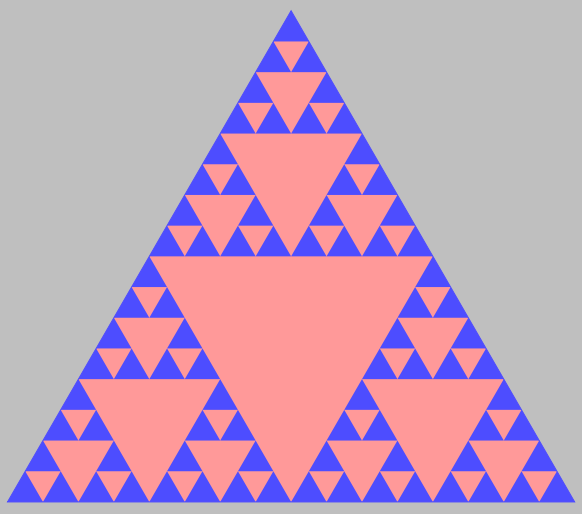
PSTricks

pst-fractal

Plotting fractals; v.0.01

February 10, 2010



Package author(s):
Herbert Voß

Contents

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.

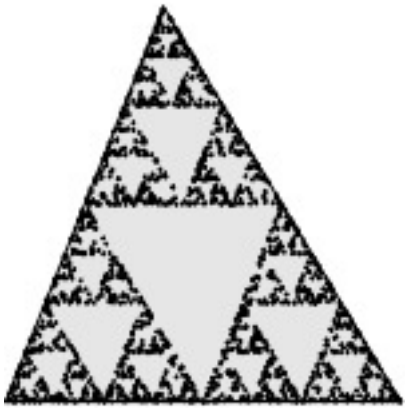
All images in this documentation were converted to the `.jpg` format to get a small pdf file size. When using the pdf format for the images the file size will be more than 20 MBytes. However, having a small file size will lead into a bad image resolution. Run the examples as single documents to see how it will be in high quality.

1 Sierpinski triangle

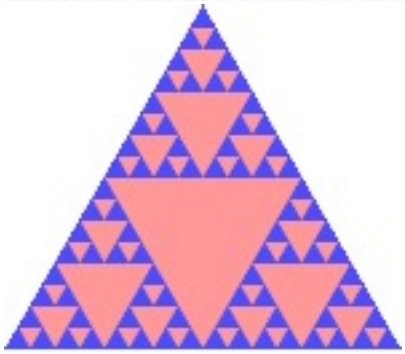
The triangle must be given by three mandatory arguments. Depending to the kind of arguments it is one of the two possible versions:

```
\psSier [Options] (x0,y0)(x1,y1)(x2,y2)
\psSier [Options] (x0,y0){Base}{Recursion}
```

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



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



```
\begin{pspicture}(5,5)
\psSier[linecolor=blue!70,
  fillcolor=red!40](0,0){5cm}{4}
\end{pspicture}
```

2 Julia and Mandelbrot sets

The syntax of the `\psfractal` macro is simple

```
\psfractal [Options] (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 $(0,0)$.

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

2.1 Julia sets

A Julia set is given with

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

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

```
\pspicture(-1,-1)(1,1)\psfractal\
    endpspicture
```

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

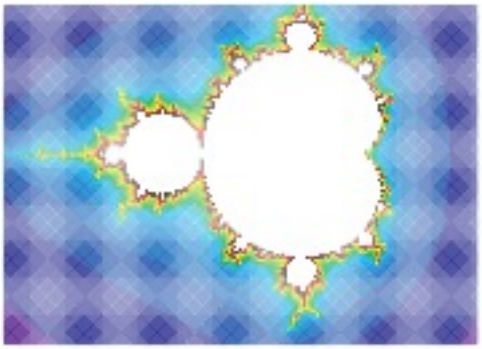
2.2 Mandelbrot sets

A Mandelbrot set is given with

$$z_{n+1}(x,y) = (z_n(x,y))^2 + C(x,y) \tag{4}$$
$$z_0 = (0;0) \tag{5}$$
$$C(x,y) = (x_0; y_0) \tag{6}$$

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

