Fractals: pst-fractal v0.05 Documentation

Herbert Voß*

September 25, 2007

Abstract

The well known `pstricks` package offers excellent macros to insert more or less complex graphics into a document. `pstricks` itself is the base for several other additional packages, which are mostly named `pst-xxxx`, like `pst-fractal`.

This version uses the extended keyval package `xkeyval`, so be sure that you have installed this package together with the spcecial one `pst-xkey` for PSTricks. The `xkeyval` package is available at CTAN:/macros/latex/contrib/xkeyval/. It is also important that after `pst-fractal` no package is loaded, which uses the old keyval interface.

The fractals are really big, which is the reason why this document is about 15 MByte when you run it without using the external png-images.

Contents

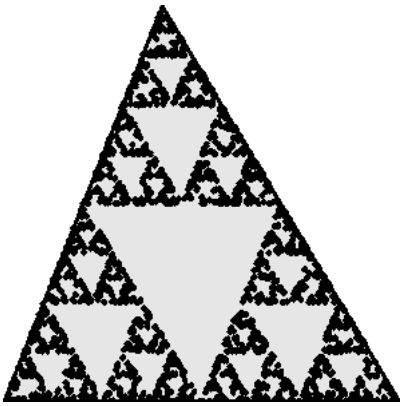
*voss@perce.de

1 Sierpinski triangle

The triangle must be given by three mandatory arguments:

```
\psSier[settings](x0,y0)(x1,y1)(x2,y2)
```

In difference to psfractal it doesn't reserve any space, this is the reason why it should be part of a pspicture environment.



```
1 \begin{pspicture}(5,5)
2   \psSier(0,0)(2,5)(5,0)
3 \end{pspicture}
```

2 Julia and Mandelbrot sets

The syntax of the `psfractal` macro is simple

```
\psfractal[settings](x0,y0)(x1,y1)
```

All Arguments are optional, `psfractal` is the same as `\psfractal(-1,-1)(1,1)`.

The Julia and Mandelbrot sets are a graphical representation of the following sequence

x is the real and y the imaginary part of the complex number z. $C(x, y)$ is a complex constant and preset by $(0, 0)$.

$$z_{n+1}(x, y) = (z_n(x, y))^2 + C(x, y) \tag{1}$$
$$\tag{2}$$

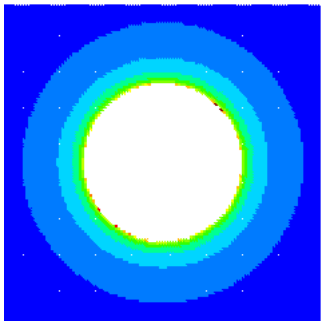
2.1 Julia sets

A Julia set is given with

$$z_{n+1}(x, y) = (z_n(x, y))^2 + C(x, y) \tag{3}$$
$$z_0 = (x_0; y_0) \tag{4}$$

$(x_0; y_0)$ is the starting value.

```
\psfractal
```



```
\psfractal[xWidth=4cm,yWidth=4cm, baseColor=white, dIter=20](-2,-2)(2,2)
```

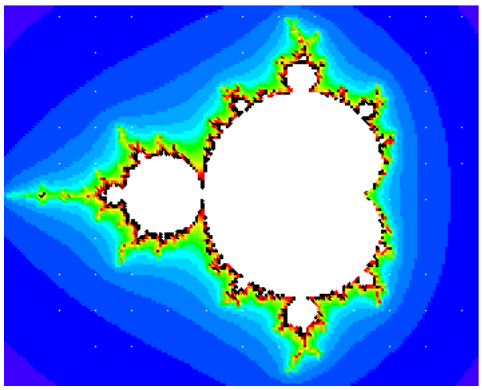
2.2 Mandelbrot sets

A Mandelbrot set is given with

$$z_{n+1}(x,y) = (z_n(x,y))^2 + C(x,y) \tag{5}$$
$$z_0 = (0;0) \tag{6}$$
$$C(x,y) = (x_0;y_0) \tag{7}$$

$(x_0;y_0)$ is the starting value.

```
\psfractal[type=Mandel]
```

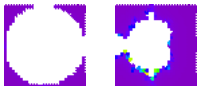


```
\psfractal[type=Mandel, xWidth=6cm, yWidth=4.8cm, baseColor=white, dIter
  =10](-2,-1.2)(1,1.2)
```

2.3 The options

2.4 type

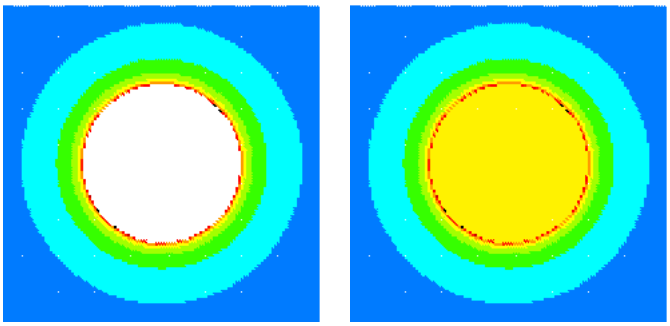
Can be of "‘Julia"’ (default) or "‘Mandel"’.



```
\psfractal
\psfractal[type=Mandel]
```

