Pythagoras Trees in PostScript

Fractal Geometry 0

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

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

Contents

Introduction

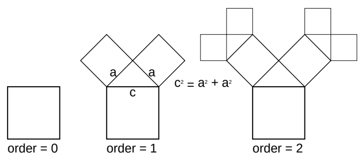
In the 70s, I was still a student, CWI treated its employees and relations on a calendar — human-meets-plotter — with illustrations made by the Calcomp 565 plotter, among others 3 Pythagoras Trees. At the time the Pythagoras Tree was a programming challenge in ALGOL60/68, the programming languages in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

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

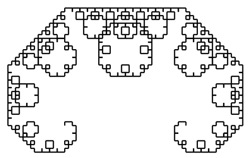
The word FRACTAL did not yet exist for us, although the spirit was all over, endlessly repeated geometric figures buzzed around, apparently, with BASIC programs by Lauwerier(1987).

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

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



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



← Lévy C curve

Julia set →

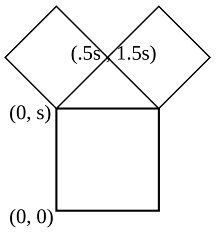
Properties The sum of the surfaces of the squares $\rightarrow \infty$ as does the circumference. The kind of infinity is characterized by the fractal dimension notion.[1] The circumference has fractal dimension 2, a local plane-filling curve.

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

The production rule for a Pythagoras Tree of order n reads

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

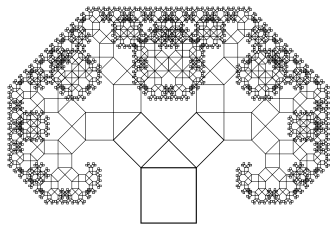
with P_n the Pythagoras Tree of order n, $\oplus$ denotes splice operator, means add properly, [means store graphics state and open a new one,] means close current graphics state and recall previous, R^{45} means rotate US 45° in the PS sense, $S_{a,b}$ means scale US by a and b, $T_{a,b}$ means translate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



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

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

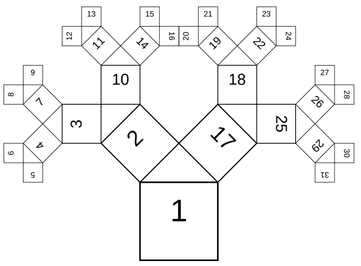
On the WWW I did not find Algol nor PASCAL nor … versions and only those by John Deubert in PS. From Lauwerier's books I picked up the Trees in BASIC. My Algol68 program — lost alas, after so many years — was definitely not so elegant as my current PS.



The PostScript program

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

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



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

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

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

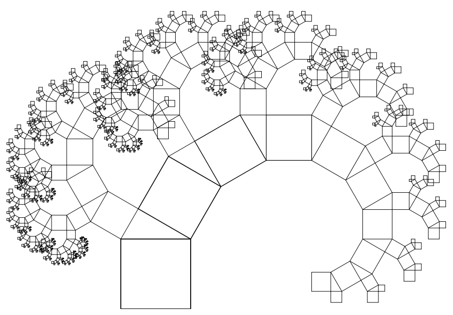
Variations

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

Realistic tree

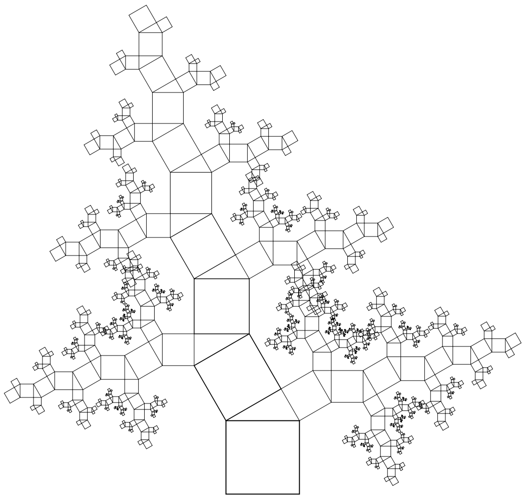
Oblique tree Let the skewness be denoted by ϕ (60°, say). We have to translate, rotate and scale before each recursive invoke. The left descendant is placed, as in the symmetrical case, at $(0, s)$ with the user space rotated by ϕ, and scaled by $(\cos\phi, \cos\phi)$. The right descendant is placed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

