3 setlinewidth .95 setgray stroke
grestore end}def
lozenge3 showpage
%%EOF
```

Mondriaan's line pieces.

Since *à la Mondriaan* the PostScript program has been extended by a square and circle as frame. In the paper the Metapost program as well as the PostScript program have been developed and discussed. Because of the seed the pictures are unique.




```
%PostScript-Adobe-3.0 EPSF-3.0
(c:\\PSlib\\PSlib.eps) run
/mondrian% 2014 extended by circle and square
%birthday: ddmmyyyy, a number as seed for srand
%three numbers from the closed interval [0, 1], for the rgb-color values: red green blue
%number for the kind of frame (0=rectanglere 1=Oval 2=Lozenge 3=square 4=circle): 0, 1 or 2
%==> generated Mondrian-alike
{0 begin gsave %savety for not changing the graphics state outside
  % used from the library: unifrmdef, maxrandom, ellipse
/form exch def
/b exch def /g exch def /r exch def /date exch def
date srand% start random generator with (birthday date) seed
100 50 translate
%wired-in parameters
/u 420 def /v u 1.618 mul def /hu u 2 div def /hv v 2 div def%BB of rectangle: 0 0 u v
/maxrandom 500 def /maxlength 20 def /maxwidth 3 def /eps 0.1 def
/hx  {u unifrmdev}            def
/hy  {v unifrmdev}           def
/l    {maxlength unifrmdev}def
/w  {maxwidth unifrmdev} def
/spread {2 unifrmdev mul }def                 %(0, 2)
%/spread {2 unifrmdev 1 add 2 div mul }def%(0.5, 1.5)
%/spread {2 unifrmdev 3 add 4 div mul }def%(0.75, 1.25)
/color{r 0 eq {eps}{r} ifelse spread
          g 0 eq {eps}{g} ifelse spread
          b 0 eq {eps}{b} ifelse spread} def
form 0 eq {/contour {0 0 moveto u 0 lineto u v lineto 0 v lineto closepath} def} if      %rectangle
form 1 eq {/contour {hu hv hu hv 0 360 ellipse} def} if    %oval
form 2 eq {/contour {hu 0 moveto u hv lineto hu v lineto 0 hv lineto closepath} def} if%lozenge
form 3 eq {/contour {0 0 moveto u 0 lineto u u lineto 0 u lineto closepath} def} if      %square
form 4 eq {/contour {hu hv hu hu 0 360 ellipse} def} if    %circle
 %oval
gsave contour clip%random pattern will only show up in (is clipped to) contour
maxrandom{%draw pattern in loop confined to contour
/xaux hx def /yaux hy def%position in (0, u) x (0, v) rectangle
/laux l 2 div def
xaux laux sub yaux moveto  xaux laux add yaux         lineto w setlinewidth color setrgbcolor stroke%h-
line
/xaux hx def /yaux hy def
/laux l 2 div def
xaux yaux laux sub moveto  xaux         yaux laux add lineto w setlinewidth color setrgbcolor stroke%v-
line
}repeat
grestore %end clipping path
contour 7 setlinewidth r g b setrgbcolor stroke%original color od choice
H12pt setfont  /nstr 8 string def %0 0 0 setrgbcolor
u 85 sub v 10 add moveto (RGB: ) show
                    r nstr cvs show ( ) show
                    g nstr cvs show ( ) show
                    b nstr cvs show
u 85 sub -20 moveto (Seed: ) show  date nstr cvs show
grestore end}def%end Mondrian
/mondrian load 0 26 dict put
%example of use
22121943 0 0 1 2 mondrian showpage
%%EOF
```

Smiley

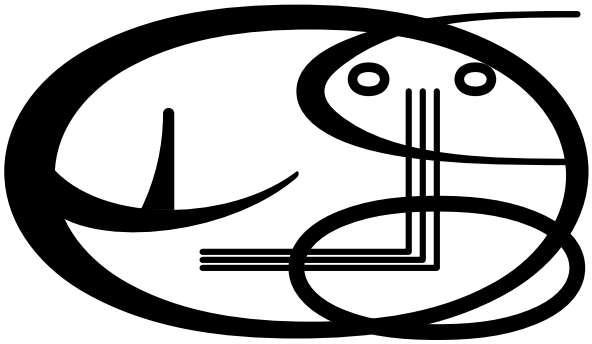
Trivial, straightforward PostScript programming.

```
%PostScript-Adobe-3.0 EPSF-3.0
%%Title: Smiley
%%BoundingBox: 150 250 455 545
%%BeginSetup
%%EndSetup
%Draw a two inch blue circle
/smiley{0 0 1 setrgbcolor
306 396 144 0 360 arc closepath fill
%Draw two half inch yellow circles for eyes
1 1 0 setrgbcolor
252 432 36 0 360 arc closepath fill
360 432 36 0 360 arc closepath fill
%Draw the smile
1 setlinecap 18 setlinewidth
306 396 99 200 340 arc stroke
}def
smiley showpage
%%EOF
```

Cat

My first exercise in graphics in Metafont. Converted into PostScript by Metapost.

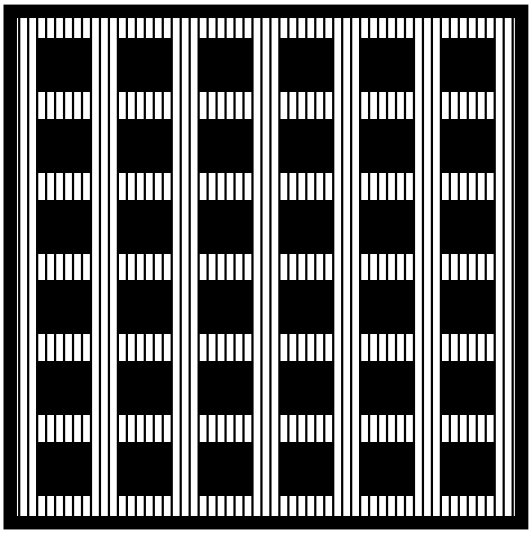
```
%!PS-Adobe-3.0 EPSF-3.0
%%Creator: MetaPost
%%CreationDate: 1996.05.03:1858
%%BoundingBox: -4 -3 179 103
%%BeginSetup
%%EndSetup
newpath 178.50075 50 moveto
%Too lengthy,  and boring PS from Metapost.
178.50075 66.10077 167.60909 79.62463 153.4761 87.70062 curveto
133.5853 99.0668 110.38673 102.00043 87.5 102.00043 curveto
... etc
%%EOF
```



Cut into coloured pieces for a puzzle.

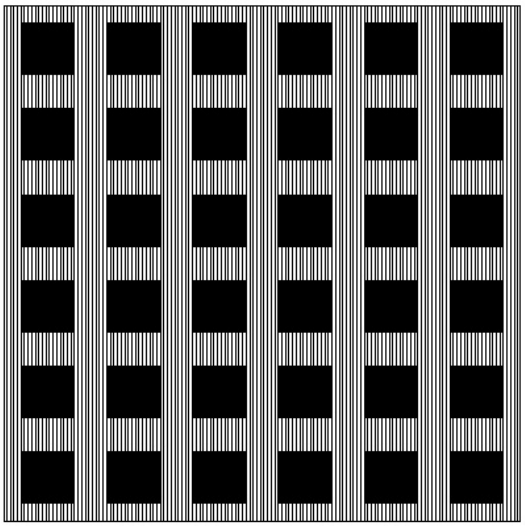
Soto's Op Art

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



Earlier in TEX alone

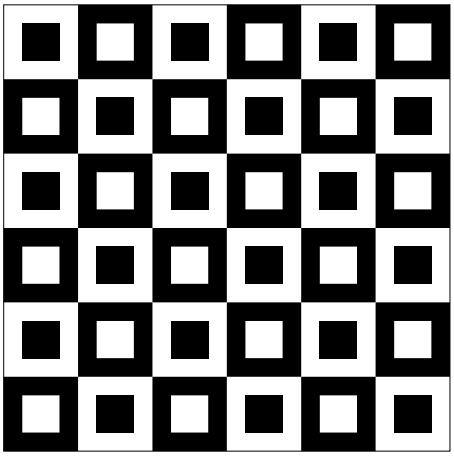
```
%cgl feb 2010 tst Soto
\def\boxit#1{\vbox{\hrule\hbox{\vrule#1\vrule}\hrule}}
\newbox\cb  \newdimen\ul \ul=6ex \newdimen\size \size12\ul
\setbox\cb\vbox to2\ul{\vss%colored box with transparent boundary
   \hbox to2\ul{\hss\vrule height1.2\ul width1.2\ul \hss}%
        \vss}%

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



Jiggling squares

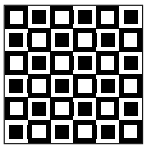
A nice optical effect. The programming consists of just putting boxes nect and on top of each other. Originally programmed in TEX alone and of EuroTEX2012 also in PostScript.

```
%!PS-Adobe-3.0  EPSF-3.0
%PostScript-Adobe-3.0 EPSF-3.0
%%BoundingBox: -1 -1 121 121
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/jigglingsquaresdict 2 dict def
jigglingsquaresdict
begin
/Oc{0 0 moveto 20 0 rlineto 0 20 rlineto -20 0 rlineto closepath}def%outer contour
/Ic{5 5 moveto 0 10 rlineto 10 0 rlineto 0 -10 rlineto closepath}def%inner contour
end
/jigglingsquares{jigglingsquaresdict begin
gsave .1 setlinewidth 0 0 120 120 rectstroke grestore
3{gsave 3{Oc Ic fill 20 0 translate
          Ic    fill 20 0 translate}repeat
 grestore
 gsave 0 20 translate
        3{Ic fill    20 0 translate
          Oc Ic fill 20 0 translate}repeat
 grestore 0 40 translate}repeat
end}def
%%Endprolog
jigglingsquares
showpage
%%EOF
```



Earlier in TEX alone

```
\def\boxit#1{\vbox{\hrule\hbox{\vrule#1\vrule}\hrule}}
\newbox\soto \newbox\sot
\setbox\soto\vbox to 2ex{%
   \hrule height.4ex width 2ex depth0pt
   \hbox to 2ex{%
    \vrule height1.2ex width.4ex depth0pt
    \hss
    \vrule height1.2ex width.4ex}%
   \hrule height.4ex width2ex depth 0pt}%
\setbox\sot\vbox to2ex{\vss
   \hbox to2ex{\hss\vrule height1.2ex width1.2ex depth0pt\hss}%
        \vss}%
\boxit{\vbox{\leaders\vbox{\offinterlineskip
   \hbox{\leaders\hbox{\copy\soto\copy\sot}\hskip12ex}%
   \hbox{\leaders\hbox{\copy\sot\copy\soto}\hskip12ex}}%
   \vskip12ex}}
\bye
```



Waves clipped by a circle

Hobby provides the picture in his Metapost manual. It shows how to use the clip operator and how to prevent the clipping path to be become part of the picture. The waves are thick lines. When contours are used the colouring is confusing.

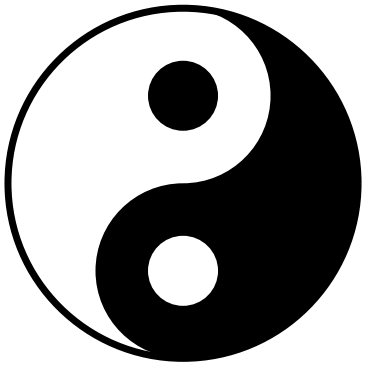
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Waves clipped by circle. cgl 2014
%%BoundingBox: -11 -11 11 11
%%BeginSetup
%%EndSetup
(c:\\\PSlib\\PSlib.eps) run
/wave{-10 1 moveto -8 2 -8 2 -6 1 curveto
             -4 0 -4 0 -2 1 curveto
              0 2  0 2  2 1 curveto
              4 0  4 0  6 1 curveto
              8 2  8 2  10 1 curveto
blue stroke} def
0 0 10 0 360 arc clip 1 setgray stroke%white clipping circle
-12 3 12{gsave 1.5 setlinewidth 0 exch translate wave grestore}for  showpage
%%EOF
```

Jackowski's variant program, which he explained on the NTG Metafont course in the 90-ies, reads

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Waves clipped by circle. Jackowski, 1994
%%BoundingBox: -50 -50 50 50
%%BeginSetup
%%EndSetup
/tile{-2 0 moveto
       25 10 25 10 50 0 curveto
       75 -10 75 -10 102 0 curveto}def
/frame{50 0 moveto 0 0 50 0 360 arc}def
/dotiling{-50 -50 translate
 .9 -.1 .1 {setgray tile stroke 0 15 translate}for}def
gsave frame clip 1 setgray stroke
10 setlinewidth dotiling
grestore showpage
%%EOF
```

Yin Yang

A time-proven picture.



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Yin Yang. cgl July 2009
%%BoundingBox: -26 -26 26 26
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/yinyangdict 7 dict def
yinyangdict begin
/R 25 def     /hR R 2 div def
/mR R neg def /mhR hR neg def
/r R 5 div def /mr r neg def
/circle {translate % center on stack
         r 0 moveto 0 0 r 0 360 arc
}def
end

/yinyang{yinyangdict begin
0 mR moveto 0   0  R    270  90 arc
            0   hR  hR   90 270 arcn
            0   mhR hR   90 270 arc  fill
R 0 moveto 0 0 R 0 360 arc stroke
gsave 0 hR circle fill grestore
gsave 0 mhR circle 1 setgray fill grestore
end}def
%%EndProlog
yinyang
showpage
%%EOF
```

In Metapost Hobby's program reads

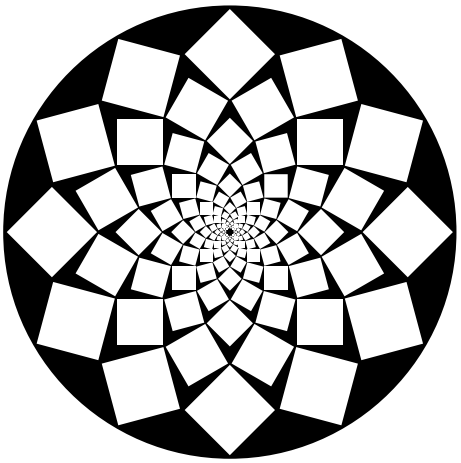
Borrowed for Hobby's Metapost manual. Concise, although the picture is not complete.

```
beginfig{21}; path p;
p=(-1cm, 0)..(0,-1cm)..(1cm,0);
fill p{up}..(0,0){-1.-2}..up}cycle;
draw p..(0,1cm)..cycle;
endfig;
```

Dwirling squares

The basic code has been borrowed from Lauwerier and translated into PostScript.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Dwirling squares, cgl 1997
%%BoundingBox: -255 -255 255 255
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/dwirlingsquaresdict 11 dict def
dwirlingsquaresdict begin
/r 200 def /hs r 15 sin mul def /s 2 hs mul def
/rotsq{hs 1 sub 0 moveto
4{90 rotate hs 1 sub 0 lineto}repeat closepath fill
}def
/ring{12{gsave r 0 translate rotsq
        grestore 30 rotate }repeat
}def
/rs r hs add 3 add def /mrs rs neg def
/frame{rs mrs moveto rs rs lineto
   mrs rs lineto mrs mrs lineto
   closepath
}def
/frame{rs 0 moveto 0 0 rs 0 360 arc closepath}def
/f r hs sub r div 15 cos mul def
end
/dwirlingsquares{dwirlingsquaresdict begin
frame 0 setgray fill
1 setgray
12{ring f f scale 15 rotate
 }repeat
end}def
%%EndProlog
dwirlingsquares
showpage
%%EOF
```

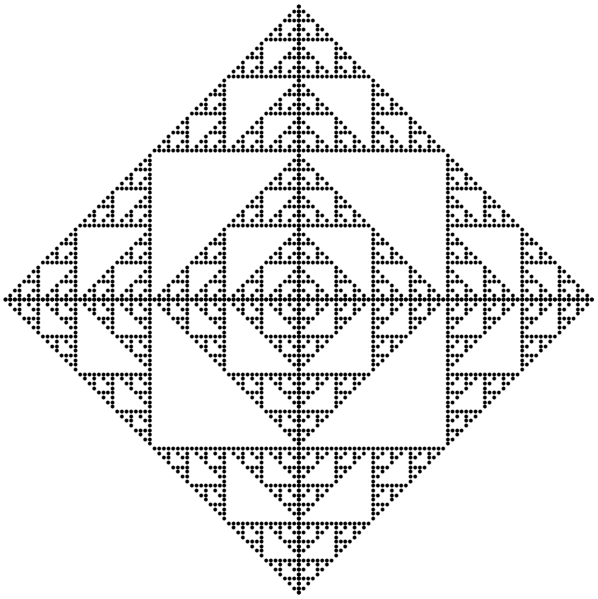


Game of Life

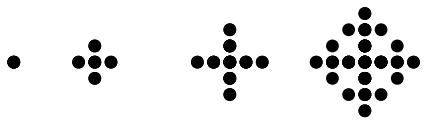
Lauwerier(1990) mentions a fractal, which he obtained from the Pickover variant of the Game of Life, made popular by Martin Gardner in a Scientific American of 1970. The game is played on a grid. Each node can be alive or dead. Once alive it stays alive. If dead it comes to life if only one neighbour,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

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

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth Cell model a la Pickover, simplified. cgl 2012
%%BoundingBox: -195 -195 195 195
%%BeginSetup
%%EndSetup
%%BeginPrologue
/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
64 pickover showpage
%%EOF
```



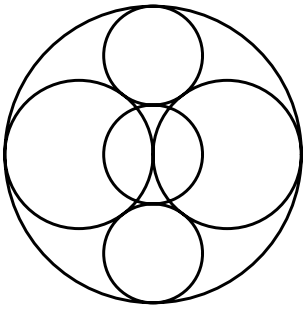
The first 4 generations



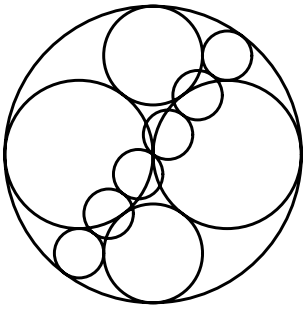
Sangaku

In Japan sangaku pictures are used to illustrate theorems.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Sangaku Circles. CGL March 2010
%%BoundingBox: -51 -51 51 51
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/sangaku1dict 3 dict def
/sangaku1{sangaku1dict begin
/R 50 def        %center  (0,0)
/r1 R 2 div def /x1 r1 def  /y1 0 def
/r2 R 3 div def %center  (0,0)
0 0 R 0 360 arc  stroke
x1 y1 r1 0 360 arc  stroke
x1 neg y1 r1 0 360 arc  stroke
0 0 r2 0 360 arc  stroke
0 r2 2 mul r2 0 360 arc stroke
0 r2 -2 mul r2 0 360 arc stroke
end} def
%%Endprolog
sangaku1 showpage
%%EOF
```

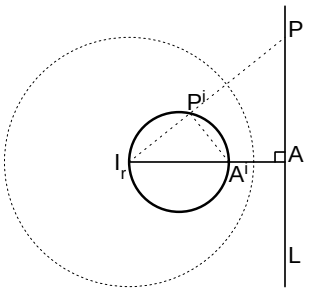


```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Sangaku Circles. CGL March 2010
%%BoundingBox: -51 -51 51 51
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
/sangaku2dict 5 dict def
/sangaku2{sangaku2dict begin
/sqrt13 13 sqrt def
/R 50 def        %center  (0,0)
/r1 R 2 div def /x1 r1 def  /y1 0 def
/r2 R 3 div def %center  (0,0)
/r3 R 6 div def %center  (0,0)
0 0 R 0 360 arc  stroke
x1 y1 r1 0 360 arc  stroke
x1 neg y1 r1 0 360 arc  stroke
0 r2 2 mul r2 0 360 arc  stroke
0 r2 -2 mul r2 0 360 arc  stroke
0 0 R neg %circumscribed
0 r2 2 mul r2
x1 y1 r1
Apollonius2 /r3a exch def /y3a exch def /x3a exch def
        /r3b exch def /y3b exch def /x3b exch def
x3a y3a r3a 0 360 arc stroke
%x3b y3b r3b 0 360 arc stroke%circle x1 neg y1 r1 0 360 arc
 x3a 5 div y3a 5 div r3a 0 360 arc stroke
 x3a -5 div y3a -5 div r3a 0 360 arc stroke
 x3a 5 div 3 mul y3a 5 div 3 mul r3a 0 360 arc stroke
 x3a neg y3a neg  r3a 0 360 arc stroke
 x3a -5 div 3 mul y3a -5 div 3 mul r3a 0 360 arc stroke
end} def
%%EndProlog
sangaku2 showpage
%%EOF
```



Line inversion

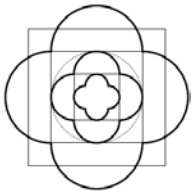
The inversion of a line in a circle is a circle through the origin of the inversion circle. The picture is a nice illustration of the theorem. The use of angle marks contributes much to the clarity of the picture.