```
\pspicture(-1,-1)(1,1)
\psfractal[type=Mandel]
\endpspicture
```



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

2.3 The options

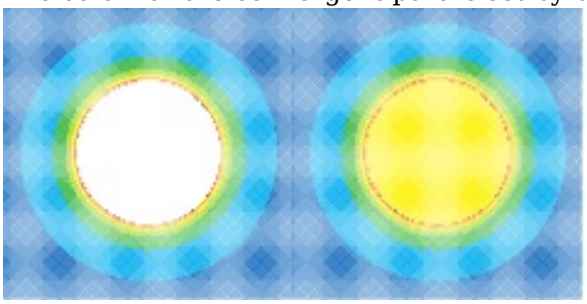
2.4 type

`txpe` can be of `Julia` (default) or `Mandel`.

```
\pspicture(-1,-1)(3,1)
\psfractal
\psfractal[type=Mandel]
\endpspicture
```

2.5 baseColor

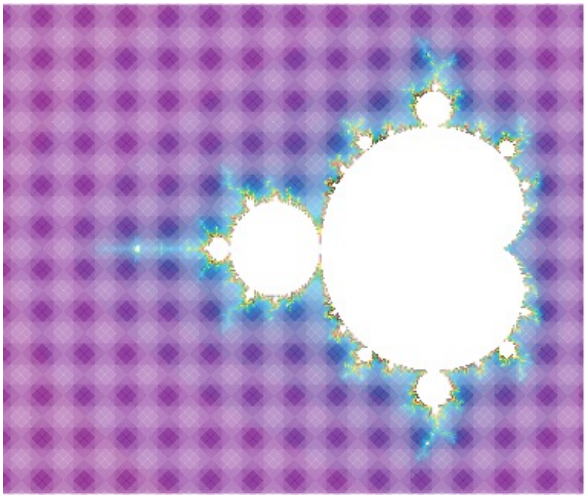
The color for the convergent part is set by `baseColor`.



```
\begin{postscript}
\psfractal[xWidth=4cm,yWidth=4cm,dIter
    =30](-2,-2)(2,2)
\psfractal[xWidth=4cm,yWidth=4cm,
    baseColor=yellow,dIter=30](-2,-2)
    (2,2)
\end{postscript}
```

2.6 xWidth and yWidth

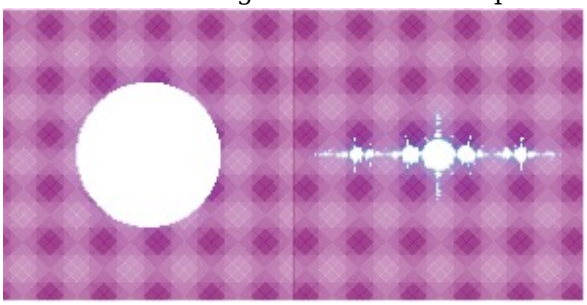
`xWidth` and `yWidth` define the physical width of the fractal.



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

2.7 cx and cy

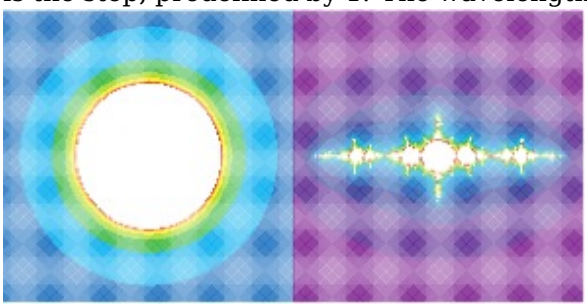
Define the starting value for the complex constant number C.



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

2.8 dIter

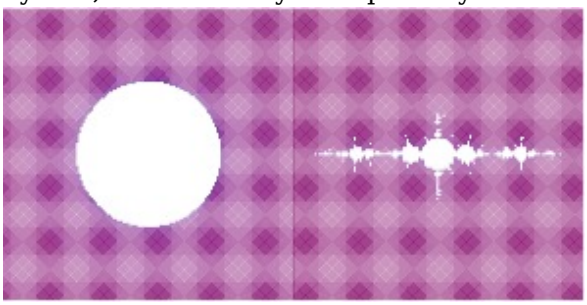
The color is set by wavelength to RGB conversion of the iteration number, where `dIter` is the step, predefined by 1. The wavelength is given by the value of `iter` added by 400.



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

2.9 maxIter

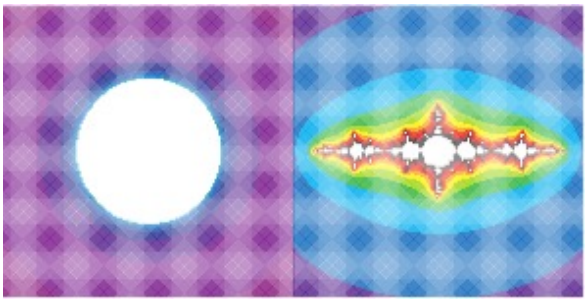
`maxIter` is the number of the maximum iteration until it leaves the loop. It is predefined by 255, but internally multiplied by `dIter`.