2.5 baseColor

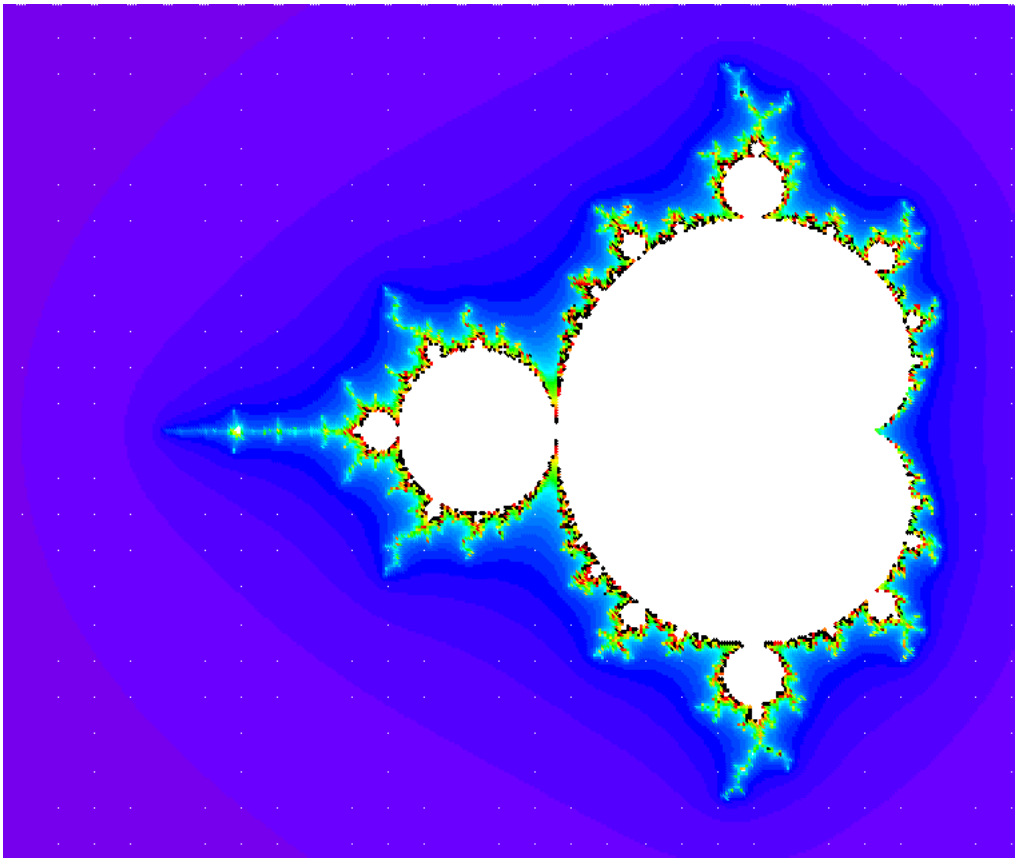
The color for the convergent part.



```
\psfractal[xWidth=4cm,yWidth=4cm,dIter=30](-2,-2)(2,2)
\psfractal[xWidth=4cm,yWidth=4cm,baseColor=yellow,dIter=30](-2,-2)(2,2)
```

2.6 xWidth and yWidth

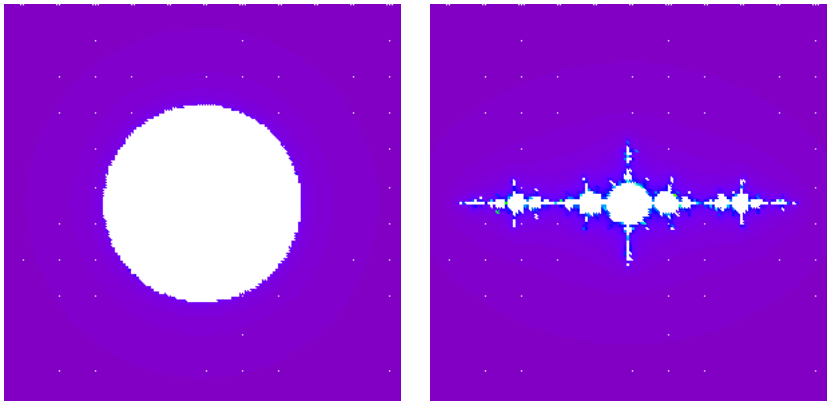
These values define the physical width of the fractal.



```
\psfractal[type=Mandel,xWidth=12.8cm,yWidth=10.8cm,dIter=5](-2.5,-1.3)
  (0.7,1.3)
```

2.7 cx and cy

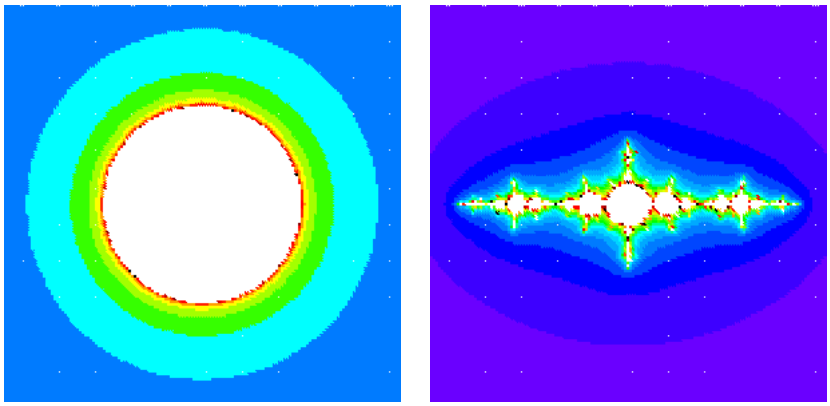
Define the starting value for the complex constant number C.



```
\psset{xWidth=5cm,yWidth=5cm}
\psfractal[dIter=2](-2,-2)(2,2)
\psfractal[dIter=2,cx=-1.3,cy=0](-2,-2)(2,2)
```