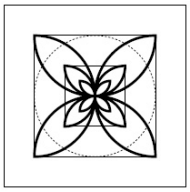



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: line inversion. CGL nov2009
%%BoundingBox: -76 -81 106 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %PS library
/lineinversiondict 10 dict def
/lineinversion{lineinversiondict begin
/r 75 def /2r r r add def H14pt setfont
gsave  [3] 0 setdash
0 0 r 0 360 arc stroke                    %inversion circle
grestore
%Marking origin of inversion circle
gsave
-9 -5 moveto (I) show 0 -5 rmoveto  H10pt setfont (r) show
grestore
%
%To be inverted line L
/L1 { 1.25 r mul 1.25 r mul} bind def
/L2 {1.25 r mul r neg} bind def
L1 moveto L2 lineto stroke
L2 moveto 2 14 rmoveto (L) show
%Point A where perpendicular meets L
0 0 moveto 1.25 r mul 0 lineto currentpoint stroke
     moveto 2 0 rmoveto (A) show
0 0 1.25 r mul 0 6 ortho
%Arbitrary point P on L
/Px 1.25 r mul def /Py r def /P {Px Py} def
gsave auxlin 0 0 moveto P lineto currentpoint stroke grestore
     moveto 2 0 rmoveto (P) show
P 0 0 r pointinversion /Piy exch def /Pix exch def /Pi {Pix Piy} def
%Inversion point Ai auxlin
/Aix r 1.25 div def /Ai {Aix 0} def
Ai moveto Pi lineto stroke
Pi moveto -2  2 rmoveto H14pt setfont (P) show
gsave 0 5 rmoveto (i) H10pt setfont show grestore
Ai moveto 0 -12 rmoveto H14pt setfont (A) show
gsave 0 5 rmoveto H10pt setfont  (i) show grestore
newpath
Aix 2 div 0 Aix 2 div 0 360 arc wanlin stroke
end}def
%%EndProlog
lineinversion showpage
%%EOF
```

Inversion of straight line pieces

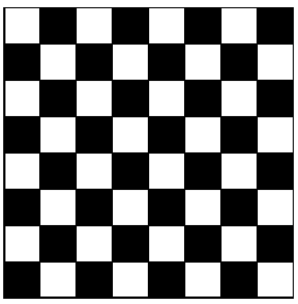
Three squares with sides r, 2r , 4r centred around the origin are inverted in the (dashed) circle $I_{r,(0,0)}$. The inversions are drawn in the same picture non-dashed and bold. The right figure is the complement: the inversions of the lines with the sides of the squares left out.

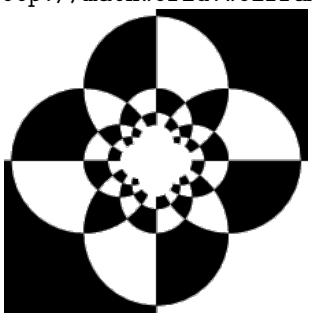




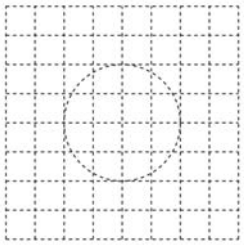
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: line piece inversion. CGL nov2009
%%BoundingBox: -76 -81 106 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %PS library
/linepieceinversiondict 20 dict def
/linepieceinversion
% x1 y1 x2 y2: the points which determine the line
% mx my r:     the centre and radius of the inversion circle
%==>
%              path of inverted cirle arc
{linepieceinversiondict  begin
/r exch def /my exch def /mx exch def
/y2 exch def /x2 exch def /y1 exch def /x1 exch def
x1 y1 x2 y2 mx my r lineinversion  /ri exch def /miy exch def /mix exch def
x1 y1 mx my r pointinversion /y1 exch def /x1 exch def
x2 y2 mx my r pointinversion /y2 exch def /x2 exch def
/phi1 y1 miy sub  x1 mix sub atan def
/phi2 y2 miy sub x2 mix  sub atan def
%(x1, y1)--(0, 0) right from (xm, ym)--(0, 0)?
/psi1 y1 x1 atan def
/psim miy mix atan def
psi1 180 gt {/psi1 psi1 360 sub def} if
                           %correction if xm in 1st and x1 in 4th quadrant
psi1 psim lt
{x1 y1 moveto mix miy ri phi1 phi2 arc}
{x2 y2 moveto mix miy ri phi2 phi1 arc} ifelse
end} def
%%EOF
```

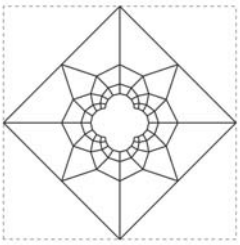
A beautiful illustration of the inversion of line pieces is the inversion of a chessboard centred at (0, 0) in a small circle also centred at (0, 0) (M. Gardner 1984, pp. 244-245; R. Dixon 1991). In the picture below the chessboard and its inversion are shown, borrowed from `http://mathworld.wolfram.com/Inversion.html`.





Circle inversion of a rectangular grid





The inversion has been simplified: circular curves have been straightened in the inversion.

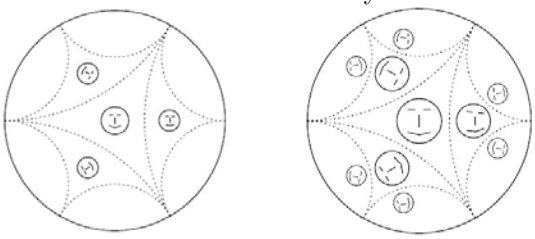
At right the realization as a skylight window.

The first picture from the left shows the grid and the inversion circle. The second picture shows the inversion of the grid. At right the resulting stainless glass skylight window, size 60x60cm, to decorate our apartment in Chisinau.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Chessboard inversion. 2010,  CGL
%%BoundingBox: -130 -130  380 130
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run
%
/chessboardinvdict 16 dict def
/chessboardinv{chessboardinvdict begin
/s 30 def  %scale parameter
/ms s neg def
/2s 2 s mul def/m2s 2s neg def
/3s 3 s mul def/m3s 3s neg def
/4s 4 s mul def/m4s 4s neg def
/r 2s def%radius inversion circle

gsave
%Inversion circle
invcir  0 0 r 0 360 arc stroke
%Chessboard
2{%horizontal and vertical lines
m4s s 4s{/d exch def
   m4s d moveto 4s d lineto stroke}for
90 rotate}repeat
grestore

m4s 0 0 0 r pointinversion /py exch def /px exch def
/c px 2 div  def
/rm4s c abs def

250 0 translate

newpath
m4s m4s moveto m4s 4s lineto 4s 4s lineto 4s m4s lineto closepath
%
m4s dup 0 0 r pointinversion moveto
c  0        rm4s 270  90 arcn
0  c  neg rm4s 180   0 arcn
c  neg 0  rm4s  90 -90 arcn
0 c         rm4s  0 -180 arcn
closepath
eoclip

%draw inverse chessboard
2{%horizontal and vertical lines
m4s s 4s{/d exch def
        d 0 ne
       {m4s d 4s d 0 0 r lineinversion /ri exch def /y exch def /x exch def
       x y ri 0 360 arc stroke}
       {gsave m4s d moveto 4s d lineto stroke grestore}ifelse
         }for
   -90 rotate}repeat
end}def
chessboardinv showpage
%%EOF
```

Inversion of a smiley pattern

can be done by use of the PostScript operator `pathforall`. This operator appends to the current path. It is not straightforward how to stroke the path created by `pathforall`, separately. I call the used technique[25] partial delayed execution, which can be useful as shown in this case. The pattern is inverted in two parts: eyes, nose and mouth by `pathforall` and the circumference by `circleinversion`.



In the right picture, with 2^{nd} level inversions, the operators `pointinversion` and `circleinversion` are invoked repeatedly: the invoke for the 1^{st} level inversion is immediately followed by the invoke for the 2^{nd} level inversion in the procedures for `pathforall`.

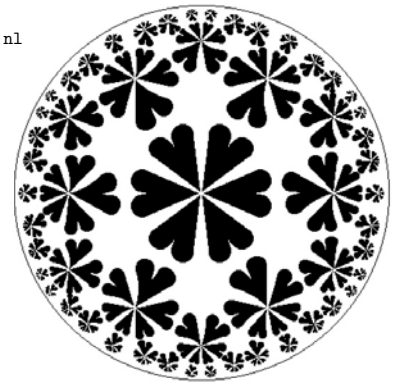
PS code with 1^{st} level inversions only. It demonstrates how to use fruitfully `pathforall`.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: pathforall use, cgl 2010
%%BoundingBox: 150 250 250 350
%%BeginSetup
%%Endsetup
(c:\\PSlib\\PSlib.eps) run
/rs 12.5 def % size smiley
r 0 moveto 0 0 r 0 360 arc % surrounding circle r=50
/smiley{% path of inner smiley
-5 5 moveto 3 0 rlineto
5 5 moveto -3 0 rlineto
0 2.5 moveto 0 -2.5 lineto
-5 -5 moveto 5 -1.25  rlineto 5 1.25 rlineto
}def
smiley % creates path of inner part of smiley
rs 0 moveto 0 0 rs 0 360 arc stroke % circumference of smiley
stroke % central smiley is drawn
%
/Ix 2r def /Iy 0 def /Ir rsqrt3 def % Inversion circle centre and radius
3{smiley  % creates path of inner smiley
[{ Ix Iy Ir pointinversion /moveto cvx}
 { Ix Iy Ir pointinversion /lineto cvx}
 { }
 { }
 pathforall
] cvx      % make array executable
newpath    % delete path of central smiley
exec       % path of inverted smiley is created (only)
stroke
 0 0 rs Ix Iy Ir circleinversion /ir exch def /iy exch def /ix exch def
newpath ix iy ir 0 360 arc stroke         % circumference of inverted smiley
Ix Iy 120 rot /Iy exch def /Ix exch def   % rotate centre of inversion circle
}repeat
%
/Mx r def /My rdsqrt3 def                 % centre of circles (sides triangle)
auxlin     % from libray: dashed auxiliary lines
3{newpath Ix Iy Ir 150 210 arc stroke     % arc of inversion circles
   120 rotate}repeat
/R {hr hr sqrt3 mul}def
6{R moveto Mx My My 150 270 arc stroke    % inverted sides
   60 rotate}repeat
showpage
```

[25] See `http://www.acumentraining.com/acumenjournal.html`.

Inversion of hearts

A Metafont programof more than a decade ago, , which should be converted into PostScript. But à la, the picture is nice and unusual.



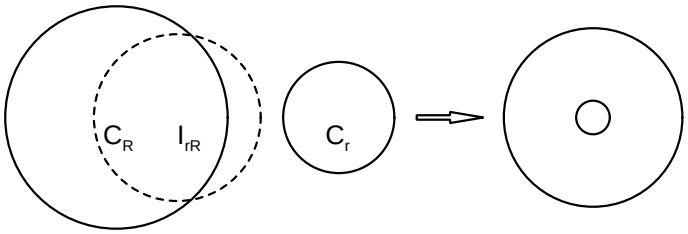
```
%Tessellation {6,4}. Circle limits [6,4] hearts a la Escher, March 2001, cgl@hetnet.nl
tracingstats:=proofing:=1;screenstrokes;%To follow drawing sequence
def openit = openwindow currentwindow
   from origin to (screen_rows,screen_cols) at (-1.1R,1.1R)enddef;
pair p[], q[],  m[], mi; path heart[], hearti;
input dunhammacros.mf
def drawinversions(expr pad)=
for k= 1 upto 6: fill (pad rotated 60k); endfor
enddef;
R=200;                                  %parameter of picture size
m= R*w2; w2=1.4142; w3=1.7321; w6=w2*w3; hdR:=(w3-1)/(2w2); h:=R*hdR;
%data first corner pnt, and halfpoint q, (m1,R) is major reflection circle
m1:=(m,0);
p1:=(w3*h,h); p11:=p1+.5(R*w6,R*w2);
q1=(.4142R,0);                          %q3=q1 rotated 120
m7:=.5[p1,p11]; r7=length(m7-p1);%circle (m7,r7) is the inversion of (q3,-q3)
heart0:=origin+(.01R,0)--p1-(.16R,.14R)..p1-(.13R,.13R)..q1+(-.08R,.01R)--q1-(.13R,0);
heart0:=heart0--(reverse heart0)reflectedabout(origin,q1)--cycle;%a symmetrical pathi
pickup pencircle scaled .1;
for k= 1 upto 6:
fill heart0; %0th layer
pathinversion(heart0,m1,R);
drawinversions(hearti);%first layer, side reflections
heart3:=hearti; heart4:=hearti; heart5:=hearti;
%second layer
 inversion((-m,0),R,m1,R);
   pathinversion(heart3,mi,ri);drawinversions(hearti);
 inversion((m,0)rotated120,R,m1,R);
   pathinversion(heart4,mi,ri);drawinversions(hearti);
 inversion((m,0)rotated-120,R,m1,R);
   pathinversion(heart4,mi,ri);drawinversions(hearti);
%
pathinversion(heart0,m7,r7);drawinversions(hearti);
%next layer, second
 heart13:=hearti; heart14:=hearti; heart15:=hearti;heart16:=hearti;
 inversion((m,0)rotated 120,R,m7,r7);
 m23:=mi;r23:=ri;
  pathinversion(heart13,m23,r23);drawinversions(hearti);
 inversion(((m,0)rotated -60),R,m7,r7);
 m24:=mi;r24:=ri;
  pathinversion(heart14,m24,r24);drawinversions(hearti);
 inversion((-m,0),R,m7,r7);
 m25:=mi;r25:=ri;
  pathinversion(heart15,m25,r25);drawinversions(hearti);
 inversion((m,0)rotated -120,R,m7,r7);
 m26:=mi;r26:=ri;
  pathinversion(heart16,mi,ri);drawinversions(hearti);
heart0:=heart0 rotated 60;
endfor
draw fullcircle scaled 2R;
end
```

Any two disjunct circles can be made concentric

by inversion by picking the inversion centre as one of the so-called limiting points.
`http://mathworld.wolfram.com/LimitingPoint.html`.

The dashed circle is the inversion circle I with abscissa of the centre $\mathrm{I}_{rR} = \frac{l \pm \sqrt{l^2 - 4d^2R^2}}{2d}$ with $l = d^2 - r^2 + R^2$. The main circle is $C_{R,(0,0)}$ and the small circle $C_{r,(d,0)}$. The radius of the inversion circle determines the size of the concentric circles. `http://mathworld.wolfram.com/ConcentricCircles.html`.

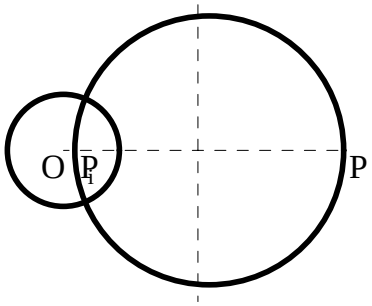
```
%!PS-Adobe-3.0
%%Title: Two circlesinverted  into two concentric circles. CGL, 2010
%%BoundingBox: -51 -51 256 51
%%BeginSetup
%%EndSetup
%%BeginProlog(C:\\PSlib\\PSlib.eps) run %PS library
/concentricinversiondict 12 dict def
/concentricinversion{concentricinversiondict begin
/r 25 def /hr r 2 div def /2r r dup add def %radius of small circle
/R 50 def                                   %radius of big circle
/d 100 def     /2d d d add def              %distance between circle centres
/O {0 0} def
/Ix d d mul r r mul sub R R mul add
     d d mul r r mul sub R R mul add
     dup mul 2 d mul R mul dup mul sub sqrt
     sub
     2d div def % function of r and R

gsave
O moveto   -6 -12 rmoveto (C)
  H12pt setfont show 0 -3 rmoveto (R) H7pt setfont show
Ix 0 moveto 0 -12 rmoveto (I)
  H12pt setfont show 0 -3 rmoveto (rR) H7pt setfont show
d 0 moveto -6 -12 rmoveto (C)
  H12pt setfont show 0 -3 rmoveto (r) H7pt setfont show
grestore
/R1 R .75 mul def
gsave Ix 0 R1 0 360 arc [3] centerdash stroke grestore  %Inversion circle
gsave d 0 r 0 360 arc stroke grestore       %Cr
gsave 0 0 R 0 360 arc stroke grestore       %CR

135 0 translate
gsave newpath 0 0 30 0 2 5 15 arrow stroke grestore
30 0 translate

d 0 r   %small circle at (d,0)
Ix 0 R1 %inversion circle
circleinversion
/rinv exch def /yinv exch def /xinv exch def
xinv yinv rinv 0 360 arc stroke

0 0 R   %big circle at (0,0)
Ix 0 R1 %inversion circle
circleinversion
/rinv exch def /yinv exch def /xinv exch def
xinv yinv rinv 0 360 arc stroke
end}def
%%EndProlog
concentricinversion showpage
%%EOF
```



Orthogonal circles I

The problem is given a circle and a point P outside the circle construct a circle through P which cuts the original circle orthogonally.

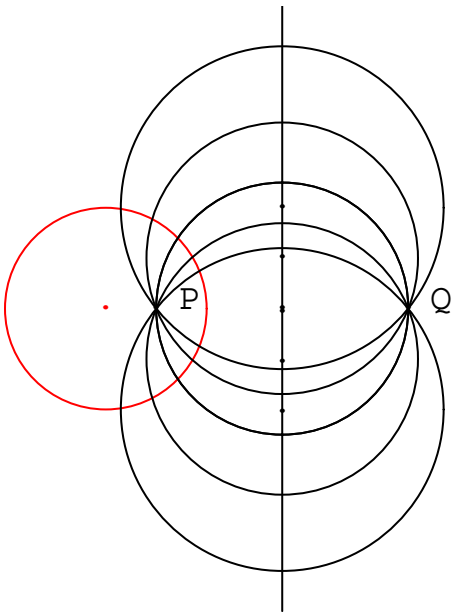
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Orthogonal circles. cgl 2014
%%BoundingBox: -11  -27 55 27
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/Times-Roman 6 selectfont
0 0 10 0 360 arc stroke
-4 -5 moveto (O) show
50 0 moveto 1 -5 rmoveto (P) show
            3 -5 moveto (P) show -2 -1 rmoveto /Times-Roman 4 selectfont (i) show
gsave 24 -27 moveto 24 27 lineto
0 0 moveto 50 0 lineto .1 setlinewidth
[2] centerdash stroke grestore
newpath 26 0 24 0 360 arc stroke
showpage
%%EOF
```



Orthogonal circles II

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

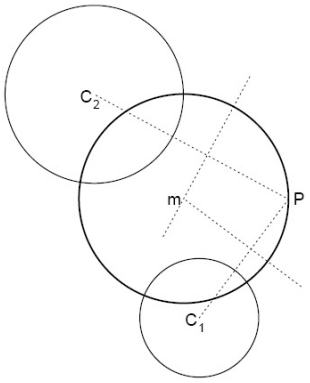
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Orthogonal circle bundle. cgl 2014
%%BoundingBox: -105 -150 125 150
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/orthogonalcirclebundledict 11 dict def
/orthogonalcirclebundle{orthogonalcirclebundledict begin
/r 50 def /Courier 18 selectfont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



Orthogonal circles III

The problem is given two circles and a point P outside the circles construct a circle through P which cuts the original circles orthogonally. Construct the inversions of P in circle I and II. The intersection point of the middle perpendiculars is the centre of the orthogonal circle.

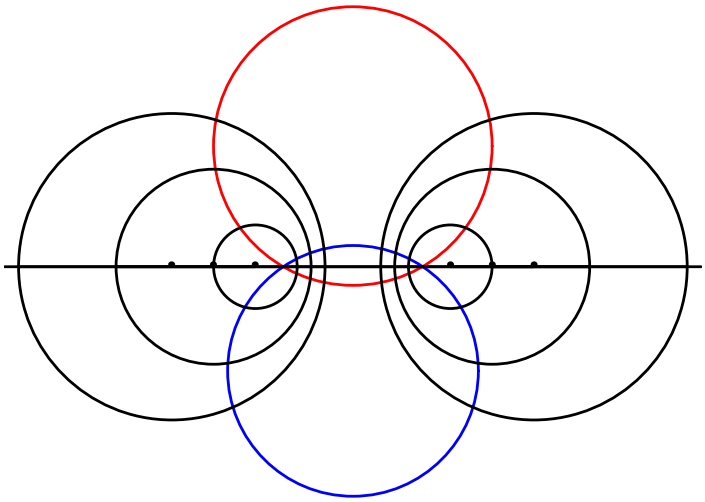
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Circle orthogonal to two circles through P. cgl April 2010
%%BoundingBox: 51 50 357 457
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
H14pt setfont
/r1 50 def/x1 0 def/y1 r1 -2 mul def
/r2 r1 1.5 mul def  /x2 r1 r2 add sqrt2 div neg def  /y2 x2 neg def
/r3 r1 2 mul def    /x3 r1 r3 add sqrt2 div  def       /y3 x3 def
x1 y1 r1 0 360 arc stroke
x2 y2 r2 0 360 arc stroke

/px 75 def /py 0 def /p {px py} def
x1 y1 moveto -12 -7 rmoveto (C) show
                     1 -5 rmoveto (1) gsave H10pt setfont show grestore
x2 y2 moveto -12 -7 rmoveto (C) show
                     1 -5 rmoveto (2) gsave H10pt setfont show grestore
p moveto 4 -5 rmoveto (P) show

x1 y1 r1 x2 y2 r2 px py circleorthogonaltotwocircles
/r exch def /my exch def /mx exch def
auxlin
x1 y1 moveto p lineto x2 y2 lineto stroke
mx 14 sub my 5 sub moveto (m) show
wanlin
newpath mx my r 0 360 arc stroke
showpage
%%EOF
```



Orthogonal circles IV

The problem is given two intersecting circles construct a circle bundle which cut the original circles orthogonally. The centres of the circle bundle are colinear.