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



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

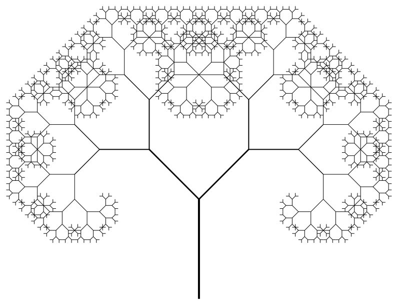
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: (Pythagoras) X-mas Tree of squares
%%Author: Kees van der Laan, kisa1@xs4all.nl, April 2011
%%BoundingBox: -235 -2 305 500
%%BeginSetup
%%EndSetup
%%BeginProlog
/drawsquare{0 0 s s rectstroke}def
/Xmastree{%on stack integer (order>=0) ==> XmasTree
          %Globals: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



Simplified Tree

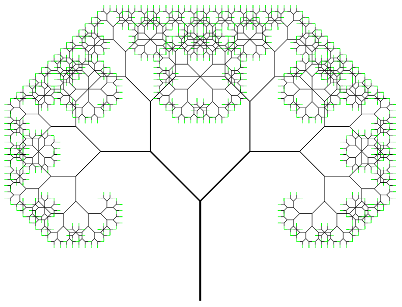
We may simplify the tree by drawing a line instead of a square. This implies that the `translate` — first adaptation of the User Space — is invoked at the beginning after drawing the line. The rotation and scaling adaptations of the user space remain at the same place.

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



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

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

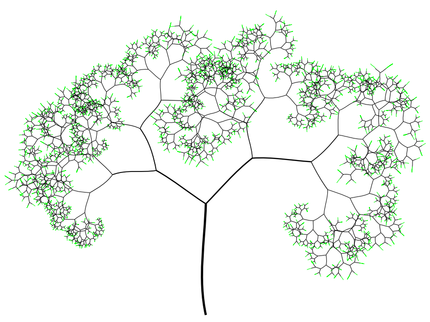


Simplified randomized Tree We might transform the user space irregularly, by some spread variation around 45°, and also with varying randomly the scale i.e. the length s of the line. Because of the diminishing scale the branches become thinner, automatically.

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

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

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

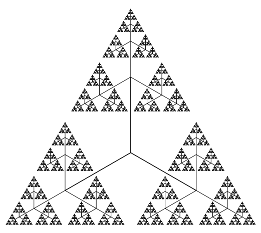


What if we don't vary randomly but according to certain rules of growth? Maybe the program can be adapted and simulate real trees, avoiding 3D and projection?

Trinary Trees

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

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

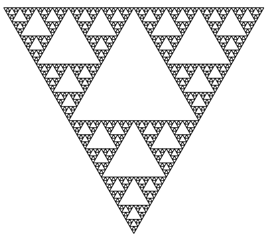


...

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

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

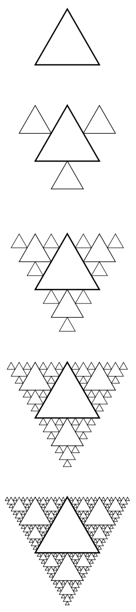
The idea behind the code below is to identify each triangle with a trinary number. Possibly because it was a side-step Lauwerier did not provide a more efficient backtracking variant.



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

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

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



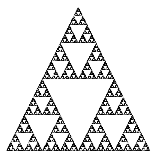
```
%%BeginProlog
/y{3 sqrt 4 div s mul}def
/SierTri{%on stack order >=1 ==> Sierpinski triangle fractal ( s global)
1 sub  dup 0 ge
{gsave 0 0 moveto s 0 lineto s 2 div .869 s mul lineto closepath stroke grestore%order1
 gsave s 4   div y neg translate .5 dup scale SierTri grestore
 gsave s .75 mul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

By iterated function system

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

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

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

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



Randomized Sierpiński triangle

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

Pascal triangle and Sierpiński triangle

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

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

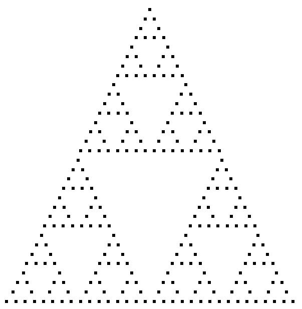
yields

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

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

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

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



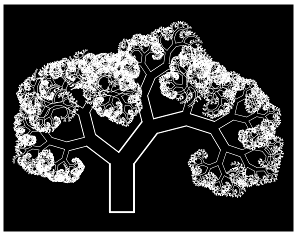
```
\newcount\n \newcount\rcnt \newcount\ccnt \newcount\tableentry \newcount\prev
\let\ea=\expandafter
%
\def\pascal#1{\n#1 \def\0{1} \ccnt1
\loop\ea\xdef\csname\the\ccnt\endcsname{0} %auxiliary sentinel
\ifnum\ccnt<\n \advance\ccnt1\repeat
\rcnt0 \ccnt0 \displaylines{\rows}}
%
\def\rows{\global\advance\rcnt1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