2.8 dIter

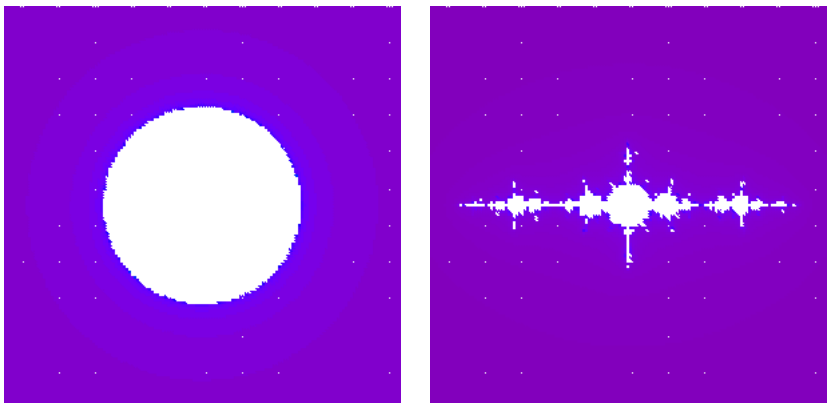
The color is set by wavelength to RGB conversion of the iteration number, where `dIter` is the step, predefined by 1. The wavelength is given by the value of iter added by 400.



```
\psset{xWidth=5cm,yWidth=5cm}
\psfractal[dIter=30](-2,-2)(2,2)
\psfractal[dIter=10,cx=-1.3,cy=0](-2,-2)(2,2)
```

2.9 maxIter

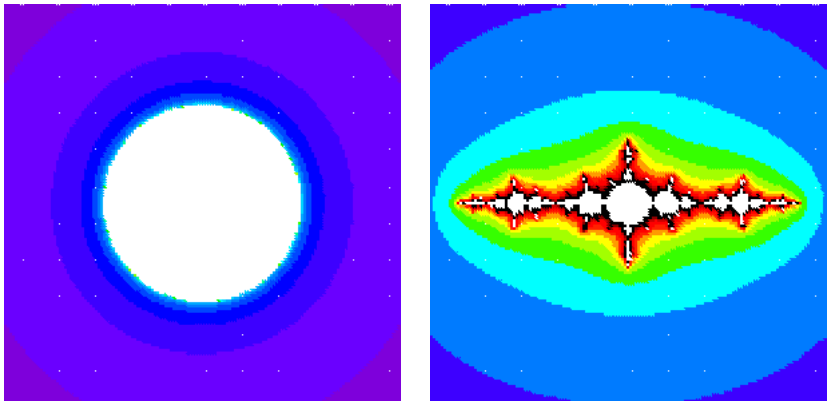
maxIter is the number of the maximum iteration until it leaves the loop. It is predefined by 255, but internally multiplied by dIter.



```
\psset{xWidth=5cm,yWidth=5cm}
\psfractal[maxIter=50,dIter=3](-2,-2)(2,2)
\psfractal[maxIter=30,cx=-1.3,cy=0](-2,-2)(2,2)
```

2.10 maxRadius

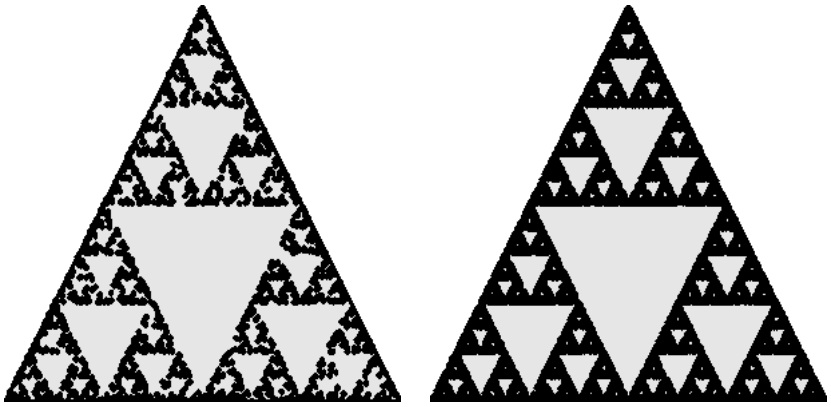
If the square of distance of z_n to the origin of the complex coordinate system is greater as maxRadius then the algorithm leaves the loop and sets the point. maxRadius should always be the square of the "'real"' value, it is preset by 100.



```
\psset{xWidth=5cm,yWidth=5cm}
\psfractal[maxRadius=30,dIter=10](-2,-2)(2,2)
\psfractal[maxRadius=30,dIter=30,cx=-1.3,cy=0](-2,-2)(2,2)
```

2.11 plotpoints

This option is only valid for the Sierpinski triangle and preset by 2000.



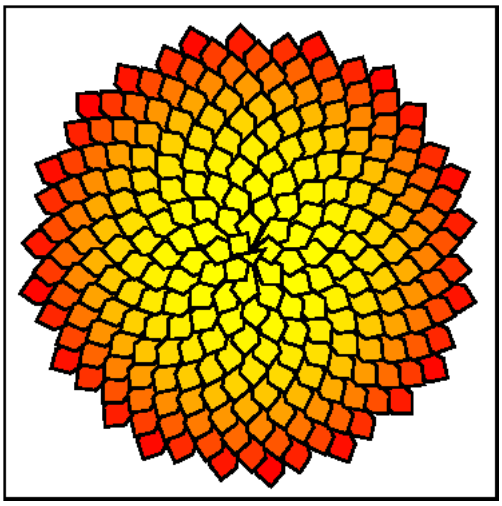
```
\begin{pspicture}(5,5)
  \psSier(0,0)(2.5,5)(5,0)
\end{pspicture}
\begin{pspicture}(5,5)
  \psSier[plotpoints=10000](0,0)(2.5,5)(5,0)
\end{pspicture}
```