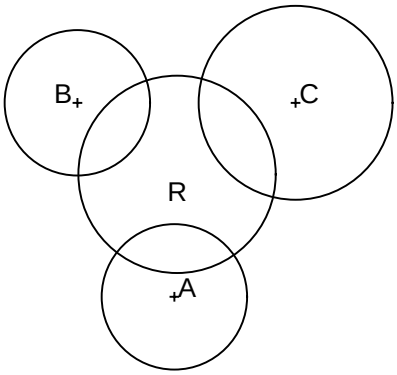
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Circles Orthogonal to 2 circles. cgl 2014
%%BoundingBox: -125 -90 125 100
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/circlesorthogonal2circlesdict 12 dict def
/circlesorthogonal2circles{circlesorthogonal2circlesdict begin
/Courier 18 selectfont
/rr 50 def/s 25 def /mr rr rr mul  s s mul sub sqrt def
gsave 1 0 0 setrgbcolor 0 mr rr 0 360 arc stroke grestore
/rb 45 def          /mb rb rb mul  s s mul sub sqrt neg def
gsave 0 0 1 setrgbcolor 0 mb rb 0 360 arc stroke grestore
-125 0 moveto 125 0 lineto stroke
2{/m1 35 def /r1 15 def m1 0 r1 0 360 arc stroke
m1 0 moveto (.) centershow
/m2 50 def /r2 35 def m2 0 r2 0 360 arc stroke
m2 0 moveto (.) centershow
/m3 65 def /r3 55 def m3 0 r3 0 360 arc stroke
m3 0 moveto (.) centershow
-1 1 scale}repeat
end}def
circlesorthogonal2circles showpage
%%EOF
```



Orthogonal circles V. Radical circle

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

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

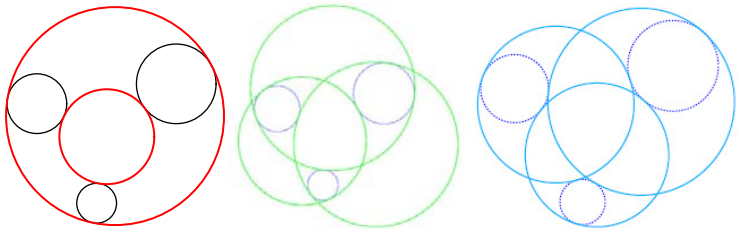


An ill-posed subproblem —touching point of 2 circles—has been reformulated in a well-posed problem.

Apollonius problem

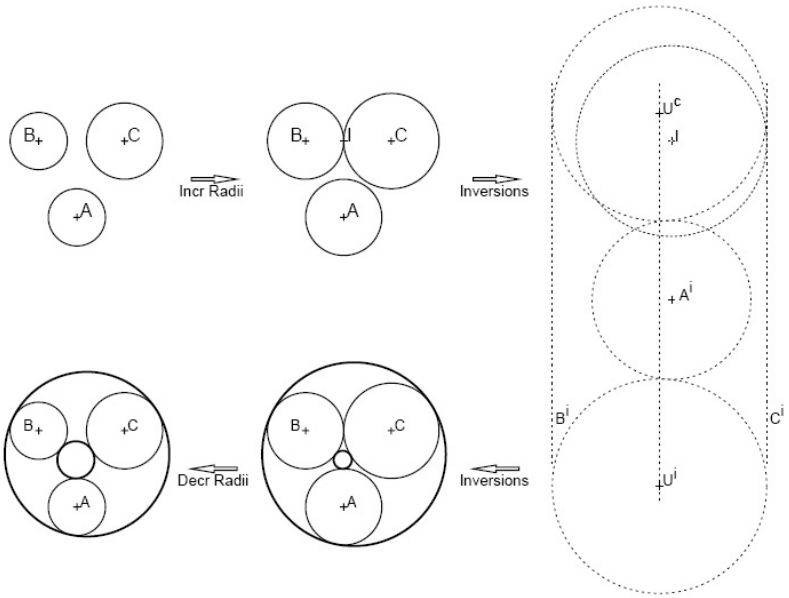
Given three disjunct circles find the circles touching the circles. In *Circle Inversion* the problem has been solved and all 8 solutions have been programmed in MetaPost and PostScript. A universal PostScript definition `apollonius` has emerged. The definition for a slight variant for the case when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

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



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

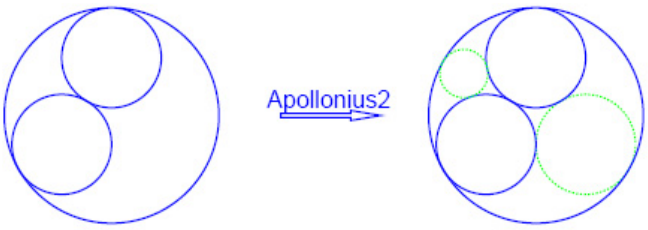
Below a picture of the inversion method.



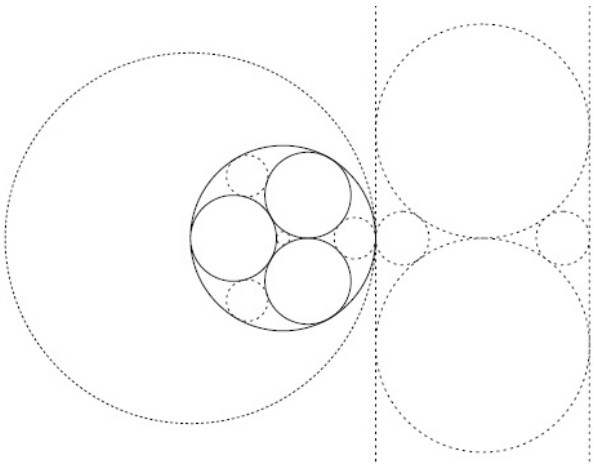
Apollonius problem, variant non-disjunct circles

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

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

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

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

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

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



Tiling a circle by circles via inversion, tedious.



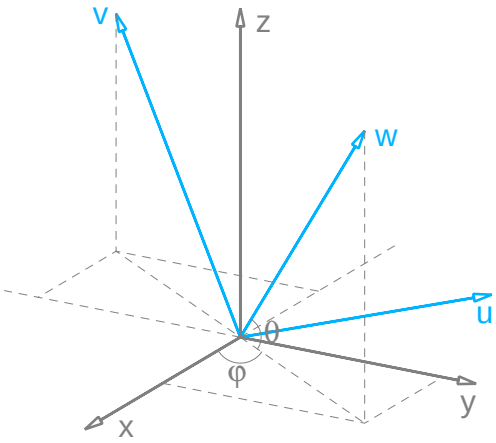
Beautiful Apollonius gaskets borrowed from the WWW

If we start with one circle and within this circle a series of circles which touch each other, then one may obtain

2D pictures as projection of 3D emulated objects

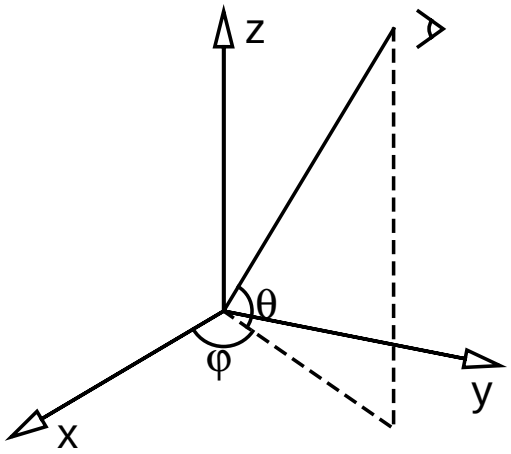
Coordinate axes

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -93 -45 115 130
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\Pslib.eps) run % used: anglemark s ptp S12pt H12pt
/xyzuvwdict 20 dict def

xyzuvwdict begin
/scalingfactor 30  def /phi 30 def /theta 20 def
1.415 setmiterlimit
/origin {0 0} def
/x { 4    s  0       0     ptp} def
/y { 0       3.5  s  0     ptp} def
/z { 0       0       4.5 s ptp} def
/p { 2    s  3    s  4   s ptp} def%w-axis
/pp{ 2    s  3    s  0     ptp} def
/u {-3    s  2    s  0     ptp} def
/v { -2    s -3 s   13 4 div s    ptp} def
/w { p }def
/x-axis { origin x .1 3 7 arrow} def
/y-axis { origin y .1 3 7 arrow} def
/z-axis { origin z .1 3 7 arrow} def
/u-axis { origin u .1 3 7 arrow} def%orthogonal to OP, viwpoint
/v-axis { origin v .1 3 7 arrow} def
/w-axis { origin w .1 3 7 arrow} def%viewpoint
end
%%EndProlog
/xyzuvw{xyzuvwdict begin
gsave greenblue
u-axis  v-axis w-axis stroke
%annotation
u moveto -6 -11 rmoveto (u) H12pt setfont show
v moveto -9 -4 rmoveto (v)               show
w moveto 4 -6 rmoveto (w)               show
grestore%because of color
.5 setgray
x moveto  13 -3  rmoveto (x) H12pt setfont show
y moveto -6  -11 rmoveto (y)               show
z moveto  6  -9  rmoveto (z)               show
x-axis y-axis z-axis stroke
origin moveto -4 s 0 0 ptp lineto%x-axis negative
origin moveto 0 -3.5 s 0 ptp lineto%y-axis negative
origin moveto pp lineto p lineto
2 s 0 0 ptp moveto pp lineto 0 3 s 0 ptp  lineto
-2 s 0 0 ptp moveto -2 s -3 s 0 ptp  lineto 0 -3 s 0 ptp lineto
v moveto -2 s -3 s 0 ptp lineto origin lineto
%0 0 4 s ptp moveto p lineto u lineto
.1 setlinewidth [3] 0 setdash stroke
%annotation
[] 0 setdash p pp origin 8 anglemark %theta
pp moveto currentpoint
2 0 0 ptp  moveto currentpoint
origin 10 anglemark %phi
-5 -17 moveto (j) S12pt setfont show % phi1
 9  -2 moveto (q)               show % theta
end}def
xyzuvw showpage
```



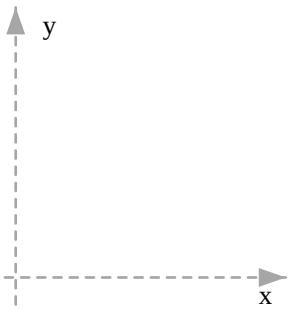
Coordinate axes with viewpoint

```
%!PS-Adobe-3.0 EPSF-3.0  %Assenstelsel met azimuth en inclination
%%Title: Coordinate axes right-screw. cgl, kisa1@xs4all.nl 2011
%%BoundingBox: -65 -39 83 89
%%BeginSetup
%%EndSetup
(C:\\PSlib\\Pslib.eps) run % used: anglemark s ptp S12pt H12pt
%%EndProlog
/scalingfactor 30  def /phi 30 def /theta 20 def
/origin {0 0} def /p { 2 s  3 s 4 s ptp} def  /pp{ 2 s  3 s   0 ptp} def
/x-axis { origin 4 s 0   0   ptp .01 5 10 arrow } def
/y-axis { origin 0   3 s 0   ptp .01 5 10 arrow } def
/z-axis { origin 0   0   3 s ptp .01 5 10 arrow } def
4 s 0   0   ptp moveto  13 -3 rmoveto (x) H12pt setfont show
 0  3 s 0   ptp moveto -8  -11 rmoveto (y)               show
 0  0   3 s ptp moveto  6  -9 rmoveto (z)               show
p moveto 7 5 rmoveto 7 -5 rlineto -7 -5 rlineto stroke
p moveto 7 -5 rmoveto currentpoint % point left leg
p moveto 7  5 rmoveto currentpoint % point right leg
p moveto 14 0 rmoveto currentpoint % corner
4 anglemark%eye
pp moveto currentpoint
2 0 0 ptp  moveto currentpoint
origin 10 anglemark %phi
p pp origin 8 anglemark %theta
-5 -17 moveto (j) S12pt setfont show % phi1
 9  -2 moveto (q)               show % theta
x-axis y-axis z-axis
origin moveto p  lineto               stroke
origin moveto pp lineto  p  lineto [3] 0 setdash stroke
showpage
%%EOF
```



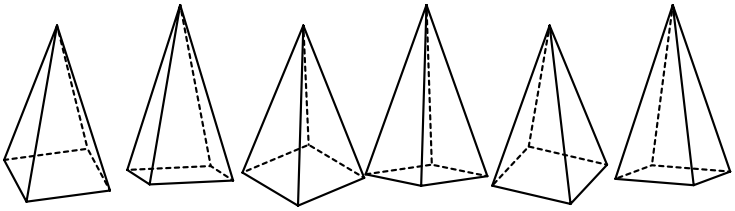
xy-axes

```
%%Title: xy-diagram
%%BoundingBox: -6 -11 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/xy-diagram{%xmin xmax and used font global
%coordinate axes
.65 setgray [3] 0 setdash
xmin 0 moveto xmax 0 lineto
0 xmin moveto 0 xmax lineto stroke
1 setlinewidth [] 0 setdash 0 setgray
xmax .9 mul -10 moveto (x) show
10 xmax .9 mul moveto (y) show
%arrowheads coordinate axes
.65 setgray
0 xmax .9 mul 0 xmax 1 7 10 arrow fill %y-axis
xmax .9 mul 0 xmax 0 1 7 10 arrow fill %x-axis
%
0 setgray%black
[] 0 setdash %normal
%xmin dup moveto xmax dup lineto stroke%diagonal y=x
-1.9 s  dup -1.9 mul 100 sub moveto
-1.8 .1 2.2 {/x exch def
    x s dup dup x mul 100 sub lineto
   }for stroke
}bind def
%%EndProlog
/Times-Roman 10 selectfont
/xmin -10 def /xmax 100 def
xy-diagram
showpage
%%EOF
```

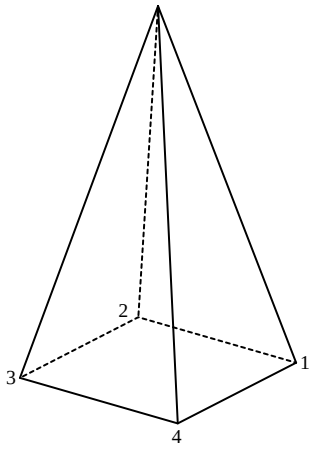


Pyramids

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



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

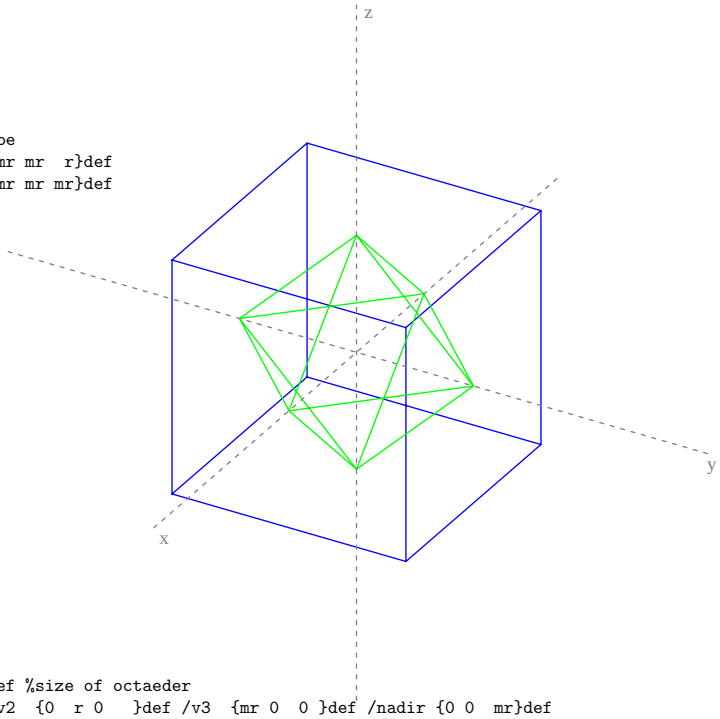



```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Pyramid in projection, cgl 1997
%%BoundingBox: -90 -40 90 190
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/pyramid{/ptp{/z exch def/y exch def/x exch def
                       x -.6 mul y .8 mul add
                        -4 x mul -3 y mul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

Octaeder in cube

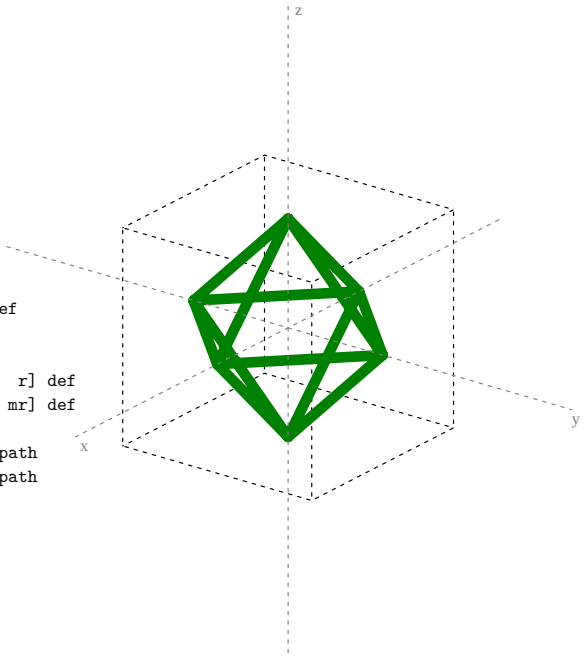
Essentially an old program. Nowadays I would use `ptp` in the definition of the points.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Cube-Oktaeder
%%BoundingBox: -280 -280 280 280
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/cubedict 12 dict def
/cube{cubedict begin/r 100 def /mr r neg def %size of cube
/u1 { r mr  r}def/u2 { r  r  r}def/u3 {mr  r  r}def/u4 {mr mr  r}def
/b1 { r mr mr}def/b2 { r  r mr}def/b3 {mr  r mr}def/b4 {mr mr mr}def
u1 ptp moveto %u1%upper
u2 ptp lineto %u2
u3 ptp lineto %u3
u4 ptp lineto %u4
u1 ptp lineto %u1
b1 ptp moveto %b1 %lower
b2 ptp lineto %b2
b3 ptp lineto %b3
b4 ptp lineto %b4
b1 ptp lineto %b1
%edges (vertical)
u1 ptp moveto %u1
b1 ptp lineto %b1
u2 ptp moveto %u2
b2 ptp lineto %b2
u3 ptp moveto %u3
b3 ptp lineto %b3
u4 ptp moveto %u4
b4 ptp lineto %b4
end}def
/octaederdict 12 dict def
/octaeder{octaederdict begin
/r 100 def /mr r neg def  /2r r 2 mul def  /m2r 2r neg def %size of octaeder
top {0 0  r  }def/v4  {0 mr 0  }def /v1  {r 0  0  }def /v2  {0  r 0   }def /v3  {mr 0  0 }def /nadir {0 0  mr}def
top ptp moveto %top
v4 ptp lineto
v1 ptp lineto %v1
v2 ptp lineto %v2
v3 ptp lineto %v3
v4 ptp lineto %v4
%
top ptp moveto %top
v1  ptp lineto %v1
top ptp moveto %top
v2  ptp lineto %v2
top ptp moveto %top
v3  ptp lineto %v3
%
nadir ptp moveto %nadir
v4 ptp lineto %v4
nadir ptp moveto
v1 ptp lineto %v1
nadir ptp moveto
v2 ptp lineto %v2
nadir ptp moveto
v3 ptp lineto %v3
}def
/cubeoktaederdict 5 dict def
/cubeoktaeder{cubeoktaederdict begin  /Times-Roman 14 selectfont
cube     0 0 1 setrgbcolor stroke
octaeder 0 1 0 setrgbcolor stroke
0.5 0.5 0.5 setrgbcolor
%x-y-z coordinate axis
/r r 3 mul def /mr r neg def
mr 0 0 ptp moveto r  0 0 ptp lineto 4 -12 rmoveto (x) show
0 mr 0 ptp moveto 0  r 0 ptp lineto 0 -12 rmoveto (y) show
0 0 mr ptp moveto 0 0  r ptp lineto 6 -12 rmoveto (z) show [3 5]6 setdash stroke
end}def
/phi 30 def/theta 30 def cubeoktaeder
showpage
%%EOF
```