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

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

Tree with stem

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

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



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

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

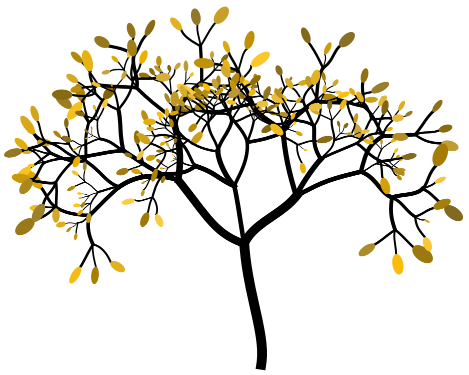
John Deubert's Tree

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

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

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

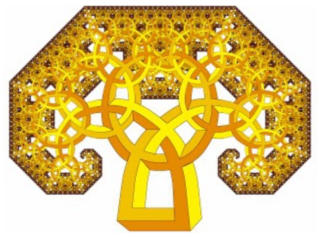
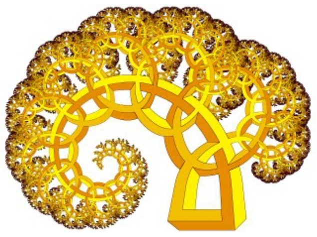
Next he discusses what might limit recursion: the sizes of the various stacks are finite, especially the graphics state stack — pushed by gsave and popped up by grestore — might cause problems by too deep recursion. Adobe implemented, since LanguageLevel 2, flexible stacks, where the (stack) limits are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



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

Art Trees from the WWW

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

2.5D Pythagoras Trees

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

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

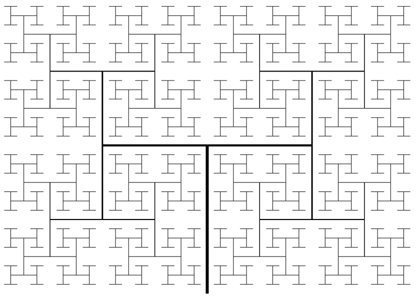
The growth rules are: from the stem branches have angle σ_k and each branch has 2 subbranches with angles α_{k_i}, β_{k_i} which lie in a plane perpendicular to the plane formed by the stem and the branch. Reduction factors are parameters.

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

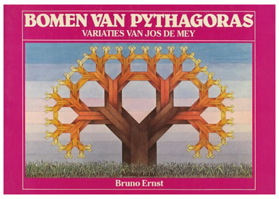
Annotated References

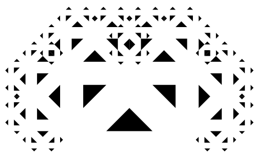
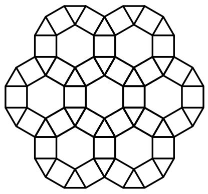
- An introductory survey: `http://en.wikipedia.org/wiki/Pythagoras_tree`.
- Adobe Red, Green and Blue Books. The musts for PS programmers. The simplified tree is a variant of `FractArrow` as given in the BLue Book p.74, which also embodies the H-fractal. One only has to vary the rotation angle, as can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

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



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



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

The appendices of the original edition were taken out since more recent publications cover this material in more depth. Instead of the focused computer programs in BASIC, the authors provide 10 interactive JAVA-applets for this second edition via `http://www.cevis.uni-bremen.de/fractals`. An encyclopedic work. Audience: Accessible without mathematical sophistication and portrays the new fields: Chaos and fractals, in an authentic manner.)