3 Phyllotaxis

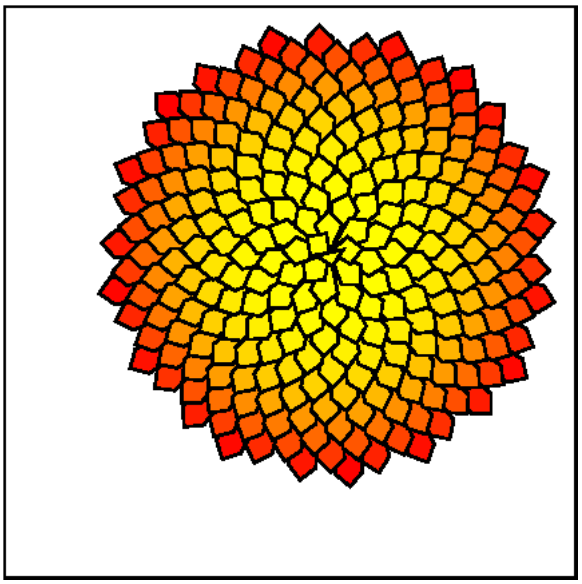
The beautiful arrangement of leaves in some plants, called phyllotaxis, obeys a number of subtle mathematical relationships. For instance, the florets in the head of a sunflower form two oppositely directed spirals: 55 of them clockwise and 34 counterclockwise. Surprisingly, these numbers are consecutive Fibonacci numbers. The Phyllotaxis is like a Lindenmayer system.

```
\psPhyllotaxis[settings](x,y)
```

The coordinates of the center are optional, if they are missing, then $(0, 0)$ is assumed.

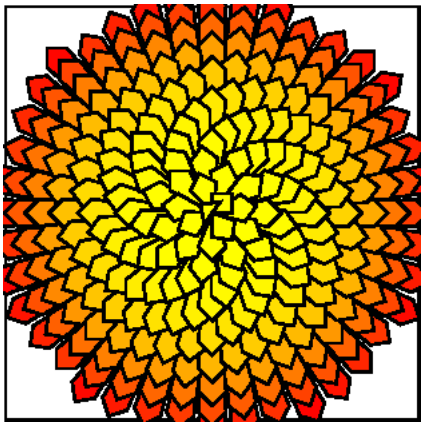


```
\psframebox{\begin{pspicture}(-3,-3)(3,3)
  \psPhyllotaxis
\end{pspicture}}
```



```
\psframebox{\begin{pspicture}(-3,-3)(4,4)
  \psPhyllotaxis(1,1)
\end{pspicture}}
```

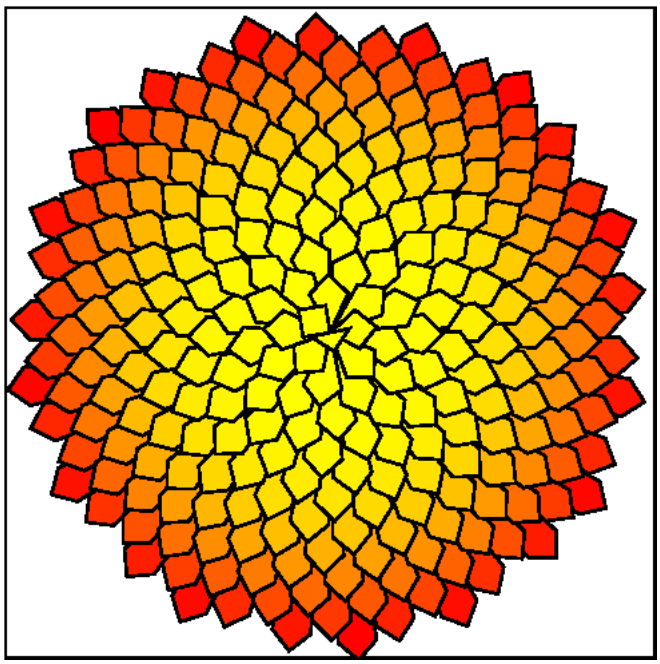
3.1 angle



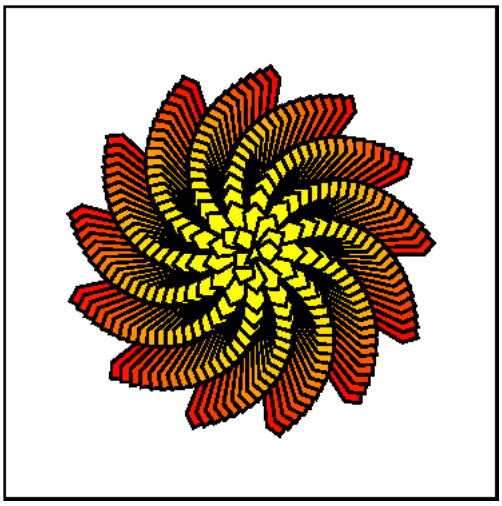
```
\psframebox{\begin{pspicture}(-2.5,-2.5)(2.5,2.5)
  \psPhyllotaxis[angle=99]
\end{pspicture}}
```

3.2 c

This is the length of one element in the unit pt.