Variant, based on arrays

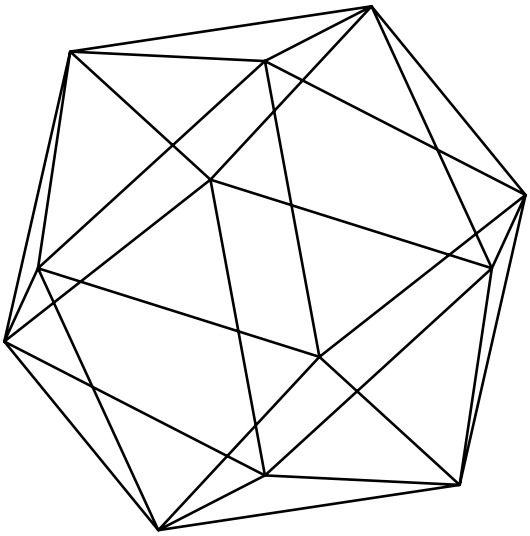
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Cube-Oktaeder(arrays). cgl 2009
%%BoundingBox: -320 -320 320 320
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/octaederincubedict 32 dict def
/octaederincube%cgl 2009 (symmetric in projection, with arrays)
{octaederincubedict begin
/ptpa{/vec exch def             %Stack [ px py pz ] -> (x, y) pair
   vec 0 get  -.6 mul vec 1 get .8 mul add
   -4 vec 0 get mul -3 vec 1 get  mul add 12 vec 2 get mul add 13 div}def
%Coordinates from center of gravity, so edge is 2r
/r 100 def /mr r neg def             %size of cube, edge 2r
/u1 [  r mr  r] def /u2 [  r  r  r] def /u3 [ mr  r  r] def /u4 [ mr mr  r] def
/b1 [  r mr mr] def /b2 [  r  r mr] def /b3 [ mr  r mr] def /b4 [ mr mr mr] def
%
/cube{u1 ptpa moveto u2 ptpa lineto u3 ptpa lineto u4 ptpa lineto closepath
      b1 ptpa moveto b2 ptpa lineto b3 ptpa lineto b4 ptpa lineto closepath
      %edges (vertical)
      u1 ptpa moveto b1 ptpa lineto
      u2 ptpa moveto b2 ptpa lineto
      u3 ptpa moveto b3 ptpa lineto
      u4 ptpa moveto b4 ptpa lineto}def

cube gsave [3 5]6 setdash stroke grestore %ordinary cube dashed

/o1 [r 0 0]def /o2 [0 r 0]def   /o3 [mr 0 0]def /o4 [0 mr 0]def /o5 [0 0 r]def /o6 [0 0 mr]def
/octaeder{o1 ptpa moveto o2 ptpa lineto o3 ptpa lineto o4 ptpa lineto closepath
o5 ptpa moveto o1 ptpa lineto
o5 ptpa moveto o2 ptpa lineto
o5 ptpa moveto o3 ptpa lineto
o5 ptpa moveto o4 ptpa lineto
o6 ptpa moveto o1 ptpa lineto
o6 ptpa moveto o2 ptpa lineto
o6 ptpa moveto o3 ptpa lineto
o6 ptpa moveto o4 ptpa lineto}def
newpath
gsave
9 setlinewidth 1 setlinecap 1 setlinejoin
octaeder 0 0.5 0 setrgbcolor stroke
grestore

0.5 0.5 0.5 setrgbcolor
%x-y-z coordinate axes
/ptp{/z exch def/y exch def/x exch def
   x -.6 mul y .8 mul add
   -4 x mul -3 y mul add 12 z mul add 13 div}def
/r r 3 mul def /mr r neg def
mr 0 0 ptp moveto r  0 0 ptp lineto 4 -12 rmoveto (x) show
0 mr 0 ptp moveto 0  r 0 ptp lineto 0 -12 rmoveto (y) show
0 0 mr ptp moveto 0 0  r ptp lineto 6 -12 rmoveto (z) show
[3 5]6 setdash stroke
end }def %octaederincube

/phi 30 def /theta 30 def /Times-Roman 14 selectfont
octaederincube showpage
%%EOF
```



Icosaeder

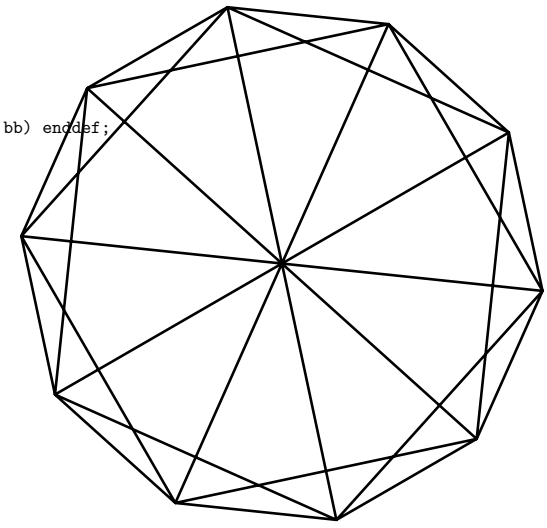
Borrowed from Lauwerier(1987): : Meetkunde met de microcomputer. The formula for the cornerpoints are nice and symmetric. Happily each coordinate can be scaled for reazonable sized pictures.

```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Ikosaeder. cgl 2014. Lauwerier Meetkunde met de microcmputer
%%BoundingBox: -101 -101 101 101
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/ikosaederdict 15 dict def
/ikosaeder{ikosaederdict begin
/r 100 def /theta 45 def /phi 60 def
/top {0 0 sqrt5 2 div r mul ptp}def
/bottom{0 0 sqrt5 -2 div r mul ptp}def
/p0{r  0 .5 r mul ptp}def
/p2{r  72 cos mul r 72 sin mul .5 r mul ptp}def
/p4{r  144 cos mul r 144 sin mul .5 r mul ptp}def
/p6{r  216 cos mul r 216 sin mul .5 r mul ptp}def
/p8{r  288 cos mul r 288 sin mul .5 r mul ptp}def
/p1{r  36 cos mul r 36 sin mul -.5 r mul ptp}def
/p3{r  108 cos mul r 108 sin mul -.5 r mul ptp}def
/p5{r  180 cos mul r 180 sin mul -.5 r mul ptp}def
/p7{r  252 cos mul r 252 sin mul -.5 r mul ptp}def
/p9{r  324 cos mul r 324 sin mul -.5 r mul ptp}def
top moveto p0 lineto
top moveto p2 lineto
top moveto p4 lineto
top moveto p6 lineto
top moveto p8 lineto
p0 moveto p2 lineto p4 lineto p6 lineto p8 lineto closepath
bottom moveto p1 lineto
bottom moveto p3 lineto
bottom moveto p5 lineto
bottom moveto p7 lineto
bottom moveto p9 lineto
p1 moveto p3 lineto p5 lineto p7 lineto p9 lineto closepath
p0 moveto p1 lineto p2 lineto p3 lineto p4 lineto p5 lineto p6 lineto
p7 lineto p8 lineto p9 lineto closepath
stroke
end}def
ikosaeder showpage
%%EOF
```



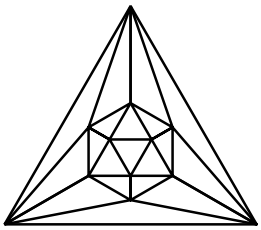
Old Metafont program

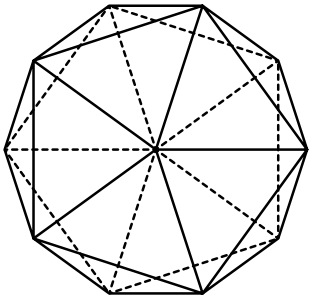
```
%Ikosaeder. April 96. cgl@rc.service.rug.nl
def pointtopair(expr x,y,z)=
(-x*cosd aa + y*sind aa, -x*sind aa * sind bb -y*cosd aa * sind bb + z*cosd bb) enddef;
pair p[]; picture blackpicture,pic[];
pickup pencircle scaled 1;
r:=50; n:=12;m:=0;
for b=-10,10: bb:=b;
for a=-10step-30until-100:%Varying viewing angles
aa:=a;
p10:=pointtopair(0,0,.5r*sqrt5); p11:=-p10;
for k:=0 step 2 until 8:
   p[k]  :=pointtopair(r*cosd36k,r*sind36k,.5r);
   p[k+1]:=pointtopair(r*cosd36(k+1),r*sind36(k+1),-.5r);endfor;
p[-2]:=p8;p[-1]:=p9;
for k:=0 step 2 until 8: draw p[k-1]--p[k]--p[k+1]; endfor
for k:=0 step 2 until 8:
   draw p[k-2]--p[k]; draw p[k-1]--p[k+1]; endfor;
for k:=0 step 2 until 8: draw p[10]--p[k];endfor
for k:=0 step 2 until 8: draw p[11]--p[k+1]; endfor
pic[m]:=currentpicture;m:=m+1;
clearit;
endfor endfor
```



Icosahedron II

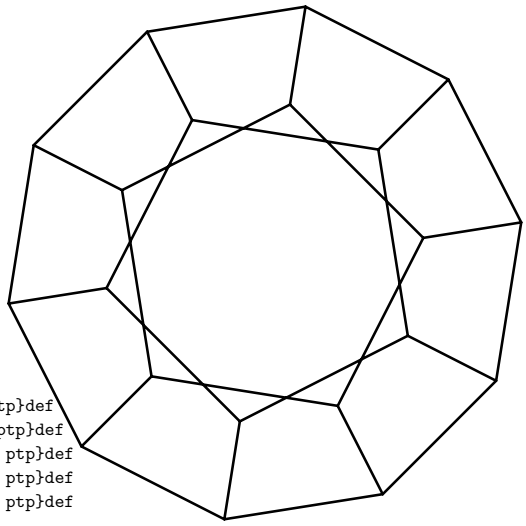
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: icasohedron, cgl 1997
%%BoundingBox: -51 -60 222 60
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run % contains growthspiral
/icosahedrondict 8 dict def
/icosahedron{icosahedrondict begin %tilLxii
2 setlinejoin 1 setlinecap
/s 100 def /ss 1.732 s mul def
/p1{.1667 s mul  -.0555 ss mul}def
/p2{.5    s mul  -.1667 ss mul}def
/p3{0            -.1111 ss mul}def
/p4{0            -.0555 ss mul}def
3{gsave
  2{p1 moveto p2 lineto p3 lineto
    p4 lineto p1 lineto p3 lineto
    -1 1 scale}repeat stroke
  grestore 120 rotate}repeat
p2 moveto
3{120 rotate p2 lineto}repeat stroke
p4 moveto
3{120 rotate p4 lineto}repeat stroke
%
/pentagon{5{0 0 moveto
   r 0 lineto
   72 rotate
   r 0 lineto}repeat}def
/r 60 def
160 0 translate r 0 moveto
10{36 rotate r 0 lineto}repeat
pentagon stroke
36 rotate
[2]1 setdash
pentagon stroke
end}def
%%EndProlog
icosahedron showpage
%%EOF
```





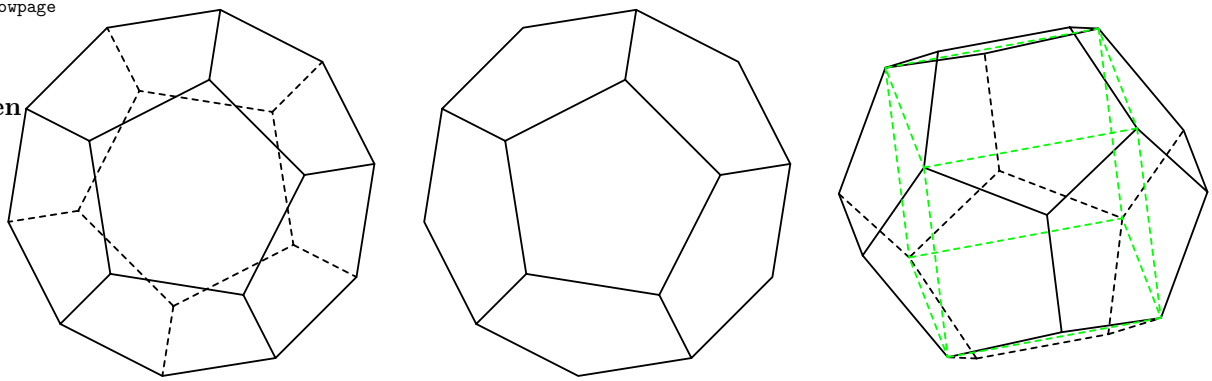
Dodecaeder

Borrowed from Lauwerier(1987): Meetkunde met de microcomputer. The formula for the cornerpoints are nice and symmetric. Unfortunately each coordinate can't be scaled for a reazonable sized picture. I kludged around. For the number of cornerpoints, H, the number of sides, S, and the number of ribs, R, there exists the relation: H+S=R+2. Wikipedia contains interesting data.



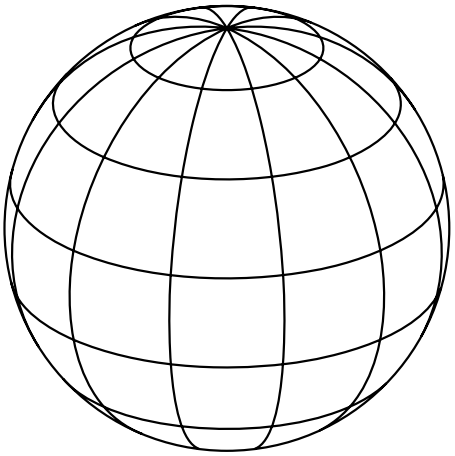
```
%!PS-Adobe-3.0  EPSF-3.0
%%Title: Dodekahedron. cgl 2014
%%BoundingBox: -102 -102 102 102
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/dodecahedrondict 28 dict def
/dodecahedron{dodecahedrondict begin
gsave .01 setlinewidth 100 100 scale
/a -1 sqrt5 add 2 div def/b 1 def
/p 1 sqrt5 add 4 div def /q 3 sqrt5 sub 4 div def
%/psi 36 def
/phi 45 def/theta 90 def%projection angles
/A0{a                 0          p ptp}def /B0{b            0             q ptp}def
/A2{a   72 cos mul a  72 sin mul p ptp}def /B2{b  72 cos mul b 72 sin mul  q ptp}def
/A4{a  144 cos mul a 144 sin mul p ptp}def /B4{b 144 cos mul b 144 sin mul  q ptp}def
/A6{a  216 cos mul a 216 sin mul p ptp}def /B6{b 216 cos mul b 216 sin mul  q ptp}def
/A8{a  288 cos mul a 288 sin mul p ptp}def /B8{b 288 cos mul b 288 sin mul  q ptp}def

A0 moveto A2 lineto A4 lineto A6 lineto A8 lineto closepath
A0 moveto B0 lineto
A2 moveto B2 lineto
A4 moveto B4 lineto
A6 moveto B6 lineto
A8 moveto B8 lineto

/D1{a   36 cos mul a  36 sin mul p neg ptp}def /C1{b  36 cos mul b  36 sin mul  q neg ptp}def
/D3{a  108 cos mul a 108 sin mul p neg ptp}def /C3{b 108 cos mul b 108 sin mul  q neg ptp}def
/D5{a neg                  0              p neg ptp}def /C5{b neg             0                       q neg ptp}def
/D7{a  252 cos mul a 252 sin mul p neg ptp}def /C7{b 252 cos mul b 252 sin mul  q neg ptp}def
/D9{a  324 cos mul a 324 sin mul p neg ptp}def /C9{b 324 cos mul b 324 sin mul  q neg ptp}def

B0 moveto C1 lineto B2 lineto C3 lineto B4 lineto C5 lineto B6 lineto C7 lineto B8 lineto C9 lineto closepath

D1 moveto D3 lineto D5 lineto D7 lineto D9 lineto closepath
D1 moveto C1 lineto
D3 moveto C3 lineto
D5 moveto C5 lineto
D7 moveto C7 lineto
D9 moveto C9 lineto
stroke
grestore end}def
dodecahedron showpage
%%EOF
```

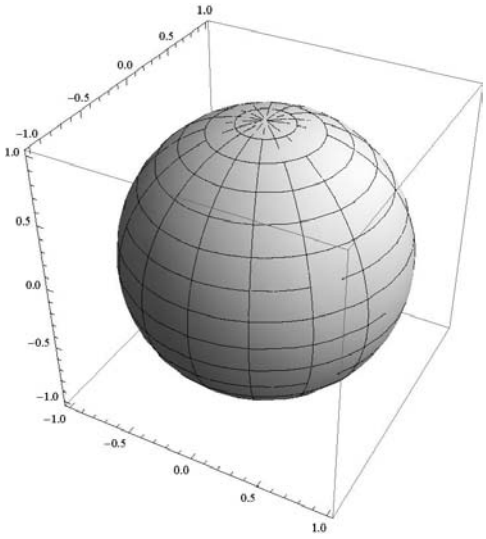
Lines hidden



Sphere with meridians and hidden lines

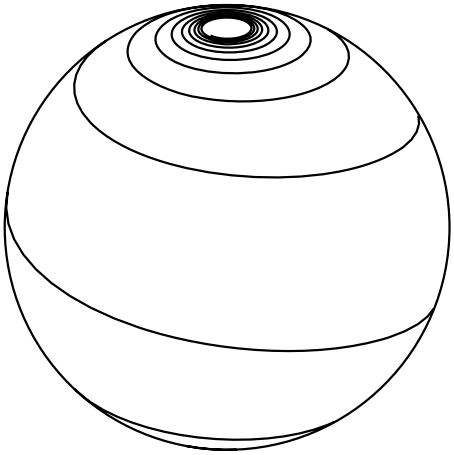
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Spere with meridans and latitude circles
%%Author: H A Lauwerier(1987): Meetkunde met de microcomputer, Epsilon
%%BoundingBox: -101 -102 101 102
%%BeginSetup
%%EndSetup
/spheremeridianslatitudesdict 25 dict def
/spheremeridianslatitudes{spheremeridianslatitudesdict begin
/m 6 def /n 6 def /r 100 def
/c .9 def /a2 1 1.41421 div def /b1 a2 1 c c mul sub sqrt mul def
/c1 c a2 mul def /c3 1 c c mul sub sqrt def
%latitude circles
1 1 m{/i exch def /t i 180 mul m 1 add div def
 /z r t cos mul def
 0 1 100{/j exch def  /s j 180 mul 50 div def
         /x{r s cos t sin mul mul}def
         /y{r s sin t sin mul mul}def
         /w{c1 x y add mul c3 z mul add}def
         /u a2 y x sub mul def
         /v c z mul b1 x y add mul sub def
         j 0 eq w 0 lt or {u v moveto}{u v lineto} ifelse
        }for %j
}for %i
%meridians
1 1 n{/i exch def /s i 180 mul n div def
 0 1 100{/j exch def  /t j 180 mul 50 div def
         /x{r s cos t sin mul mul}def
         /y{r s sin t sin mul mul}def
         /z{r t cos mul}def
         /w{c1 x y add mul c3 z mul add}def
         /u a2 y x sub mul def
         /v c z mul b1 x y add mul sub def
         j 0 eq w 0 lt or {u v moveto}{u v lineto} ifelse
        }for %j
}for %i
%circle in projection plane
r 0 moveto
1 1 100{/k exch def  /t k 180 mul 50 div def
        r t cos mul r t sin mul lineto
}for %k
stroke
end} bind def
%%Endprolog
spheremeridianslatitudes showpage
%%EOF
```



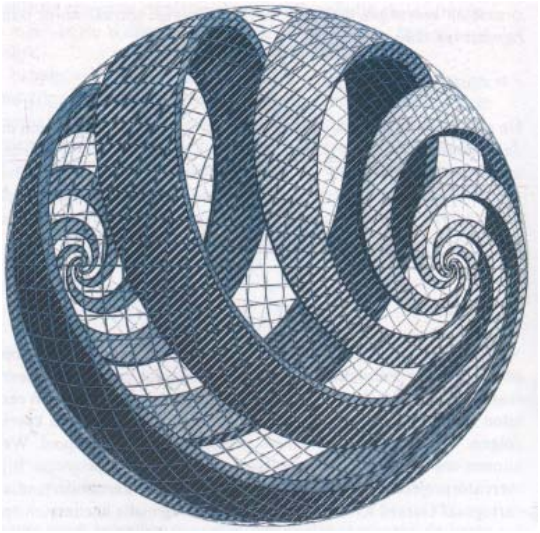


Sphere with growth spiral

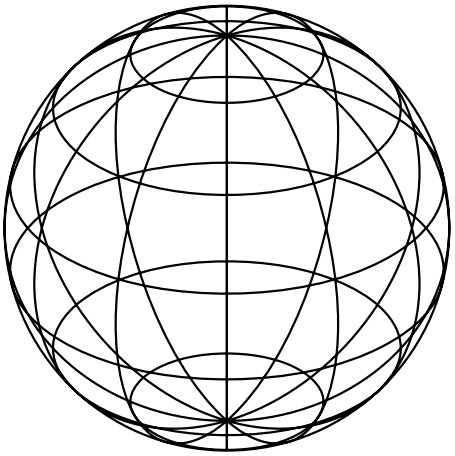
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Spheric Spiral, cgl nov 2012
%%BoundingBox: -102 -102 102 102
%%BeginSetup
%%EndSetup
%%BeginProlog
/bolspiradict 13 dict def
/bolspira{bolspiradict begin
%mods: scaling radius r, atan yields degrees
/r 100 def%scale
/a .15 def %spiral constant, the smaller the more windings
/p .7071 def  /c .9 def
/c1 c p mul def %p=1/sqrt 2
/c3 1 c c mul sub sqrt def
/q p  c3 mul def %p q projection numbers
/pi 3.14159 def
/r2d {180 pi div mul} def%conversion radials into degrees
/b 500 def%limit for the looping.
b neg 1 b{/n exch def
    /s n pi mul 25 div def /t a s mul 1 atan def
    /x s r2d cos t cos mul r mul def
    /y s r2d sin t cos mul r mul def
    /z t sin r mul neg def
    p y x sub mul  c z mul q x y add mul sub %u,v on stack
    /w c1 x y add mul c3 z mul add  def
    b neg n eq w 0 lt or{moveto}{lineto}ifelse
   }for
r 0 moveto 0 0 r 0 360 arc  %projection circle
end} bind def
%%Endprolog
bolspira stroke showpage
```