- Swanson, E(1986, revised ed): Mathematics into Type. American Mathematical Society.
- Szabó, P(2009): PDF output size of TEX documents. Proceedings EuroTEX2009/ConTEXt, p57–74. (Various tools have been compared for the purpose.)
- Van der Laan, C.G(1992): LIFO and FIFO sing the Blues. MAPS 92.2.
- Van der Laan, C.G(1995): Publishing with TEX. Public Domain. (See TEX archives.)
- Van der Laan, C.G(1997): Tiling in PostScript and MetaFont — Escher's wink. MAPS 97.2.
- Van der Laan, C.G(unpublished, BachoTEX workshop): TEXing Paradigms. (A plea is made for standardized macro writing in TEX to enhance readability and correctness.)
- Van der Laan, C.G(2009): TEX Education: an overlooked approach. EuroTEX2009-3rd ConTEXt proceedings. (Launched my `PSlib.eps` library.)
- Van der Laan, C.G(2011): Gabo's Torsion. MAPS 42. (Contains a summary of the PS language and its developments.)
- Van der Laan, C.G(submitted MAPS): Julia fractals in PostScript.
- Veith, U(2009): Experiences typesetting mathematical physics. Proceedings EuroTEX2009/ConTEXt, p31–43. (Practical examples where we need to adjust TEX's automatic typesetting.)
- Links to Pythagoras Trees in Art
  http://www.arsetmathesis.nl (A site about Art&Math, such as Pythagoras Trees, Escher, … .)
  `http://mathpaint.blogspot.com/2007/03/pythagoras-tree.html`
  `http://flickr.com/search/?w=32286042@N00&q=Pythagoras&m=text.`

Conclusions

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

PS programs can be written as readable as literature.

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

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

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

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

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

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

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

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

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

Acknowledgments

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

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

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

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

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

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

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

Afterthoughts: Pondering about my languages and tools

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

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

The TEX collection DVD with

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

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

Would I advise new users TEX&Co?

20 years ago I would.

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

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

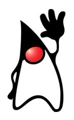
Adobe InDesign CS5.5 software lets you design and preflight engaging page layouts for print or digital distribution with built-in creative tools and precise control over typography. Integrate interactivity, video, and audio for playback on tablets, smart-phones, and computers. No hands-on experience.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

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

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

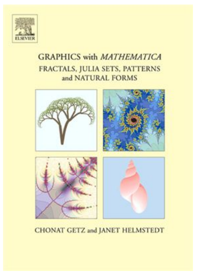
Acrobat is another useful tool with its `.pdf` format. It was developed to compensate for PS deficiencies in porting documents, especially the glyphs of the used fonts were not correctly rendered everywhere. I use `Acrobat Pro` as interpreter/viewer for my EPSF. `Acrobat Reader`, the `.pdf` viewer, is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

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


Java Duke

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

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


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

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

Quite something!

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

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

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

Notes

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

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

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

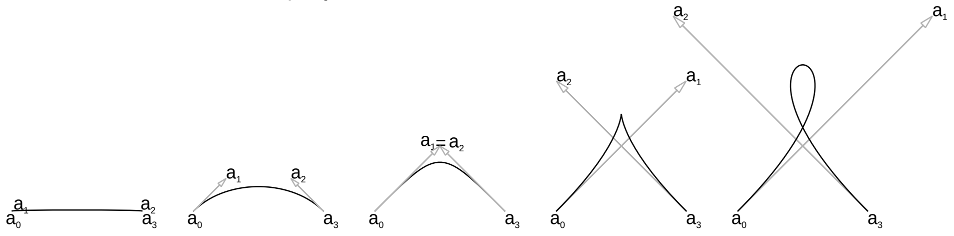
Appendix 0 Splines

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

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

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

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



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

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

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

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

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

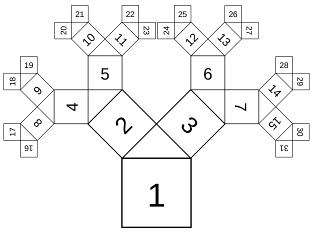
A PS spline path is created by `curveto` and painted to the current page by `stroke`. For the evaluation the de Casteljau algorithm is generally used

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

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

Appendix 1: Lauwerier's BASIC versions

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



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

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

Notes

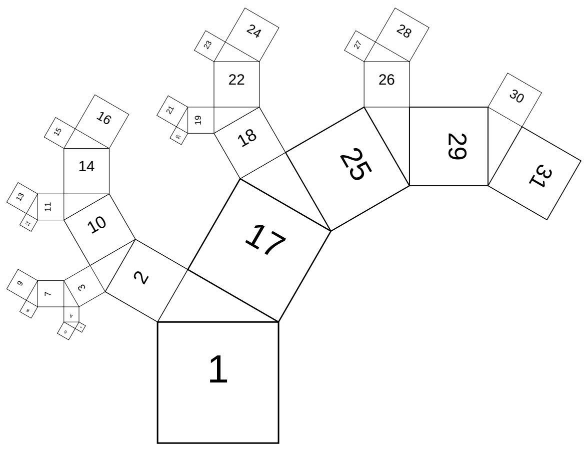
In the transcription use has been made of

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

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

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

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



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

Appendix 2: My Metafont programs of old

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

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

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

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

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

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