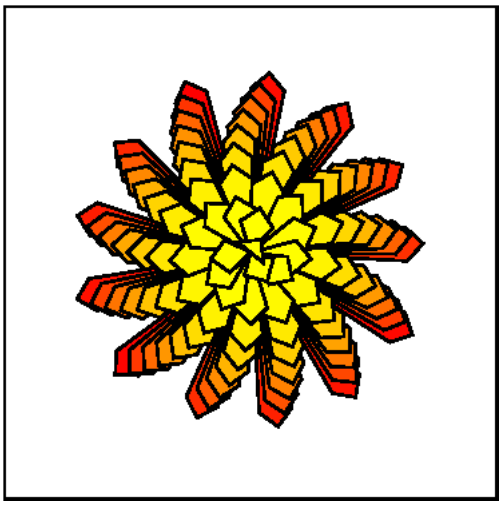
```
\psframebox{\begin{pspicture}(8,8)
  \psPhyllotaxis[c=7](4,4)
\end{pspicture}}
```



```
\psframebox{\begin{pspicture}(-3,-3)(3,3)
  \psPhyllotaxis[c=4,angle=111]
\end{pspicture}}
```

3.3 maxIter

This is the number for the iterations.



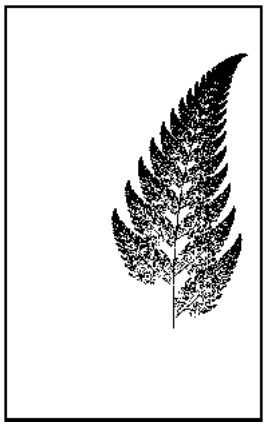
```
\psframebox{\begin{pspicture}(-3,-3)(3,3)
  \psPhyllotaxis[c=6,angle=111,maxIter=100]
\end{pspicture}}
```

4 Fern

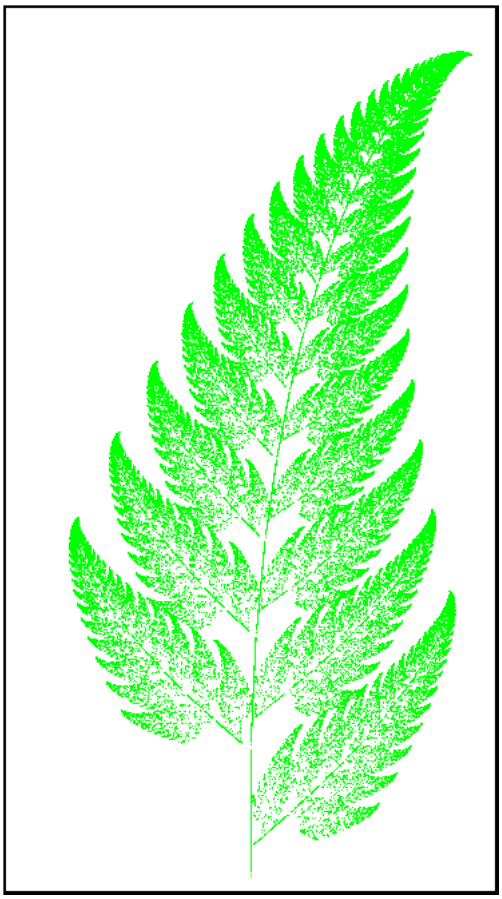
```
\psFern[settings](x,y)
```

The coordinates of the starting point are optional, if they are missing, then $(0, 0)$ is assumed.

```
\psframebox{\begin{pspicture}(-1,0)(1,4)
  \psFern
\end{pspicture}}
```



```
\psframebox{\begin{pspicture}(-1,0)(2,5)
  \psFern(1,1)
\end{pspicture}}
```

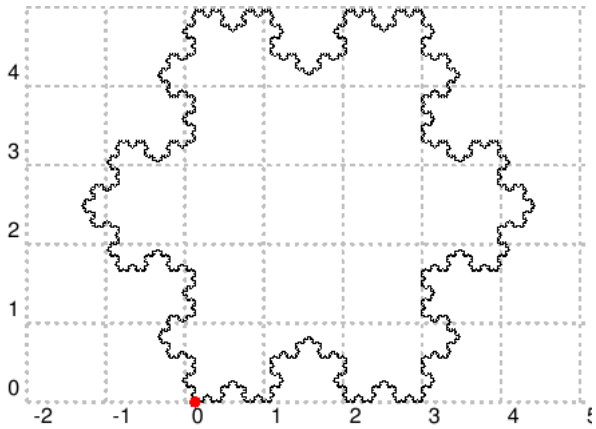


```
\psframebox{\begin{pspicture}(-3,0)(3,11)
  \psFern[scale=3,maxIter=100000,linecolor=green]
\end{pspicture}}
```

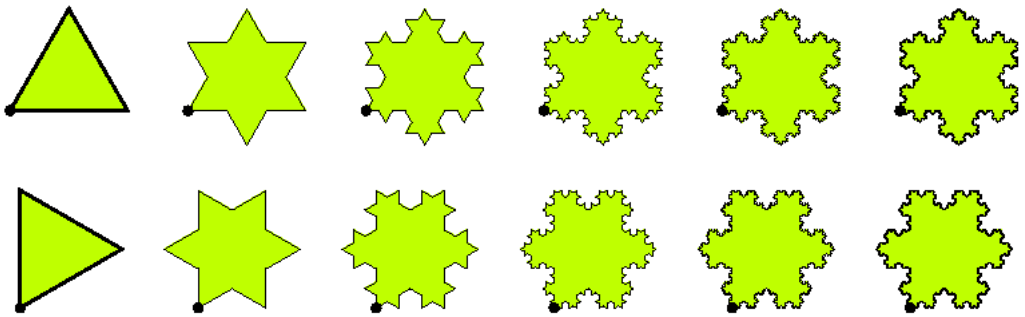
5 Koch flake

```
\psKochflake[settings](x,y)
```