Escher's spirals on a sphere

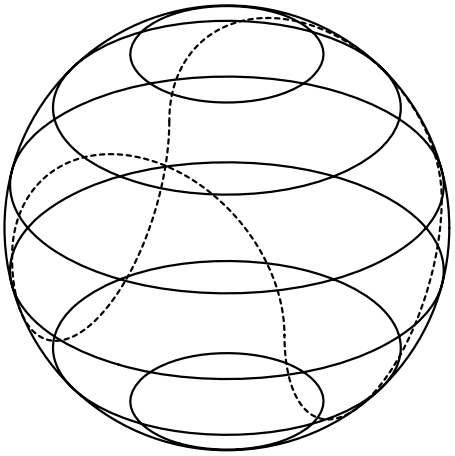


Sphere with meridians and latitude circles



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

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

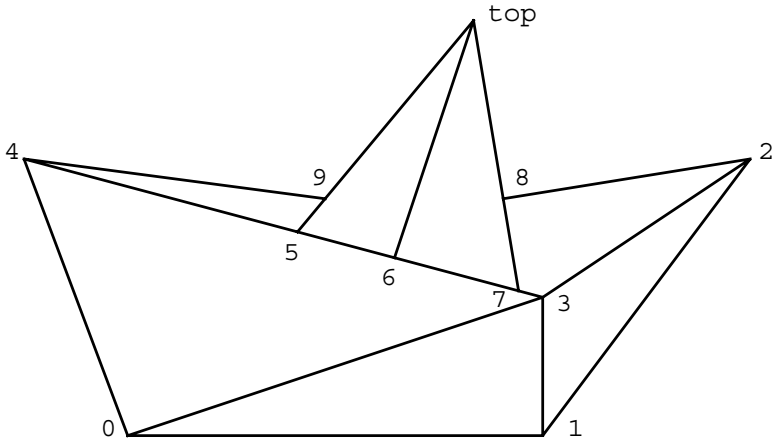


```
%tenniscurve
/tennisball{/r r 2 sqrt div def /-r r neg def
%Tennisball part {t:(1, sin t, cos t)}
r -r 0 ptp moveto
-89 1 90{/t exch def
 r  t sin r mul  t cos r mul  ptp lineto
}for
%Tennisball part {t:(-1, sin t, cos t)}
-r   -r   0  ptp moveto
-89 1 90{/t exch def
-r   t sin r mul  t cos r mul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

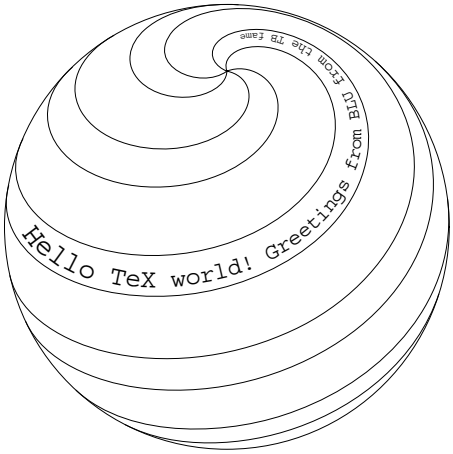
Gust battleship logo

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

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



Sphere with spiral and text



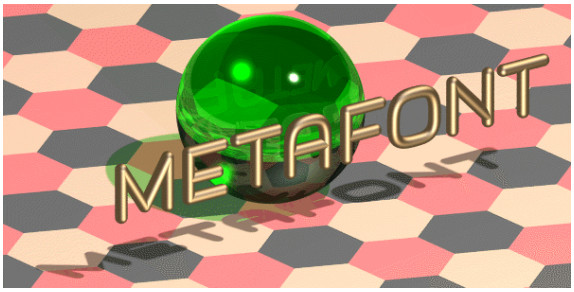

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Spere with belts and text, kisa1@xs4all.nl, 2012
%%BoundingBox: -101 -102 101 102
%%BeginSetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
/Courier 14 selectfont
pathtextdict begin /size 14 def
/linetoproc
  { /oldx newx def /oldy newy def
/newy exch def /newx exch def
/dx newx oldx sub def
/dy newy oldy sub def
/dist dx dup mul dy dup mul add sqrt def
dist 0 ne
  { /dsx dx dist div ovr mul def
    /dsy dy dist div ovr mul def
    oldx dsx add oldy dsy add transform
    /cpy exch def /cpx exch def
    /pathdist pathdist dist add def
{
/size size .08 sub def
/Courier findfont [size 0 0 size 0 0] makefont setfont
                setdist pathdist le
  { charcount str length lt
{setchar} {exit} ifelse }
  { /ovr setdist pathdist sub def exit }
  ifelse
} loop
      } if
  } def
end
/sphereandspiraldict 30 dict def
/sphereandspiral{sphereandspiraldict begin /r 100 def
/c .7 def /a2 1 1.41421 div def /b1 a2 1 c c mul sub sqrt mul def
/c1 c a2 mul def /c3 1 c c mul sub sqrt def
/spiral{-90 1 +90{/thetaj exch def
 /phij thetaj 4 mul phi0 add def%windings
 /x r phij cos thetaj cos mul mul def
 /y r phij sin thetaj cos mul mul def
 /z r thetaj sin mul def
 /w c1 x y add mul c3 z mul add def
 /u a2 y x sub mul def
 /v c z mul b1 x y add mul sub def
 w 0 lt{u v moveto}{u v lineto}ifelse
}for %thetaj
}def%spiral
gsave
0 c r mul moveto /phi0   0 def spiral
0 c r mul moveto /phi0  45 def spiral stroke
grestore
gsave
0 c r mul moveto /phi0 120 def spiral
0 c r mul moveto /phi0 165 def spiral stroke
grestore
gsave
0 c r mul moveto /phi0 240 def spiral stroke
0 c r mul moveto /phi0 285 def spiral stroke
0 c r mul moveto /phi0 275 def spiral
( Hello TeX world Greetings from BLU from the TB fame) 20 pathtext
grestore
%circle in projection plane
r 0 moveto
1 1 100{/k exch def  /t k 180 mul 50 div def
        r t cos mul r t sin mul lineto
}for %k
stroke end} bind def
```

Jacko's Toruń98 logo.

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

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

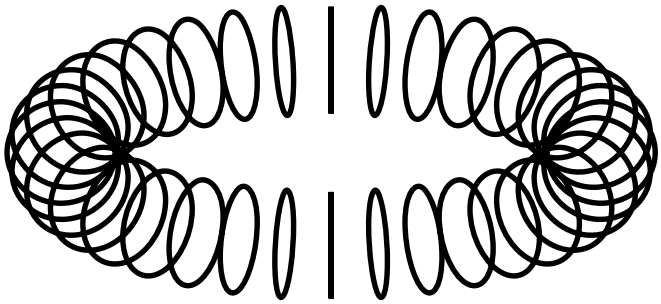
This has been shown in the 90-ies by Bogusław Jackowski and Marek Ryćko. Non-scalability is not relevant for pieces of art.



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

Toroid

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



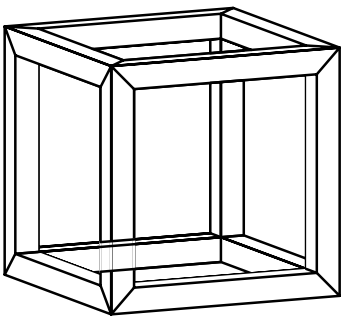
```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -61 -28 61 28
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\Pslib.eps) run
%
/toroiddict 16 dict def
toroiddict begin%locals
/R 50 def /r 10 def /phi 0 def /theta 20 def
/a1{R r sub t cos mul   R r sub t sin mul   0         ptp}def
/a2{R r sub t cos mul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

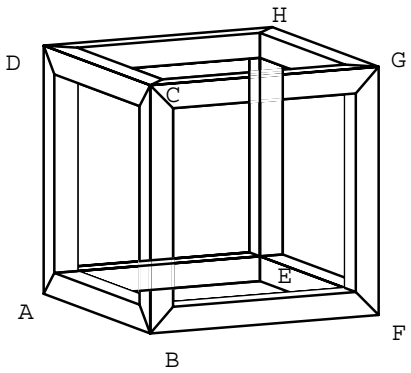
A direct translation of Lauwerier's BASIC code

```
%!PS-Adobe-3.0 EPSF=3.0  %Torus, cgl Jan 07
%%BoundingBox: -300 -100 300 300
%%Creator: H Lauwerier. Meetkunde met de microcomputer. Adapted to PS cgl Jan 2007
/PI 3.141593 def /C .8 def  /C1 .7071 1 C C mul sub sqrt mul def
/r1 235 def /r2 40 def
/ptp2{/z exch def/y exch def/x exch def %point to pair projection
   y x sub .7071 mul
   C z mul x y add C1 mul sub}def
/n 75 def 300 300 translate
0 1 n 1 sub{/k exch def
   /A k n div 360 mul cos def
   /B k n div 360 mul sin def
   0 10 360{/t exch def
            B neg r1 r2 t cos mul add mul %x stacked
            A     r1 r2 t cos mul add mul %y stacked
            r2 t sin  mul                 %z stacked
            ptp2                          %u,v stacked xyz consumed
            t 0 eq{moveto}{lineto}ifelse
   }for stroke %less memory requirement than stroke on the end
}for
showpage
```

Escher's impossible cubes

```
%!PS-Adobe-3.0  EPSF-3.0%PostScript-Adobe-3.0 EPSF-3.0
%%Title: Imp Cube in projection2. cgl 2009
%%BoundingBox: -20 -30 150 120
%%Beginsetup
%%EndSetup
(C:\\PSlib\\PSlib.eps) run
/ptp{/z exch def/y exch def/x exch def
   x neg a cos mul y a sin mul add
   x neg a sin mul b sin mul y neg a cos mul b sin mul add
     z b cos mul add}def
/a 25 def /b 10 def%projection parameters
/r 100 def /mr r neg def /hr r 2 div def%size
/d 10 def  /md d neg def  /rmd r d sub def  /dmr rmd neg def  %d<<r
1 setlinewidth 1 setlinejoin /Courier 10 selectfont
/a1 {0 0 0 ptp} def
/a2 {0 d 0 ptp} def
/a3 {0 d d ptp} def
/a4 {0 0 d ptp} def
/a5 {md 0 0 ptp} def
/a6 {md d 0 ptp} def
/a7 {md d d ptp} def
/a8 {md 0 d ptp} def
%a1 moveto -10 -10 rmoveto (A) show
/b1 {0  rmd  0 ptp} def
/b2 {0  r    0 ptp} def
/b3 {0  r    d ptp} def
/b4 {0  rmd  d ptp} def
/b5 {md rmd  0 ptp} def
/b6 {md r    0 ptp} def
/b7 {md r    d ptp} def
/b8 {md rmd  d ptp} def
%b1 moveto 10 -15 rmoveto (B) show
/c1 {0  rmd  rmd ptp} def
/c2 {0  r    rmd ptp} def
/c3 {0  r    r   ptp} def
/c4 {0  rmd  r   ptp} def
/c5 {md rmd  rmd ptp} def
/c6 {md r    rmd ptp} def
/c7 {md r    r   ptp} def
/c8 {md rmd  r   ptp} def
%c2 moveto 6 3 rmoveto (C) show
/d1 {0  0 rmd ptp} def
/d2 {0  d rmd ptp} def
/d3 {0  d r   ptp} def
/d4 {0  0 r   ptp} def
/d5 {md 0 rmd ptp} def
/d6 {md d rmd ptp} def
/d7 {md d r   ptp} def
/d8 {md 0 r   ptp} def
%d1 moveto -15 0 rmoveto (D) show
/e1 {dmr 0 0 ptp} def
/e2 {dmr d 0 ptp} def
/e3 {dmr d d ptp} def
/e4 {dmr 0 d ptp} def
/e5 {mr 0 0 ptp} def
/e6 {mr d 0 ptp} def
/e7 {mr d d ptp} def
/e8 {mr 0 d ptp} def
%e5 moveto 2 -3 rmoveto (E) show
/f1 {dmr rmd  0 ptp} def
/f2 {dmr r    0 ptp} def
/f3 {dmr r    d ptp} def
/f4 {dmr rmd  d ptp} def
/f5 {mr  rmd  0 ptp} def
/f6 {mr  r    0 ptp} def
/f7 {mr  r    d ptp} def
```

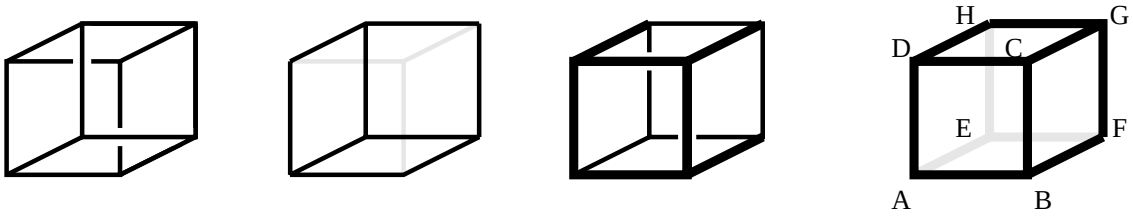




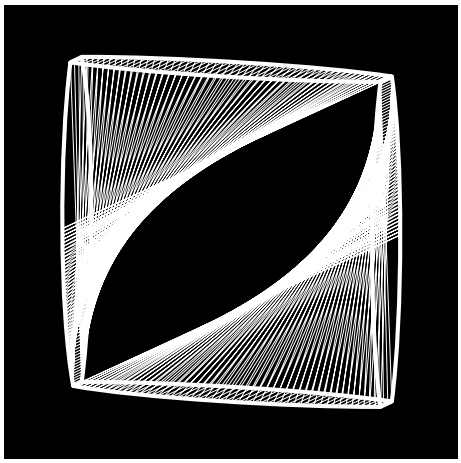
```
/f8 {mr  rmd  d ptp} def
%f6 moveto 5 -10 rmoveto (F) show
/g1 {dmr rmd  rmd ptp} def
/g2 {dmr r    rmd ptp} def
/g3 {dmr r    r   ptp} def
/g4 {dmr rmd  r   ptp} def
/g5 {mr  rmd  rmd ptp} def
/g6 {mr  r    rmd ptp} def
/g7 {mr  r    r   ptp} def
/g8 {mr  rmd  r   ptp} def
%g7 moveto 5  0 rmoveto (G) show
/h1 {dmr 0 rmd ptp} def
/h2 {dmr d rmd ptp} def
/h3 {dmr d r   ptp} def
/h4 {dmr 0 r   ptp} def
/h5 {mr 0 rmd ptp} def
/h6 {mr d rmd ptp} def
/h7 {mr d r   ptp} def
/h8 {mr 0 r   ptp} def
%h8 moveto 0 2 rmoveto (H) show
%AEHD
/AEHDo {a1 moveto e5 lineto h8 lineto d4 lineto closepath}def
/AEHDoi{a8 moveto e4 lineto h1 lineto d5 lineto closepath}def
/AEHDii{a7 moveto e3 lineto h2 lineto d6 lineto closepath}def
%EFGH
/EFGHo {e5 moveto f6 lineto g7 lineto h8 lineto closepath}def
/EFGHoi{e7 moveto f8 lineto g1 lineto h6 lineto closepath}def
/EFGHii{e3 moveto f4 lineto g1 lineto h2 lineto closepath}def
%ABCD
/ABCDo{a1 moveto b2 lineto c3 lineto d4 lineto closepath}def
/ABCDoi{a3 moveto b4 lineto c1 lineto d2 lineto closepath}def
%BFGC
/BFGCo{b2 moveto f6 lineto g7 lineto c3 lineto closepath}def
/BFGCoi{b7 moveto f3 lineto g2 lineto c6 lineto closepath}def
%CGHD
/CGHDii{c3 moveto g7 lineto h8 lineto d4 lineto closepath}def
/CGHDoi{c8 moveto g4 lineto h3 lineto d7 lineto closepath}def
/CGHDo {c3 moveto g7 lineto h8 lineto d4 lineto closepath}def
%ABFE
/ABFEOi{a6 moveto b5 lineto f1 lineto e2 lineto closepath}def
/ABFEii{a7 moveto b8 lineto f4 lineto e3 lineto closepath}def
%Snijpunten ivm clipping window
/uc1{h2 d6 d7 c8 intersect}def
/uc2{h2 g1 c8 g4 intersect}def
/uc3{c6 g2 g1 f4 intersect}def
/uc4{f4 b8 b7 c6 intersect}def
/uc5{d6 a7 d2 c1 intersect}def
/uc6{a7 b8 b4 c1 intersect}def
%3 windows as clipping path
/windows{newpath %
a7 moveto uc6 lineto c1 lineto uc5 lineto closepath %ABCDoi
c6 moveto uc3 lineto f4 lineto uc4 lineto closepath %BFGCoi
c8 moveto uc1 lineto h2 lineto uc2 lineto closepath %CGHDoi
}def
%Draw Front and top
ABCDo ABCDoi BFGCo BFGCoi CGHDo CGHDoi stroke
%upper what is in sight
h8 moveto h3 lineto h2 lineto
d2 moveto d4 lineto d7 lineto
g4 moveto g7 lineto g2 lineto
c1 moveto c3 lineto c8 lineto
c3 moveto c6 lineto
h2 moveto uc1 lineto
h2 moveto uc2 lineto stroke
% einde achterbovenkant in zicht, lower corners
a1 moveto a3 lineto a7 lineto
b4 moveto b2 lineto b7 lineto
e4 moveto e3 lineto e2 lineto
e3 moveto e7 lineto
f4 moveto f3 lineto f6 lineto stroke
```

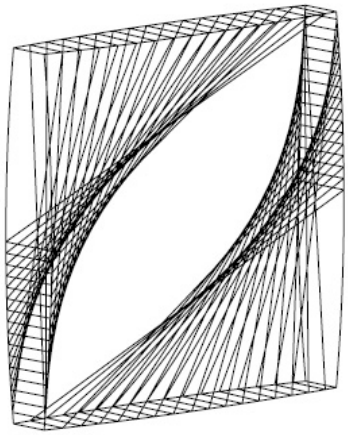
```
gsave windows clip AEHDii AEHDoi EFGHii ABFEii stroke
a8 moveto e4 lineto stroke
EFGHoi
a6 moveto e2 lineto f1 lineto b5 lineto closepath stroke
grestore
%Maak IMPCUBE
/uc10{c8 g8 e1 h1 intersect}def
/uc11{c3 g3 e1 h1 intersect}def
/uc12{c2 g2 e1 h1 intersect}def
%
/uc13{c8 g8 e7 h7 intersect}def
/uc14{c3 g3 e7 h7 intersect}def
/uc15{c2 g2 e7 h7 intersect}def
%
newpath
uc10 moveto uc13 lineto
uc11 moveto uc14 lineto
uc12 moveto uc15 lineto stroke
%Impossible at the bottom
/uc20{a4 e4 b1 c1 intersect}def
/uc21{a4 e4 b2 c2 intersect}def
/uc22{a4 e4 b6 c6 intersect}def
/uc23{a2 e2 b1 c1 intersect}def
/uc24{a2 e2 b2 c2 intersect}def
/uc25{a2 e2 b6 c6 intersect}def
/uc26{a3 e3 b1 c1 intersect}def
/uc27{a3 e3 b5 c5 intersect}def

uc20 moveto uc23 lineto
uc21 moveto uc24 lineto
uc22 moveto uc25 lineto
1 setgray  stroke %white
uc20 moveto e4 lineto
a3   moveto e7 lineto
uc23 moveto e2 lineto
0 setgray stroke
showpage
%%EOF
```

From the Metafont book ex13.7



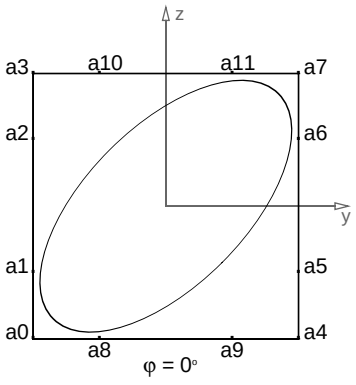
Emulation Gabo's linear construction no1

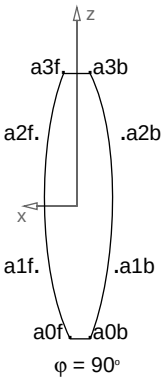




```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox: -61 -28 61 28
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\Pslib.eps) run
/linearconstructionno1dict 50 dict def

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

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

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

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

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

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

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

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

/sampleellipsestroke{gsave
-45 rotate
.6 1.2 scale%kind of ellipse
1 1 360{/t exch def
2 s t cos mul 2 s t sin mul }for %sample the skewed ellipse
count /n exch def
0 3 1 roll  % 0 y z
ptp         % u v projected point
moveto
n 2 div 1 sub cvi {0 3 1 roll  % 0 y z
             ptp         % u v projected point
             lineto
            }repeat closepath
stroke grestore} def
%
/approxellipsestroke{gsave
-45 rotate
.6 1.2 scale%kind of ellips
/a0 {0 -2 s    0     ptp} def
/a1 {0 -2 s    1.1 s ptp} def
/a2 {0 -1.1 s  2 s   ptp} def
/a3 {0  0      2 s   ptp} def
```

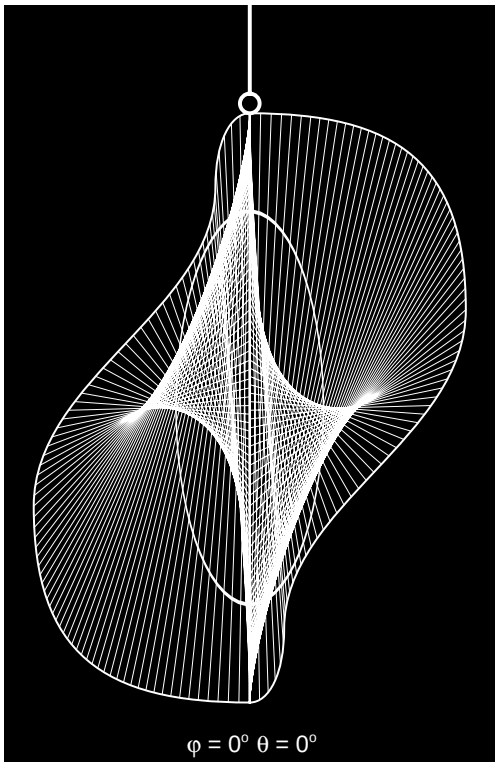
```
/a4 {0  1.1 s  2 s   ptp} def
/a5 {0  2 s    1.1 s ptp} def
/a6 {0  2 s    0     ptp} def
/a7 {0  2 s   -1.1 s ptp} def
/a8 {0  1.1 s -2 s   ptp} def
/a9 {0  0     -2 s   ptp} def
/a10{0 -1.1 s -2 s   ptp} def
/a11{0 -2 s   -1.1 s ptp} def
a0 moveto a1  a2  a3 curveto
          a4  a5  a6 curveto
          a7  a8  a9 curveto
          a10 a11 a0 curveto stroke
grestore} def

/frame{
a0f moveto  a1f  a2f  a3f curveto
            a10f a11f a7f curveto
            a6f  a5f  a4f curveto
            a9f  a8f  a0f curveto

a0b moveto  a1b  a2b  a3b curveto
           a10b a11b a7b curveto
            a6b  a5b  a4b curveto
            a9b  a8b  a0b curveto
a0f moveto a0b lineto
a3f moveto a3b lineto
a4f moveto a4b lineto
a7f moveto a7b lineto
} def

/dostringing{
0.02 .02 .5001{/t exch def
t       a0f a1f  a2f  a3f tOnSpline moveto
2 t mul a3b a10b a11b a7b tOnSpline lineto
2 t mul a3f a10f a11f a7f tOnSpline lineto
t       a0b a1b  a2b  a3b tOnSpline lineto
closepath
t       a7f a6f  a5f  a4f tOnSpline moveto
2 t mul a4b a9b  a8b  a0b tOnSpline lineto
2 t mul a4f a9b  a8f  a0f tOnSpline lineto
t       a7b a6b  a5b  a4b tOnSpline lineto
closepath
}for } bind def

.1 setlinewidth stroke

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

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

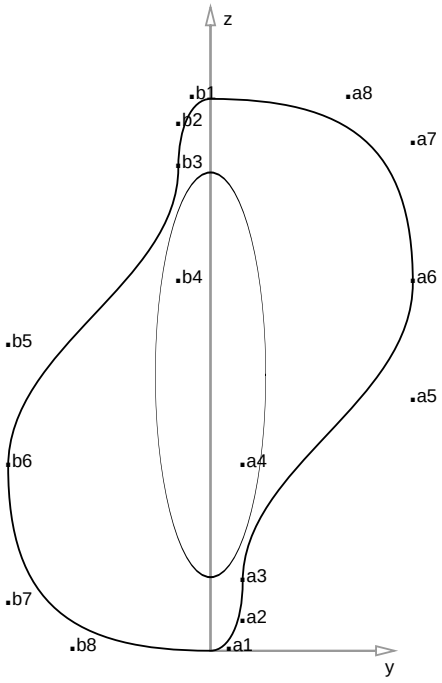




Emulation Gabo's linear construction no2

```
%PostScript-Adobe-3.0 EPSF-3.0
%%Title: Lin Constr in Space n0 2 Frame
%%BoundingBox: -125 -30 125 355
%%BeginSetup
%%EndSetup
(C:\\PSlib\\Pslib.eps) run
/linearconstructionno2dict 50 dict def
linearconstructionno2dict  begin
/reversevideo{0 setgray%Black, Mimics BoundingBox
-125 -30 250 385 rectfill } def
/rope{0 355 moveto
      0 0 3 s ptp 5 add  5 90 450 arc%ring
}def
%yz plane
/a0 {0 0        0       ptp} def
/a1 {0  .1   s  0       ptp} def
/a2 {0  .175 s   .15  s ptp} def
/a3 {0  .175 s   .375 s ptp} def
/a4 {0  .175 s  1     s ptp} def
/a5 {0 1.1   s  1.35  s ptp} def
/a6 {0 1.1   s  2     s ptp} def
/a7 {0 1.1   s  2.75  s ptp} def
/a8 {0  .75  s  3     s ptp} def
/a9 {0  0       3     s ptp} def
%Sampling of half frame, with phi=0 and theta=0, ie in +yz plane
/sampleframe{/phi 0 def /theta 0 def
0 .175  .95{ a0 a1 a2 a3 tOnSpline }for %10 pairs on stack
0.02 .028 .99{ a3 a4 a5 a6 tOnSpline }for %35 pairs on stack
0  .025 1  { a6 a7 a8 a9 tOnSpline }for %41 pairs on stack
}def
%rotated =xz plane
/a0r { 0       0 0       ptp} def
/a1r {  .1   s 0 0       ptp} def
/a2r {  .175 s 0  .15  s ptp} def
/a3r {  .175 s 0  .375 s ptp} def
/a4r {  .175 s 0 1     s ptp} def
/a5r { 1.1   s 0 1.35  s ptp} def
/a6r { 1.1   s 0 2     s ptp} def
/a7r { 1.1   s 0 2.75  s ptp} def
/a8r {  .75  s 0 3     s ptp} def
/a9r { 0       0 3     s ptp} def
%mirrored and upside down, -yz plane
/b0 {0  0       3     s ptp} def
/b1 {0  -.1   s 3     s ptp} def
/b2 {0  -.175 s 2.85  s ptp} def
/b3 {0  -.175 s 2.625 s ptp} def
/b4 {0  -.175 s 2     s ptp} def
/b5 {0 -1.1   s 1.65  s ptp} def
/b6 {0 -1.1   s 1     s ptp} def
/b7 {0 -1.1   s  .25  s ptp} def
/b8 {0 -.75   s 0       ptp} def
/b9 {0   0      0       ptp} def
%rotated -xz plane
/b0r { 0       0 3     s ptp} def
/b1r { -.1   s 0 3     s ptp} def
/b2r { -.175 s 0 2.85  s ptp} def
/b3r { -.175 s 0 2.625 s ptp} def
/b4r { -.175 s 0 2     s ptp} def
/b5r {-1.1   s 0 1.65  s ptp} def
/b6r {-1.1   s 0 1     s ptp} def
/b7r {-1.1   s 0  .25  s ptp} def
/b8r { -.75  s 0 0       ptp} def
/b9r { 0       0 0       ptp} def
```





```
/frame{%under projection angles
a0 moveto a1 a2 a3 curveto a4 a5 a6 curveto a7 a8 a9 curveto
a0r moveto  a1r a2r a3r curveto a4r a5r a6r curveto a7r a8r a9r curveto
b0 moveto b1 b2 b3 curveto b4 b5 b6 curveto b7 b8 b9 curveto
b0r moveto b1r b2r b3r curveto b4r b5r b6r curveto b7r b8r b9r curveto closepath
}def
%
/dostringing
{count %number of samples x and y on the stack
/n exch def
n copy %copy the stack
n array astore /samples exch def % store the samples in the array
n copy %for stringing 2 quadrants
%Stringing +yz<->-xz
%other quadrants
%+xZ<->+yz
/np1 n 1 add def%n-1 last element of samples array
/cnt 0       def%first element of samples array
n 2 div cvi {
/cnt cnt 2 add def
% in +yZ plane
0                        %x value
samples n   cnt sub get %y value
samples np1 cnt sub get %z value
ptp moveto
% rotate to +xz plane
samples cnt 2 sub get %y value
0
samples cnt 1 sub get %z value
ptp lineto
}repeat
%-xz<->-yz
/np1 n 1 add def%n-1 last element of samples array
/cnt 0       def%first element of samples array
n 2 div cvi {
/cnt cnt 2 add def
% in +yZ plane
0                           %x value
samples n   cnt sub get neg %y value
samples np1 cnt sub get 3 s sub neg %mirrored z value
ptp moveto
% rotate to +xz plane
samples cnt 2 sub get neg %y value
0
samples cnt 1 sub get 3 s sub neg %z value
ptp lineto
}repeat
} bind def
%
/yz-ellipsestroke{gsave%implicit in yz plane, viewing angles must be available
0 1.5 s translate
.3  1.1 scale%kind of ellipse
1 1 360{/t exch def
1 s t cos mul 1 s t sin mul }for %sample the skewed ellipse in +yz plane
%count /n exch def
0 3 1 roll  % 0 y z
ptp         % u v projected point
moveto
359{0 3 1 roll % 0 y z
   ptp         % u v projected point
   lineto
   }repeat stroke grestore} def
%
/xz-ellipsestroke{gsave%implicit in yz plane, viewing angles must be available
0 1.5 s translate
.3  1.1 scale%kind of ellipse
1 1 360{/t exch def
1 s t cos mul 1 s t sin mul }for %sample the skewed ellipse in +yz plane
%
0 2 1 roll  % 0 y z
ptp         % u v projected point
```

```
moveto
359{0 2 1 roll % 0 y z
   ptp         % u v projected point
   lineto
   }repeat stroke grestore} def
%
/yz-approxellipseNo2stroke{gsave%implicit in yz plane, viewing angles must be available
0 1.5 s translate
.3  1.1 scale%kind of ellipse
/a0 {0 -1   s  0     ptp} def
/a1 {0 -1   s  .55 s ptp} def
/a2 {0 -.55 s  1   s ptp} def
/a3 {0  0      1   s ptp} def
/a4 {0  .55 s  1   s ptp} def
/a5 {0  1   s  .55 s ptp} def
/a6 {0  1   s  0     ptp} def
/a7 {0  1   s -.55 s ptp} def
/a8 {0  .55 s -1   s ptp} def
/a9 {0  0     -1   s ptp} def
/a10{0 -.55 s -1   s ptp} def
/a11{0 -1   s -.5  s ptp} def
a0 moveto a1  a2  a3 curveto
          a4  a5  a6 curveto
          a7  a8  a9 curveto
          a10 a11 a0 curveto stroke
grestore} def  %end yz-approxellipseNo2stroke
%
/xz-approxellipseNo2stroke{gsave%implicit in xz plane, viewing angles must be available
0 1.5 s translate
.3  1.1 scale%kind of ellipse
/a0 {-1   s 0  0     ptp} def
/a1 {-1   s 0  .55 s ptp} def
/a2 {-.55 s 0  1   s ptp} def
/a3 { 0     0  1   s ptp} def
/a4 { .55 s 0  1   s ptp} def
/a5 { 1   s 0  .55 s ptp} def
/a6 { 1   s 0  0     ptp} def
/a7 { 1   s 0 -.55 s ptp} def
/a8 { .55 s 0 -1   s ptp} def
/a9 { 0     0 -1   s ptp} def
/a10{-.55 s 0 -1   s ptp} def
/a11{-1   s 0 -.55 s ptp} def
a0 moveto a1  a2  a3 curveto
          a4  a5  a6 curveto
          a7  a8  a9 curveto
          a10 a11 a0 curveto stroke
grestore} def  %end xz-approxellipseNo2stroke

/annotations{-32 -25  moveto
 S12pt setfont (j) show ( = ) H12pt setfont show phi (  ) cvs show
   gsave 0 5 rmoveto (o) H7pt setfont show grestore
8 0 rmoveto
 S12pt setfont (q) show ( = ) H12pt setfont show theta (  ) cvs show
        0 5 rmoveto (o) H7pt setfont show} def
end

/linearconstructionno2
{linearconstructionno2dict begin
/scalingfactor 100 def
reversevideo 1 setgray %white lines further
sampleframe %with phi=theta=0 /phi 30 def /theta 10 def% viewing angle
.5 setlinewidth dostringing stroke
3  setlinewidth frame       stroke
2  setlinewidth yz-approxellipseNo2stroke
                xz-approxellipseNo2stroke
                rope        stroke
annotations
end}def
linearconstructionno2
%%EOF
```

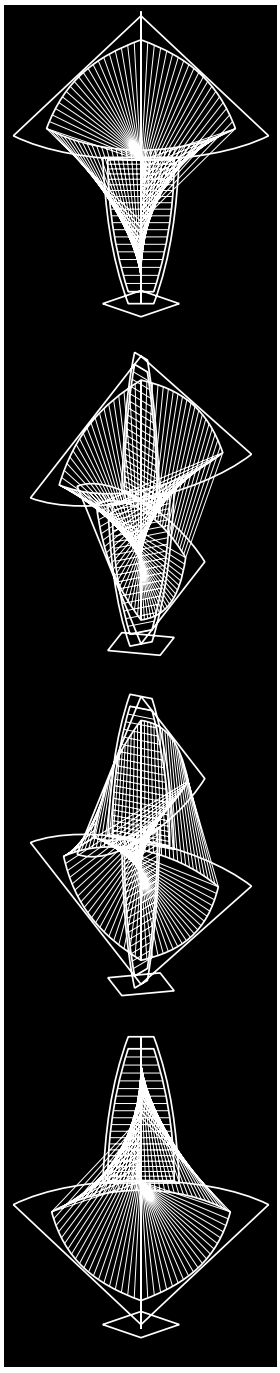
My first version op Linear Construction no I, I did in Metafont. The picture has been incorporated in LaTeX's Graphics Companion.

```
%Linear Construction no2, Naum Gabo. CGL, 1996
%tracingmacros:=1;
 tracingstats:=1; tracingtitles:=1; proofing:=1;screenstrokes;
 xsize=80; ysize=.125xsize; zsize=xsize;
 def pointtopair(expr x,y,z)=
 %The projection of a point into a pair in the projected plane.
 %x,y,z coordinates of a point
 %For transformation coeff. see Lauwerier p.26, 47 e.v.
     (-.6x+.8y,-4/13x-3/13y+12/13z) enddef;
def pointtopair(expr x,y,z)=
(-x*cosd aa + y*sind aa, -x*sind aa * sind bb -y*cosd aa * sind bb + z*cosd bb) enddef;
def openit = openwindow currentwindow    from origin to (screen_rows,screen_cols) at (-1.5xsize,300)enddef;
path p[];  picture blackpicture, storedpicture; storedpicture:=currentpicture;
pickup pencircle scaled .0025pt;
for b=-10: bb:=b;
    for a= -10step-15until-45:%Varying viewing angles
          aa:=a; currentpicture:=storedpicture shifted(1.5xsize,0);
          p1:= pointtopair(0,0,0)..pointtopair(.5xsize,-.25ysize,0)..pointtopair(xsize,0,0);
          p2:= pointtopair(xsize,0,0)--pointtopair(xsize,ysize,0);
          p3:= pointtopair(xsize,ysize,0)..pointtopair(.5xsize,1.25ysize,0)..pointtopair(0,ysize,0);
          p4:= pointtopair(0,ysize,0)--pointtopair(0,0,0); draw p1..p2..p3..p4;
          p5:= pointtopair(0,0,zsize)..pointtopair(.5xsize,-.25ysize,zsize)..pointtopair(xsize,0,zsize);
          p6:= pointtopair(xsize,0,zsize)--pointtopair(xsize,ysize,zsize);
          p7:= pointtopair(xsize,ysize,zsize)..pointtopair(.5xsize,1.25ysize,zsize)..pointtopair(0,ysize,zsize);
          p8:= pointtopair(0,ysize,zsize)--pointtopair(0,0,zsize); draw p5..p6..p7..p8;
          p9:= pointtopair(0,0,0)..pointtopair(0,-.25ysize,.5zsize)..pointtopair(0,0,zsize);
          p10:= pointtopair(xsize,0,0)..pointtopair(xsize,-.25ysize,.5zsize)..pointtopair(xsize,0,zsize);
          p11:= pointtopair(xsize,ysize,0)..pointtopair(xsize,1.25ysize,.5zsize)..pointtopair(xsize,ysize,zsize);
          p12:= pointtopair(0,ysize,0)..pointtopair(0,1.25ysize,.5zsize)..pointtopair(0,ysize,zsize);
          draw p9; draw p10; draw p11; draw p12; n:=xsize/6;
          for k:= 1 upto n: draw point k/n of p11--point 2-2k/n of p5--point 2k/n of p7--point k/n of p10--cycle;
                                  draw point2k/n of p3--point 1+k/n of p9--point 1+k/n of p12--point 2-2k/n of p1--cycle;
           endfor showit; storedpicture:=currentpicture;
%Undo screenstrokes, MB 277, to speed up rotated variants
def addto_currentpicture text t= addto currentpicture t enddef;
endfor
endfor; cullit; blackpicture:= currentpicture; clearit;
fill (-1.25xsize,-.25zsize)--(-1.25xsize,1.25zsize)--(3.25xsize,1.25zsize)--(3.25xsize,-.25zsize)      --cycle;
currentpicture:=currentpicture - blackpicture; showit; clearit; end
```

Emulation Gabo's Torsion

The construction, a stringed metal frame, consists essentially of 2 identical rectangular isosceles triangles, with a curved hypotenuse and curved inner sides, on top of each other, with the upper triangle turned upside down and rotated over 90°. The inner curves of the lower triangle are connected by lines to the inner curves of the upper triangle, the strings of the object. Cap and cup have horizontal strings. The stringing yields what I call stringed surfaces. The spline data on the triangles of the object are important and shown in the accompanying figures.