The coordinates of the starting point are optional, if they are missing, then $(0, 0)$ is assumed. The origin is the lower left point of the flake, marked as red or black point in the following example:




```
\begin{pspicture}[showgrid=true](-2.4,-0.4)(5,5)
  \psKochflake[scale=10]
  \psdot[linecolor=red,dotstyle=*](0,0)
\end{pspicture}
```



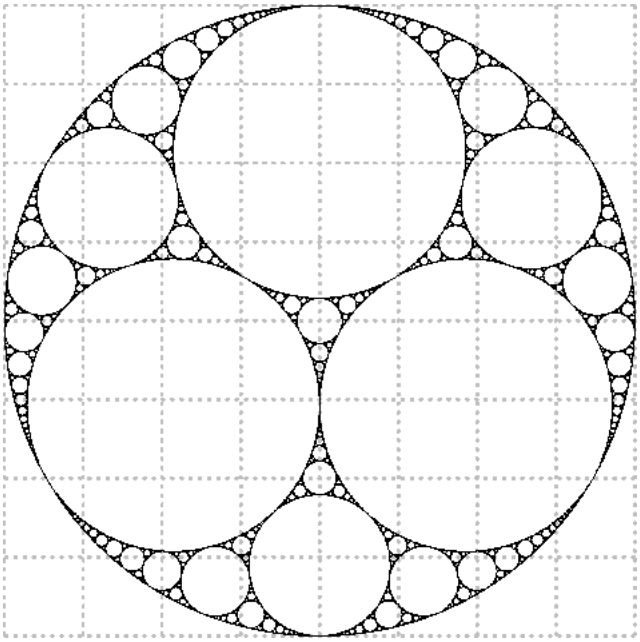
```
\begin{pspicture}(-0.4,-0.4)(12,4)
  \psset{fillcolor=lime,fillstyle=solid}
  \multido{\iA=0+1,\iB=0+2}{6}{%
    \psKochflake[angle=-30,scale=3,maxIter=\iA](\iB,2.5)\psdot*(\iB,2.5)
    \psKochflake[scale=3,maxIter=\iA](\iB,0)\psdot*(\iB,0)}
\end{pspicture}
```

Optional arguments are `scale`, `maxIter` (iteration depth) and `angle` for the first rotation angle.

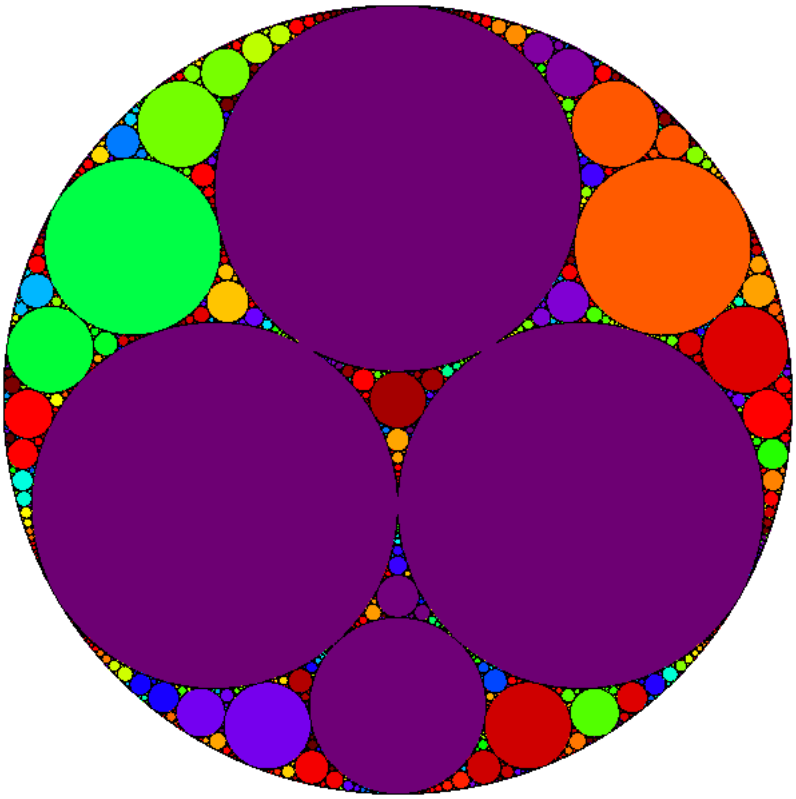
6 Apollonius circles

```
\psAppolonius[settings](x,y)
```

The coordinates of the starting point are optional, if they are missing, then $(0, 0)$ is assumed. The origin is the center of the circle:



```
\begin{pspicture}[showgrid=true](-4,-4)(4,4)
  \psAppolonius[Radius=4cm]
\end{pspicture}
```



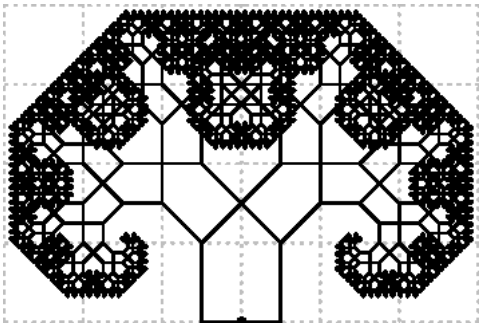
```
\begin{pspicture}(-5,-5)(5,5)
  \psAppolonius[Radius=5cm,Color]
\end{pspicture}
```

7 Trees

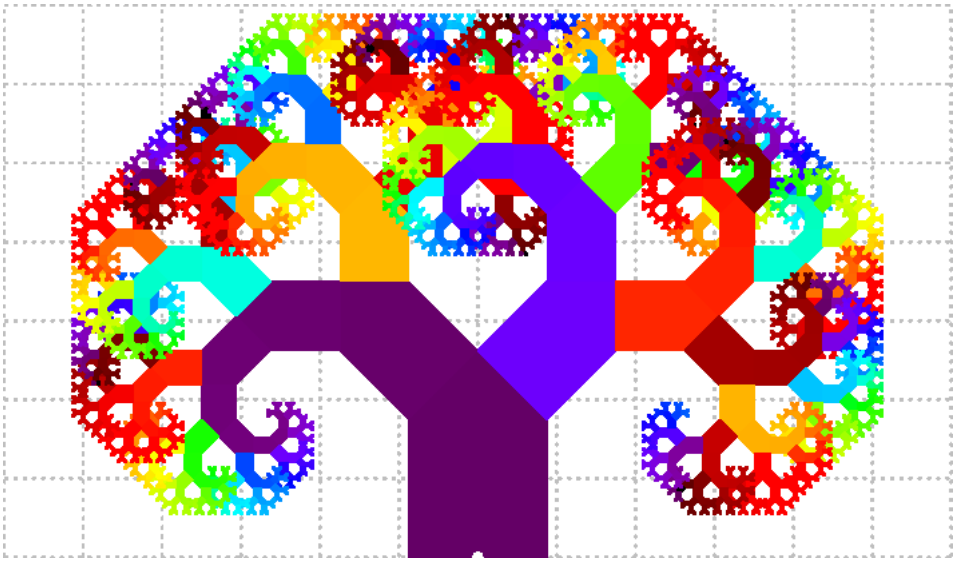
```
\psPTree[settings](x,y)
\psFArrow[settings](x,y){fraction}
```

The coordinates of the starting point are optional, if they are missing, then $(0, 0)$ is assumed. The origin is the center of the lower line, shown in the following examples by the dot. Special parameters are the width of the lower basic line for the tree and the height and angle for the arrow and for both the color option. The color step is given by `dIter` and the depth by `maxIter`. Valid optional arguments are

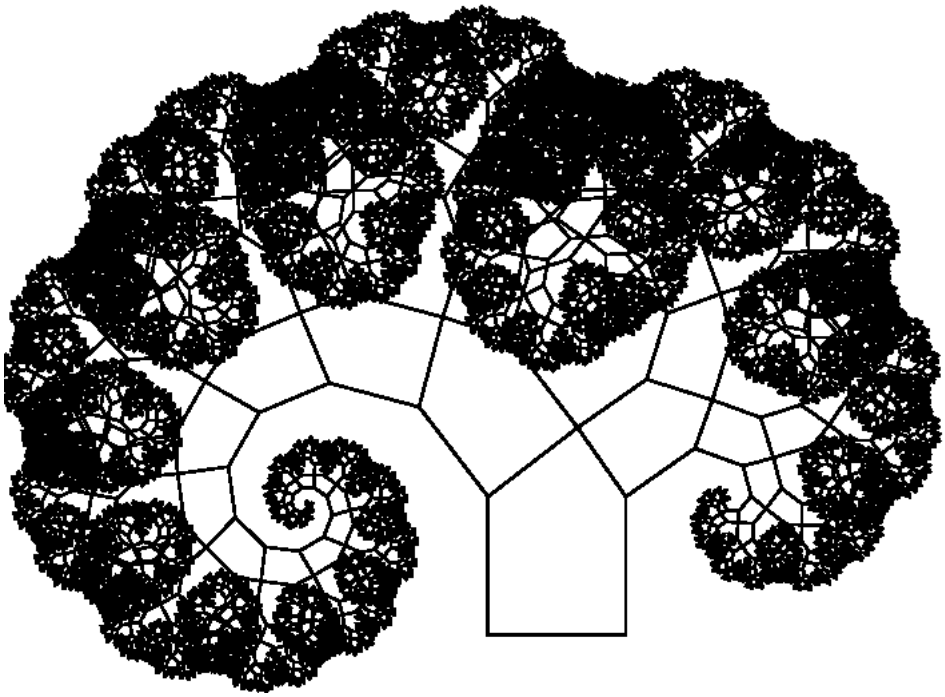
Name	*Meaning*	*default*
`xWidth`	first base width	1cm
`minWidth`	last base width	1pt
`c`	factor for unbalanced trees (0<c<1)	0.5
`Color`	colored tree	fasle



```
\begin{pspicture}[showgrid=true](-3,0)(3,4)
  \psPTree
  \psdot*(0,0)
\end{pspicture}
```

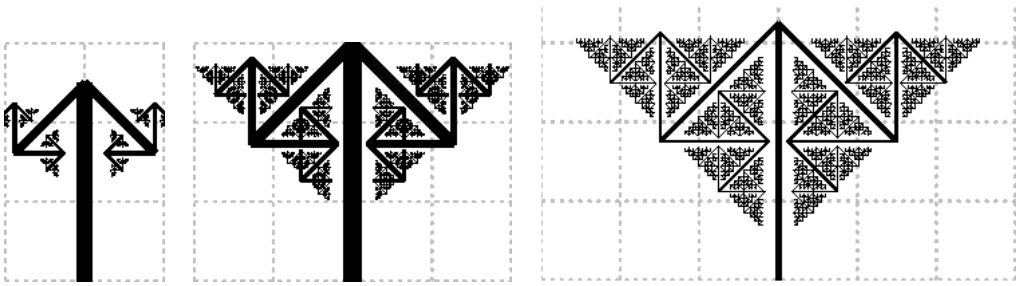


```
\begin{pspicture}[showgrid=true](-6,0)(6,7)
  \psPTree[xWidth=1.75cm,Color=true]
  \psdot*[linecolor=white](0,0)
\end{pspicture}
```