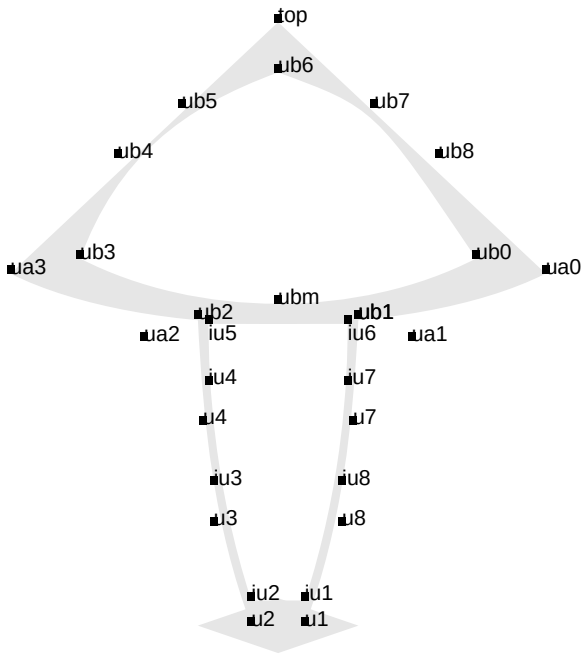
```
%PostScript-Adobe-3.0 EPSF-3.0
%%Title: Gabo Torsion.   2011 CGL, kisa1@xs4all.nl
%%BoundingBox: -80 -25 80 775
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %contains ptp and tOnSpline and ... Torsion
%%EndProlog
-80 -25 160 825 rectfill %reverse video. filling with black
1 setgray%white lines
/theta 20 def /scalingfactor 3 def%fixed declination and scaling

80 0  moveto    /phi  0 def gabotorsion
0 200 translate /phi 30 def gabotorsion
0 200 translate /phi 60 def gabotorsion
0 200 translate /phi 90 def gabotorsion
showpage
```



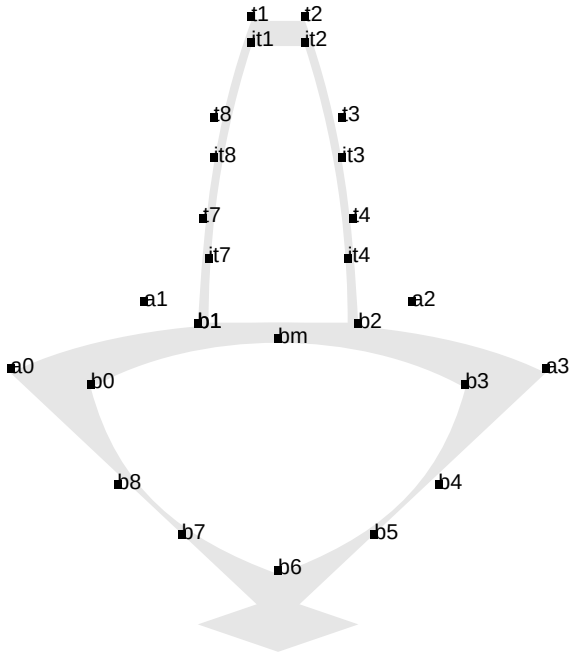
The library contains

```
/gabotorsiondict 70 dict def
gabotorsiondict begin
/s {scalingfactor mul} def
%Specification of data in 3D and projections in 2D, dependent on phi and theta
%lowertorsiontriangle data
/a0 { 0 -25   s  25     s ptp } def %outer points
/a1 { 0 -12.5 s  31.67  s ptp } def
/a2 { 0  12.5 s  31.67  s ptp } def
/a3 { 0  25   s  25     s ptp } def
/origin { 0 0 } def
/b0 { 0 -17.5 s  23.5   s ptp } def%inner points
/b1 { 0  -7.5 s  29.5   s ptp } def
/bm { 0   0      28     s ptp } def
/b2 { 0   7.5 s  29.5   s ptp } def
/b3 { 0  17.5 s  23.5   s ptp } def
/b4  {0  15   s  13.5   s ptp } def
/b5 { 0   9   s   8.5   s ptp } def
/b6 { 0   0       5     s ptp } def
/b7 { 0  -9   s   8.5   s ptp } def
/b8 { 0 -15   s  13.5   s ptp } def
%uppertorsiontriangle, 90 rotated, upside down data
/ua0 { -25   s 0  35    s   ptp } def%outer points
/ua1 { -12.5 s 0  28.34 s ptp } def
/ua2 {  12.5 s 0  28.34 s ptp } def
/ua3 {  25   s 0  35    s ptp } def
/top { 0 0 60 s ptp } def
/ub0 { -18.5 s 0  36.5 s  ptp } def%inner points of uframe
/ub1 {  -7.5 s 0  30.5 s  ptp } def
/ub2 {   7.5 s 0  30.5 s  ptp } def
/ub3 {  18.5 s 0  36.5 s  ptp } def
/ub4 {  15   s 0  46.5 s  ptp } def
/ub5 {   9   s 0  51.5 s  ptp } def
/ub6 {   0     0  55   s  ptp } def
/ub7 {  -9   s 0  51.5 s  ptp } def
/ub8 { -15   s 0  46.5 s  ptp } def
/ubm {   0     0  32   s  ptp} def
```

```
/t1 { 0 -2.5 s 60 s ptp } def
/t2 { 0  2.5 s 60 s ptp } def
/t3 { 0  6   s 50 s ptp } def
/t4 { 0  7   s 40 s ptp } def
/t5 { b2 } def
/t6 { b1 } def
/t7 { 0 -7   s 40 s ptp } def
/t8 { 0 -6   s 50 s ptp } def
%cap inner
/it1 { 0 -2.5 s 57.5 s ptp } def
/it2 { 0  2.5 s 57.5 s ptp } def
/it3 { 0  6   s 46   s ptp } def
/it4 { 0  6.5 s 36   s ptp } def
/it5 { 0  6.5 s 30   s ptp } def
/it6 { 0 -6.5 s 30   s ptp } def
/it7 { 0 -6.5 s 36   s ptp } def
/it8 { 0 -6   s 46   s ptp } def
%cup outer; mirrored vertically around z=30 and rotated
/u1 { -2.5 s 0   0   ptp } def
/u2 {  2.5 s 0   0   ptp } def
/u3 {  6   s 0  10 s ptp } def
/u4 {  7   s 0  20 s ptp } def
/u5 { ub2 } def
/u6 { ub1 } def
/u7 { -7 s  0  20 s ptp } def
/u8 { -6 s  0  10 s ptp } def
%cup inner
/iu1 { -2.5 s 0  2.5 s ptp } def
/iu2 {  2.5 s 0  2.5 s ptp } def
/iu3 {  6   s 0 14   s ptp } def
/iu4 {  6.5 s 0 24   s ptp } def
/iu5 {  6.5 s 0 30   s ptp } def
/iu6 { -6.5 s 0 30   s ptp } def
/iu7 { -6.5 s 0 24   s ptp } def
/iu8 { -6   s 0 14   s ptp } def
%frame, formally in x- and y-components of the points
/lowertorsiontriangle{origin moveto %outer path
       a0 lineto
       a1 a2 a3 curveto closepath
       b6  moveto                                %inner path
       b7 b8 b0 curveto
       b1 b2 b3  curveto
       b4 b5 b6 curveto closepath} def
/uppertorsiontriangle{top moveto %outer path
        ua0 lineto
        ua1 ua2 ua3 curveto closepath
        ub6 moveto                          %inner path
        ub7 ub8 ub0 curveto
        ub1 ub2 ub3 curveto
        ub4 ub5 ub6 curveto closepath} def
/foot{-7.5 s     0   0 ptp moveto
         0    -7.5 s 0 ptp lineto
       7.5 s     0   0 ptp lineto
         0     7.5 s 0 ptp lineto closepath} def
/cap{ t1 moveto t2 lineto %outer
        t3 t4 b2 curveto
        b1 lineto
       t7 t8 t1 curveto closepath
      it1 moveto it2 lineto %inner
      it3 it4 it5 curveto
      it6 lineto
      it7 it8 it1 curveto closepath} def
/cup{ u1 moveto u2 lineto %outer
      u3 u4 u5 curveto
      u6 lineto
      u7 u8 u1 curveto closepath
     iu1 moveto iu2 lineto %inner
      iu3 iu4 iu5 curveto
      iu6 lineto
      iu7 iu8 iu1 curveto closepath} def
```

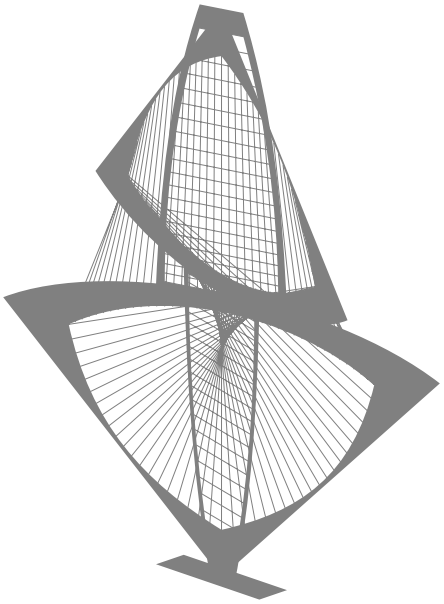


φ = 90°



φ = 0°

```
%shorthand to draw the chords
/wiring{%
0 0.05 1.01{/t exch def
  t b0  b8  b7  b6  tOnSpline moveto
  t 2 div ub0 ub1 ub2 ub3 tOnSpline lineto
%
  t 2 div b0  b1  b2  b3  tOnSpline moveto
  t ub3 ub4 ub5 ub6 tOnSpline lineto
%
  t b3  b4  b5  b6  tOnSpline moveto
  t 2 div ub3 ub2 ub1 ub0 tOnSpline lineto
%
  t 2 div .5 add b0  b1  b2  b3  tOnSpline moveto
  t ub6 ub7 ub8 ub0 tOnSpline lineto
%cap
  t it1  it8  it7   it6  tOnSpline moveto
  t it2  it3  it4   it5  tOnSpline lineto
%cup
  t iu1  iu8  iu7   iu6  tOnSpline moveto
  t iu2  iu3  iu4   iu5  tOnSpline lineto
}for
}def
end

/gabotorsion{gabotorsiondict begin
gsave 0.1 setlinewidth wiring stroke grestore
foot   stroke
lowertorsiontriangle  stroke
uppertorsiontriangle stroke
cap cup stroke
end}def
```

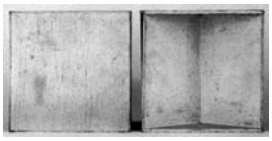


$\varphi = 30^{o}\ \theta = 20^{o}$

Emulation Gabo's Spheric Theme

Spheric Themes demonstrate another way of how Gabo constructed (the idea of) surfaces: by planes orthogonal to the surface. He called it the stereometric method of representing surfaces.

Stereometry is a branch of mathematics concerned with the description of 3D objects and calculations related to these. For Gabo a cube is not represented by its 6 surfaces, but by top, bottom, and the intersecting diagonal planes, making the inside visible



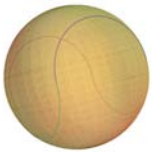
Famous are his constructed head series, such as the earlier shown `Head in a Corner Niche`, and the `Head ...` on the book cover. In the Spheric Theme we are about to emulate both ways of representing surfaces: stereometric and stringed surfaces are integrated in one object.

Gabo invented how to model his Spheric Theme curves:

> 'The basic form was made by taking two identical flat pierced circles or broad rings and making a single cut in each along the line of a radius. The two discs are bent in a serpentine curve and butt-jointed to each other at both ends. The resulting figure fits exactly into a sphere and the outer edges of the discs form the interlocking curves like those that divide the pieces of felt covering the tennis ball.'

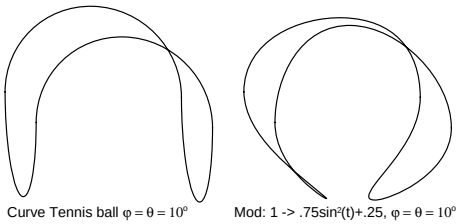
Frame of Spheric Theme

The outer circumference of the frame is the curve on the tennis ball, a little deformed. The four parts of the tennis ball curve are given by the formulas



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

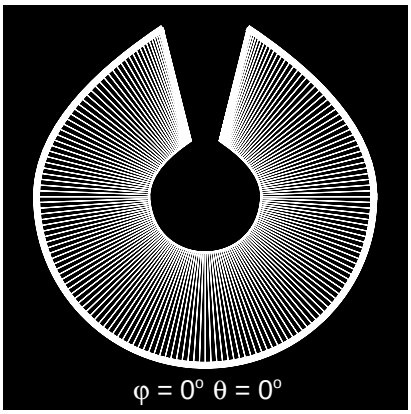
$$t\in(-\frac{\pi}{2},\frac{\pi}{2})$$

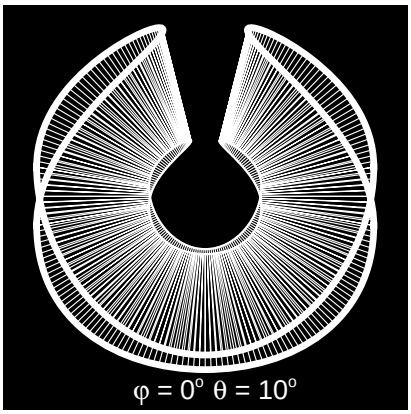


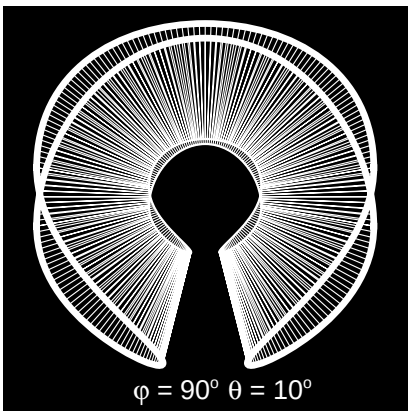
I modified the tennis ball curve by the deformation $1 \to (1-d)\sin^2 t + d$, $d = .25$, which comes close to Gabo's. For the inner circumference I took the outer scaled by a factor 3.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Emulation of Naum Gabo Spheric Theme. cgl, 2011, 2014 kisa1@xs4all,nl.
%%BoundingBox: -90 -95 90 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %used: s ptp
%
/sphericthemedict 75 dict def

sphericthemedict begin
/reversevideo{-90 -95 180 185 rectfill}def %Mimics the BoundingBox
```

```
/sphericpart{%part upper front x>0 {t:(ft, sin t, -cos t)}, f(-90)=1
1 s   -1 s    0 ptp moveto
-89 1 90{/t exch def
ft s   t sin s   t cos s neg ptp lineto
}for stroke
%part upper back x<0 {t:(-ft, sin t, -cos t)}
-1 s   -1 s   0  ptp moveto
-89 1 90{/t exch def
ft neg s   t sin s   t cos s neg  ptp lineto
}for stroke
%part right y>0 {t:(-sin t, ft,  cos t)}
1 s   1 s   0 ptp moveto
-89 1 90{/t exch def
t sin neg s   ft s   t cos  s    ptp lineto
}for stroke
%part left y<0 {t:(-sin t, -ft,  cos t)}
1 s   -1 s   0 ptp moveto
-89 1 90{/t exch def
t sin neg s    ft neg  s    t cos s   ptp lineto
}for stroke
}def%end sphericpart

/dostringing
{%part upper front x>0 {t:(ft, sin t, -cos t)}, f(-90)=1
/v {.333 mul} def /step 2 def% implies number of strings
-90  step 90{/t exch def
ft s     t sin s    t cos s   neg ptp moveto
ft s v   t sin s v  t cos s v neg ptp lineto
}for stroke
%part upper back x<0 {t:(-ft, sin t, -cos t)}
-90 step 90{/t exch def
ft neg s    t sin s    t cos s   neg ptp moveto
ft neg s v  t sin s v  t cos s v neg ptp lineto
}for stroke
%part right y>0 {t:(-sin t, ft,  cos t)}
-90 step 90{/t exch def
t sin neg s     ft s     t cos s     ptp moveto
t sin neg s v   ft s v   t cos s v   ptp lineto
}for stroke
%part left y<0 {t:(-sin t, -ft,  cos t)}
-90 step 90{/t exch def
t sin neg s     ft neg  s     t cos s    ptp moveto
t sin neg s v   ft neg  s v   t cos s v  ptp lineto
}for stroke
}def%end dostringing

/annotation{-32 -90  moveto
 S12pt setfont (j) show ( = ) H12pt setfont show phi (  ) cvs show
   gsave 0 5 rmoveto (o) H7pt setfont show grestore
8 0 rmoveto
 S12pt setfont (q) show ( = ) H12pt setfont show theta (  ) cvs show
         0 5 rmoveto (o) H7pt setfont show}def%end annotation
end%sphericthemedict dictionary