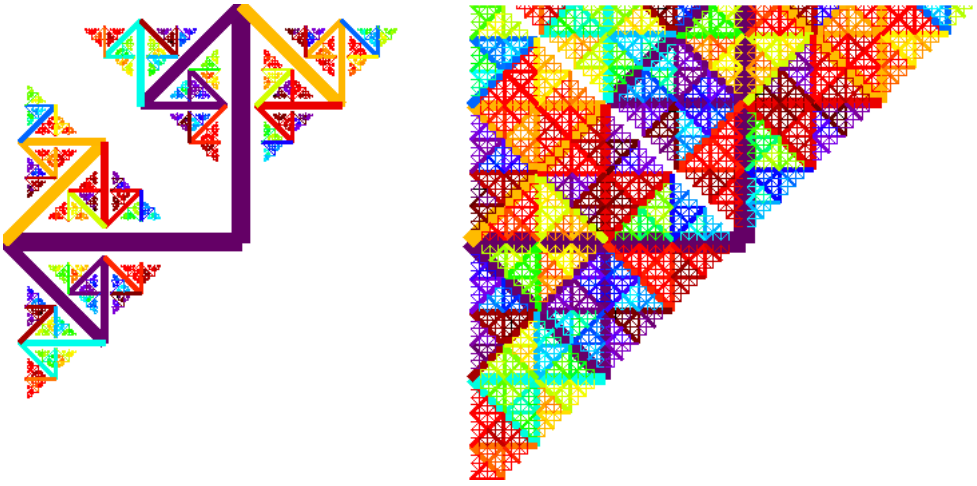
```
\begin{pspicture}(-7,-1)(6,8)
  \psPTree[xWidth=1.75cm,c=0.35]
\end{pspicture}
```

```
\begin{pspicture}(-5,-1)(7,8)
  \psPTree[xWidth=1.75cm,Color=true,c=0.65]
\end{pspicture}
```



```
\begin{pspicture}[showgrid=true](-1,0)(1,3)
  \psFArrow{0.5}
\end{pspicture}
\quad
\begin{pspicture}[showgrid=true](-2,0)(2,3)
  \psFArrow{0.6}
\end{pspicture}
\quad
\begin{pspicture*}[showgrid=true](-3,0)(3,3.5)
  \psFArrow[linewidth=3pt]{0.65}
\end{pspicture*}
```

```
\begin{pspicture}(-1,0)(1,3)
  \psFArrow[Color]{0.5}
\end{pspicture}
\quad
\begin{pspicture}(-2,0)(2,3)
  \psFArrow[Color]{0.6}
\end{pspicture}
\quad
\begin{pspicture*}(-3,0)(3,3.5)
  \psFArrow[Color]{0.65}
\end{pspicture*}
```



```
\begin{pspicture}(-3,-3)(2,3)
  \psFArrow[Color]{0.6}
  \psFArrow[angle=90,Color]{0.6}
\end{pspicture}
\quad
\begin{pspicture*}(-4,-3)(3,3)
  \psFArrow[Color]{0.7}
  \psFArrow[angle=90,Color]{0.7}
\end{pspicture*}
```

8 PDF output

pst-fractal is based on the popular pstricks package and writes pure PostScriptcode[3], so it is not possible to run TEX files with pdfLATEX when there are pstricks macros in the document. If you still need a PDF output use one of the following possibilities:

- package pdftricks.sty[6]
- the for Linux free available program VTeX/Lnx[1]
- build the PDF with ps2pdf (dvi→ps→pdf)
- use the pst-pdf package.[2]

You do not need to load pstricks.sty, it will be done by pst-fractal by default.

[1] http://www.micropress-inc.com/linux/
[2] http://www.ctan.org/CTAN/macros/latex/contrib/pst-pdf/

9 FAQ

- The fractal is not correct placed.

  Be sure that you view your output with a dvi viewer which can show PostScript code, like kdvi but not xdvi. It is better to run `dvips` and then view the ps-file with `gv`.

- Unknown PostScript command:

  Be sure that you have the "newest" `pstricks-add.tex` file

```
\def\fileversion{2.85}
\def\filedate{2007/04/01}
```

10 Credits

References

[1] Michel Goosens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voß. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 2007.

[2] Laura E. Jackson and Herbert Voß. Die Plot-Funktionen von `pst-plot`. *Die TeXnische Komödie*, 2/02:27–34, June 2002.

[3] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.

[4] Manuel Luque. *Vue en 3D*. http://members.aol.com/Mluque5130/vue3d16112002.zip, 2002.

[5] Herbert Voß. Die mathematischen Funktionen von Postscript. *Die TeXnische Komödie*, 1/02:40–47, March 2002.

[6] Herbert Voss. *PSTricks Support for pdf*. http://PSTricks.de/pdf/pdfoutput.phtml, 2002.

[7] Herbert Voß. *LaTeX in Mathematik und Naturwissenschaften*. Franzis-Verlag, Poing, 2006.

[8] Herbert Voß. *PSTricks – Grafik für TeX und LaTeX*. DANTE – Lehmanns, Heidelberg/Hamburg, 4. edition, 2007.

[9] Michael Wiedmann and Peter Karp. *References for TeX and Friends*. http://www.miwie.org/tex-refs/, 2003.

[10] Timothy Van Zandt. *PSTricks - PostScript macros for Generic TeX*. http://www.tug.org/application/PSTricks, 1993.