/sphericttheme{sphericthemedict begin reversevideo
/d .25 def 1 setgray
/ft {1 d sub t sin dup mul mul d add} def
3 setlinewidth /scalingfactor 75 def sphericpart
1 setlinewidth /scalingfactor 25 def sphericpart
/scalingfactor 75 def
.5 setlinewidth dostringing
annotation
end}def%sphericttheme
%%EndProlog
%
%---Program--- the script
%
/phi 90 def /theta 10 def  sphericttheme showpage
%%EOF
```







Emulation of Gabo's Linear Construction in Space Suspended

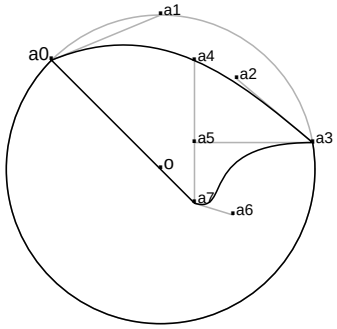
Linear Construction in Space Suspended is one of Gabo's favorites, which he used to display on retrospective exhibitions.

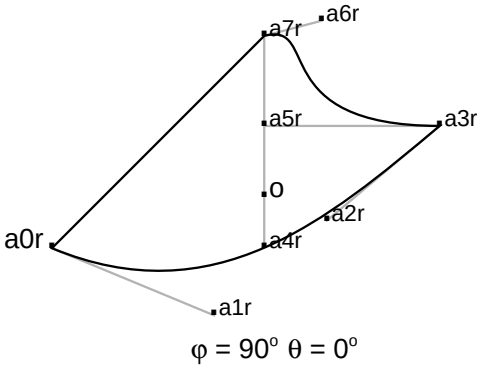
The object consists of 2 identical, 'triangular' slabs of clear plastic orthogonal (rotated and mirrored) to each other, wired, and suspended on a metal arc with a wooden base.

What motivated Gabo to create just these triangular shapes? What were the restrictions? Did the balance of the object play a role, in choosing the rotation symmetry axes? I don't think so, just his feeling for beauty, I guess. The unbalance creates tension.

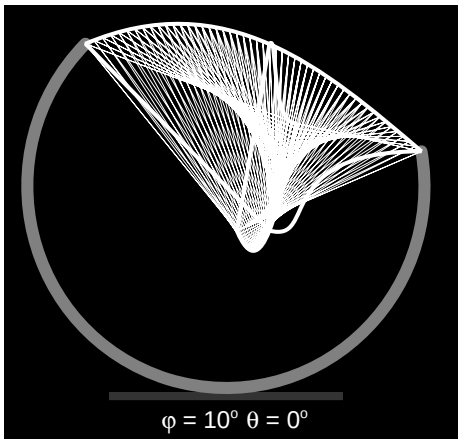
The frame part in the yz plane is realized by 2 splines: `a0 moveto a1 a2 a3 curveto a5 a5 a7 curveto closepath`. For the first B-cubic the data points are a_0, a_3 with control points a_1, a_2. For the second B-cubic the data points are a_3, a_7 with control points a_5, a_6; not critical. The axis of rotation is the line from a_4 to a_7. The suffix `r` denotes the mirrored and rotated triangle points.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Gabo Linear Construction Suspended
%%BoundingBox: -110 -125 115 90
%%BeginSetup
%%EndSetup
%%BeginProlog
(c:\\PSlib\\PSlib.eps) run
/suspendeddict 75 dict def
suspendeddict begin
/reversevideo{-110 -125 225 215 rectfill}def
%data points in yz plane
/a0y { r s sqrt2 div neg} def
/a0z { a0y neg } def
/a0  { a0y a0z } def
/a1y   0   def
/a1z { r s }  def
/a1  { a1y a1z } def
/a3y { r s 10 cos mul }  def
/a3z { r s 10 sin mul } def
/a3  { a3y a3z } def
/a2y { a1y  a3y add 2 div } def
/a2z { a1z  a3z add 2 div}  def
/a2  { a2y a2z } def
/d   {.5 a0 a1 a2 a3 tOnSpline pop} def
/a4y { d }  def
/a4z { .5 a0 a1 a2 a3 tOnSpline exch pop } def
/a4  { a4y a4z } def
/a5y { d } def
/a5z { a3z }def
/a5  { a5y a5z } def
/a6y { d r s 4 div add } def
/a6z { d -1.3 mul } def
/a6  { a6y a6z } def
/a7y { d } def
/a7z { d neg } def
/a7  { a7y a7z } def
/2m  { a4z a7z add } def
%in 3D projected under phi and theta
/a0p { 0 a0y a0z ptp} def
/a1p { 0 a1y a1z ptp} def
/a2p { 0 a2y a2z ptp} def
/a3p { 0 a3y a3z ptp} def
/a4p { 0 a4y a4z ptp} def
/a5p { 0 a5y a5z ptp} def
/a6p { 0 a6y a6z ptp} def
/a7p { 0 a7y a7z ptp} def
```





```
%rotated around a3-a8, mirrored towards m; (a3z + a8z)= 48.579 = 2m
/a0r { a0y neg d add    d    2m a0z sub   ptp} def
/a1r { d                d    2m a1z sub   ptp} def
/a2r { a2y d sub neg    d    2m a2z sub   ptp} def
/a3r { a3y d sub neg    d    2m a3z sub   ptp} def
/a4r { a7p } def
/a5r { 0                d    2m a5z sub   ptp} def
/a6r { a6y d sub neg    d    2m a6z sub   ptp} def
/a7r { a4p} def
%
/arc {%0 0 r s 135 370 arc
135 1 370{/t exch def
r s t cos mul r s t sin mul}for% yz pairs on stack
0 3 1 roll ptp moveto
235{0 3 1 roll ptp lineto}repeat
}def
%
/foot{ .5 r s mul dup neg  dup     2 mul 4 sub ptp moveto% x -y -z
       .5 r s mul dup      dup neg 2 mul 4 sub ptp lineto% x +y -z
      -.5 r s mul dup neg  dup neg 2 mul 4 sub ptp lineto%-x +y -z
      -.5 r s mul dup      dup     2 mul 4 sub ptp lineto%-x -y -z
      closepath
}def
%
/frameyz{a0p  moveto a1p  a2p  a3p  curveto a5p  a6p  a7p  curveto
         closepath} def
/framexz{a0r moveto a1r a2r a3r curveto a5r a6r a7r curveto
         closepath} def
%
/dostringing{
0 0.02 1.01{/t exch def
%t a0p a1p a2p a3p tOnSpline moveto
%t a0r a1r a2r a3r tOnSpline lineto
%t a3p a2p a1p a0p tOnSpline lineto
%t a3r a2r a1r a0r tOnSpline lineto
t a0p a1p a2p a3p tOnSpline moveto
t a3r a2r a2r a0r tOnSpline lineto
t a3p a2p a1p a0p tOnSpline lineto
t a0r a1r a2r a3r tOnSpline lineto
closepath
}for
}def
%
/annotations{[ (a1r) (a2r) (a3r) (a4r) (a5r) (a6r) (a7r)] dotsandnames
 a0r    moveto (.)  H15pt setfont centershow
-23 0 rmoveto (a0r) H12pt setfont show
  0 0  moveto (.)  H15pt setfont centershow
             (o) H12pt setfont show
}def
%
/phiandtheta{
-32 -120  moveto
 S12pt setfont (j) show ( = ) H12pt setfont show phi (   ) cvs show
   gsave 0 5 rmoveto (o) H7pt setfont show grestore
8 0 rmoveto
 S12pt setfont (q) show ( = ) H12pt setfont show theta (   ) cvs show
         0 5 rmoveto (o) H7pt setfont show} def
end%dictionary

/suspended{suspendeddict begin
/r 5 def /scalingfactor 20 def H12pt setfont
/phi 10 def /theta 0 def
reversevideo
gsave .2 setgray foot stroke
   4 setlinewidth foot stroke
   .5 setgray
   6 setlinewidth 1 setlinecap arc stroke
grestore
1 setgray
framexz 2 setlinewidth stroke
frameyz 2 setlinewidth stroke
```



```
gsave 2 setmiterlimit
dostringing .1 setlinewidth stroke grestore
%annotations
phiandtheta
end}def %suspended
%%EndProlog
suspended showpage
%%EOF
```

Use of PSlib.eps in MetaPost

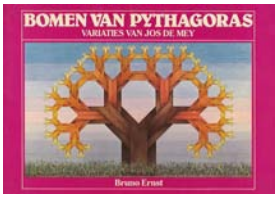
In *Circle Inversions,*MAPS40, 2010, the use of `PSlib.eps` in Metapost has been shown. It complicates, does not obey the SoC principle, and I won't repeat it here.

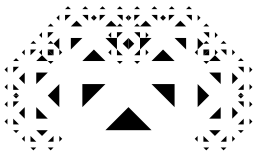
Annotated References

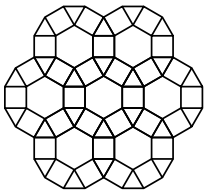
- Adobe Red, Green and Blue Books. The musts for PostScript programmers.
- Adobe PostScript 3 Fonts Set ... come standard with 136 distinctive and stylish fonts, including those packaged with the leading operating systems...
- Bauer, F.L.(1973): Software and software engineering. SIAM REV, vol 15, no 2, 469–480.
- Barnsley, M.F(1988): Fractals Everywhere. Academic Service.

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

Chapter 7 Julia sets. Chapter 8 Parameter Spaces and Mandelbrot sets. BASIC programs are included: 3.8.1 Example of Deterministic Algorithm; 3.8.2 Random Iteration Algorithm; 6.2.1 Fractal Interpolation; 7.1.1 Example of the Escape Time Algorithm. (No efficiency short-cuts such as use of symmetry in the codes.)
- Barnsley, M.F(2006): SuperFractals. Patterns of Nature. Cambridge University Press. 464p. (.....Superfractals would be a superb addition to the bookshelves of any scientists who use fractal analysis techniques in their research, be they physicist, biologist or economist. The author concludes by promising that the introduction of superfractals will revolutionize the way mathematics, physics, biology and art are combined, to produce a unified description of the complex world in which we live. After reading this book, I have no doubt that he is correct. From NATURE Vol 445 18 January 2007.)
- Biography of H.A. Lauwerier: `http://bwnw.cwi-incubator.nl/cgi-bin/uncgi/alf.`
- Courant, R(1937, sec.ed.): Differential and Integral Calculus.
- Deubert, J: Acumen Journal. `http://www.acumentraining.com/acumenjournal.html`. Highly educative, with respect to PostScript, PDF and XPS. (From the november2011 issue: PostScript Tech - Transparency in PostScript Using pdfmark PostScript implements a strictly opaque imaging model; objects painted on the page completely obscure anything on the page beneath them. That is, unless you are handing your PostScript file to Distiller; in that case your PostScript code can draw translucent objects on the final PDF page using the Distiller-only pdfmark operator.)
- PostScript FAQ `http://en.wikipedia.org/wiki/Wikibooks:PostScript_FAQ`
- 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.)

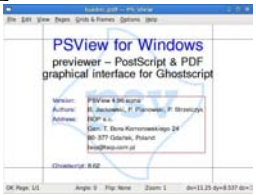






- Fractal Foundation. `http://www.fractalfoundation.org`.

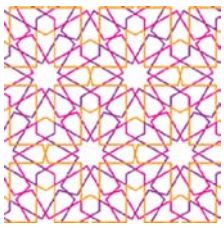
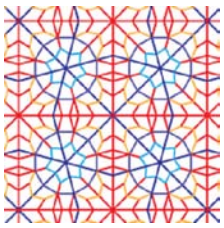
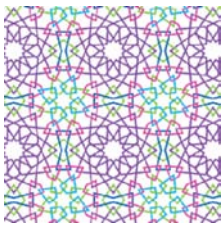
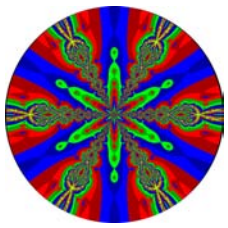
- Gabo, N (1987): 60 years of constructivism. Prestel-Verlag.
- Goossens, M(2007, sec.ed.) c.s.: LaTeX Graphics Companion. ISBN 978 0 321 50892 8.
- Heck, A(2005): Learning MetaPost by Doing. MAPS 32. (A tutorial with nice practical examples.)
- Henderson, T(2009): MetaPost previewer. `www.tlhiv.org/mppreview`. TeXLive2014.
- Hobby, J.D(1995): Drawing Graphs with MetaPost. CSTR Report 16.
- Jackowski, B, P. Strzelczyk, P. Pianowski(1995-2008): PSView5.12.
`bop@bop.com.pl`. (Extremely fast previewer for `.eps` and `.pdf`.
Allows `PSlib`(rary) inclusion via the `run` command.
Error messages appear in the pop-up GhostScript window.)



- Knuth, D.E, T. Larrabee, P.M. Roberts(1989): Mathematical Writing.
MAA notes 14. The Mathematical Association of America.

- Knuth, D.E(1990, 10^{th} printing): The TEXbook. Addison-Wesley. ISBN 0-201-13447-0. (A must for plain TEXies. I never read a manual so many times. Well ... Adobe's RGB-books come close.)
- Knuth, D.E(1986): The Metafont book. Addison-Wesley. ISBN 0-201-13445-4.
- Kroonenberg, S(2007): Epspdf, easy conversion between PostScript and PDF. EuroBachoTEX.
- Lancaster, D(1989): PSsecrets. From the www.
- Lancaster's PostScript use: `http://www.tinaja.com`, for free. Informative.
- Lauwerier, H.A(1987): Analyse met de Microcomputer. Epsilon 7.
- Lauwerier, H.A(1987): FRACTALS — meetkundige figuren in eindeloze herhaling. Aramith. (Contains BASIC programs. Lauwerier, H.A (1991): Fractals: Endlessly Repeated Geometrical Figures. Translated by Sophia Gill-Hoffstadt, Princeton University Press, Princeton NJ. ISBN 0-691-08551-X, cloth. ISBN 0-691-02445-6 paperback. "This book has been written for a wide audience ... " Includes sample BASIC programs in an appendix. With respect to Julia fractals it contains only the programs JULIAB(acktracking) and MANDEL, for the black-and-white banded Mandelbrot fractal. Intended for instructors, (high-school) students,) and the educated layman.)
- Lauwerier, H.A(1988): Meetkunde met de Microcomputer. Epsilon 8.
- Lauwerier, H.A(1989): Oneindigheid — een onbereikbaar ideaal. Aramith. ISBN 90 6834 055 7. (Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1990): Een wereld van FRACTALS. Aramith. ISBN 90 6834 076 X. (A sequel and updated version of Lauwerier(1987). Intended for instructors, (high-school) students, and the educated layman. Contains a.o. PYT3DBT a BASIC backtracking program for 3D bare Pythagoras Trees.)

- Lauwerier, H.A(1992): Computer Simulaties — De wereld als model. Aramith. ISBN 90 6834 106 5. (The last chapter is called Orde en Chaos, and applies to this paper.)
- Lauwerier, H.A(1994): Spelen met Graphics and Fractals. Academic Service. ISBN 90 395 0092 4. (An inspiring book with Math at the high-school level for a wide audience. The BASIC programs I consider outdated for direct use. Intended for instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1995): Symmetrie, Kunst en Computers. Aramith. (The permutation group P_n is used to classify various symmetries. The possible tilings of the plane by regular polygons are discussed and their mathematical classification is explained. Tilings of Duat and Penrose are included. The Platonic and Archimedic polyhedra are treated. The symmetries in the (pseudo) Julia and Mandelbrot fractal are mentioned. Intended for instructors, (high-school) students, and the educated layman. The BASIC codes are available from the fileserver of the THE.)

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

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

- Mandelbrot, B(1982); The Fractal structure of Nature. ISBN 07 167 11869.
(This essay of erudition, originality and insight, is about the fundamentals of Math: Euclidean geometry and the dimension concept are widened up into fractal geometry and fractal dimension. A plea is made for fractals to describe natural phenomena: clouds, waves, landscapes, length of coasts with the ill-posed question of length answered by: each coastline has a fractal dimension, The 'montruous' curves of 19^{th} century Math find a honourable niche in fractal geometry, with a fractal dimension added.)

- Minsky, M(1969): Form and Content in Computer Science. Turing Award lecture. In: ACM Turing Award Lectures. The first twenty years 1966-1985. ACM Press. (With TEX and similar quality tools for typesetting, I get the impression that pre-press publications within the TEX-community are out of balance: too much attention is paid to form.)
- Peitgen, H.O, H.Jürgens, D. Saupe(2004 sec.ed.): Chaos and Fractals. New frontiers of Science. (Images of the fourteen chapters of this book cover the central ideas and concepts of chaos and fractals as well as many related topics including: the Mandelbrot set, Julia sets, cellular automata, L-systems, percolation and strange attractors. This new edition has been thoroughly revised throughout. The appendices of the original edition were taken out since more recent publications cover this material in more depth.[26] Instead of the focused computer programs in BASIC, the authors provide 10 interactive JAVA-applets for this second edition via `http://www.cevis.uni-bremen.de/fractals`. An encyclopaedic work. Multiple Reduction Copying Machines, as model for feedback systems, are associated with IFSystems, The Barnsley fern has been elaborated upon and the chaos game is explained. Audience: No mathematical sophistication is required, so a broad audience is aimed at. It portrays the new fields: Chaos and fractals, in an authentic manner. The first edition has been typeset by (La)TEX&Co, the second edition as well, I presume.)
- Peitgen, H.O, P.H. Richter(1986): The Beauty of Fractals. Images of complex dynamical systems. Springer-Verlag. (Frontiers of Chaos. Verhulst Dynamics. Julia Sets and their Computergraphical generation. Sullivan's Classification and Critical Points. The Mandelbrot Set. External Angles and the Hubbard Trees. Newton's method for Complex Polynomials: Cayley's Problem. Newton's method for Real Equations. A discrete Volterra-Lotka System. Magnetism and Complex boundaries. Yang-Lee Zeros. Invited contributions:
Fractals and the Rebirth of Iteration Theory by B. Mandelbrot;
Julia Sets and the Mandelbrot Set by A. Douady;
Freedom, Science, and Aesthetics by G. Eilinger;
Refractions on Science into Art by H.W. Franke.
The book ends with DO IT YOURSELF and DOCUMENTATION.)
- Swanson, E(1986, rev.ed.): Mathematics into Type. American Mathematical Society.
- Polya, G(1957): How to solve it.
- 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(1993): What is mTEX and metafont all about? MAPS 11, 67–87.
- Van der Laan, C.G(1993): Blues bibliography. MAPS 11, 205–210.
- Van der Laan, C.G(1993): Matrix Icons via LaTEX. MAPS 11, 211–212.
- Van der Laan, C.G(1994): Blue's Graphs. MAPS 131, 200–206.
- Van der Laan, C.G(1995): Publishing with TEX. Public Domain. (See TEX archives. `BLUe.tex` comes with `pic.dat` the database of pictures in TEX-alone. No Julia fractals, but some strange attractors have been done by TEX-alone. The advantage of TEX alone approach is portability in place and time. All chapters still compile under pdfTEX without adaptation. The inelegance of the font aspects is inherited from the bitmap-fonts plain TEX-engine. A successor of this work should be based on an OTF-TEX-engine.)

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

- Van der Laan, C.G(unpublished, BachoTEX workshop): TEXing Paradigms. (A plea is made for standardized macro writing in TEX to enhance readability and correctness. Topics: Plain's `items` extended, Headache (about headings), Two-part macros. Parametrization I — Options, The wind and hallfwinds (macros for turtle graphics in TEX), It's all in the game — Dialogue with TEX, Loops, Searching, Sorting, Just a little bit of PostScript, FIFO and LIFO sing the BLUes, Syntactic Sugar.)
- Van der Laan, C.G(1996): Paradigms. It's all in the game. MAPS 16, 82–84.
- Van der Laan, C.G(1996): Paradigms. The winds and halfwinds. Details matter. MAPS 16, 85–90.
- Van der Laan, C.G(1996): Just a little bit of PostScript. MAPS 17. 137–150.
- Van der Laan, C.G(1997): Graphics and TEX— a reappraisal of Metafont and PostScript.. MAPS 16. 100–107.
- Van der Laan, C.G(1997): Stars around I. PostScript straight away. MAPS 18. 93–98.
- Van der Laan, C.G(1997): Stars around II. What a little math can do. MAPS 18. 99–102.
- Van der Laan, C.G(1997): DVIPS manual —- good old portability and some more. MAPS 18. 155–159.
- Van der Laan, C.G(1997): Tiling in PostScript and Metafont — Escher's wink. MAPS 19. 39–67.
- Van der Laan, C.G(2009): TEX Education — an overlooked approach. EuroTEX2009-3rdConTEXt proceedings. MAPS 39. E5–E33. (Launched my `PSlib.eps` library.)
- Van der Laan, C.G(2010): Circle Inversions —with a serious undertone—. MAPS 40. 9–65. (Contains the solution of Apollonius problem as a PS `def`. `PSlib.eps` is introduced.)
- Van der Laan, C.G(2010): `à la Mondrian` MAPS 41. 79–90. Presented at BachoTEX 2010.
- Van der Laan, C.G(2011): Gabo's Torsion. MAPS 42. 69–110. (Contains also a summary of the PostScript language and its developments.)
- Van der Laan, C.G(2012): Pythagoras Trees in PostScript — Fractal Geometry 0. EuroBachoTEX2012-proceedings, MAPS 44, 27–48. Submitted Informatsionnie Texnologii i Matematicheskoe Modelirovanie.
- Van der Laan, C.G(2012): Classical Math Fractals in PostScript — Fractal Geometry I. EuroBachoTEX2012-proceedings, MAPS 44, 49–78. Submitted Informatsionnie Texnologii i Matematicheskoe Modelirovanie.
- Van der Laan, C.G(2012): Recreational use of TEX&Co. EuroTEX2012-6thConTEXt proceedings.
- Van der Laan, C.G(2012): Julia fractals in PostScript. EuroTEX2012-6thConTEXt proceedings.
- Van der Laan, C.G(2013): Spirals in PostScript — Polar coordinates and PostScript are mates. EuroBachoTEX2013-proceedings, 59–66, Submitted MAPS.
- Van der Laan, C.G(2013): Head and tail in summation — catching up with numerical math and mathematica. EuroBachoTEX2013-proceedings, 67–74. Submitted MAPS.
- Voss, H(2011): PSTricks—Graphics and PostScript for TEX and LaTEX. UIT Cambridge Ltd. ISBN 978-1-906860-13-4.
- Wirth, N(1976): Algorithms + Data Structures = Programs. Prentice-Hall.

Conclusions

The use of PostScript pictures in AllTEX documents has a history of 20 years already, to start with the use of `DVIPS`. With pdfTEX the use of direct PostScript inclusion was hampered. `.eps` has to be converted into `.pdf`. Adobe seems to have stopped the support of the run command for library inclusion, which is a big loss. It seems that the use of PostScript pictures is still possible in TEX documents, but the future does not look so good. This holds also for MetaPost and PStricks pictures, because Metapost is just a preprocessor for PostScript and PSTricks uses PostScript under the hood. For the future we need a better and stable graphics tool to cooperate with TEX&Co. On the other hand: is it too optimistic to expect that the PostScript programs will be read and used?
A different class of pictures come from fractals, which are generated in a new way than just drawing.
The Lorenz and Rössler attractor have revolutionized themethod of solution of nonlinear numerical partial differential equations.

Acknowledgements

Thank you Adobe for your maintained, adapted to LanguageLevel 3 since 1997, good old, industrial standard PostScript and `Acrobat Pro` (actually DISTILLER) to view it, Don Knuth for your stable plain TEX, Jonathan Kew for the TEXworks IDE,
Hàn Thế Thành for pdf(La)TEX,
Thank you Jos Winnink and Henk Jansen for proofing. MAPS editors for improving my use of English.