```
\begin{postscript}
\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)
\end{postscript}
```

2.10 maxRadius

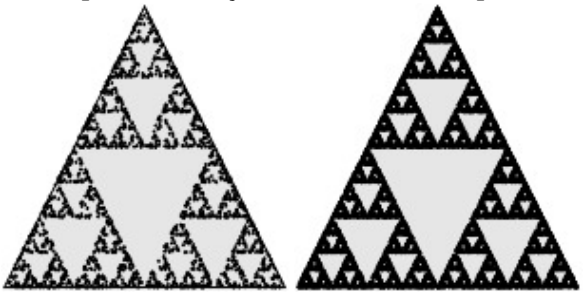
If the square of distance of z_n to the origin of the complex coordinate system is greater as `maxRadius` then the algorithm leaves the loop and sets the point. `maxRadius` should always be the square of the "'real"' value, it is preset by 100.



```
\begin{postscript}
\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)
\end{postscript}
```

2.11 plotpoints

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



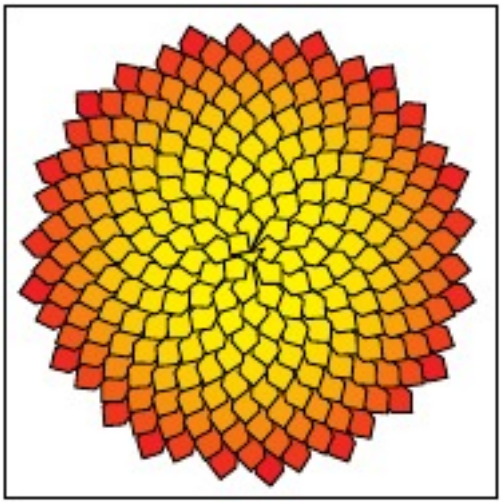
```
\begin{pspicture}(5,5)
 \psSier[plotpoints=10000](0,0)(2.5,5)
     (5,0)
\end{pspicture}
```

3 Phyllotaxis

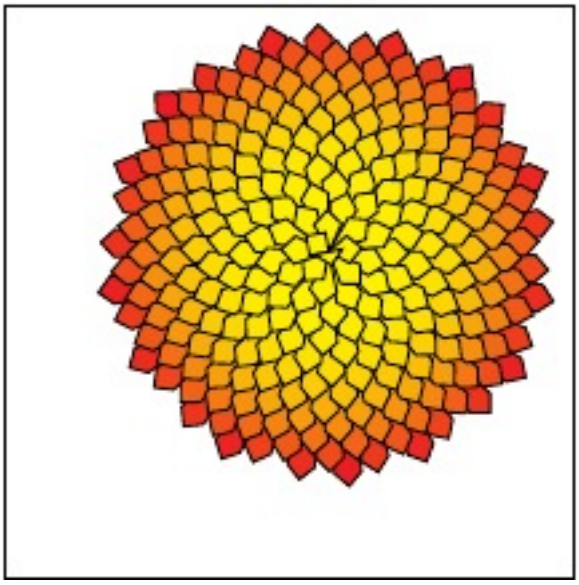
The beautiful arrangement of leaves in some plants, called phyllotaxis, obeys a number of subtle mathematical relationships. For instance, the florets in the head of a sunflower form two oppositely directed spirals: 55 of them clockwise and 34 counterclockwise. Surprisingly, these numbers are consecutive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

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

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

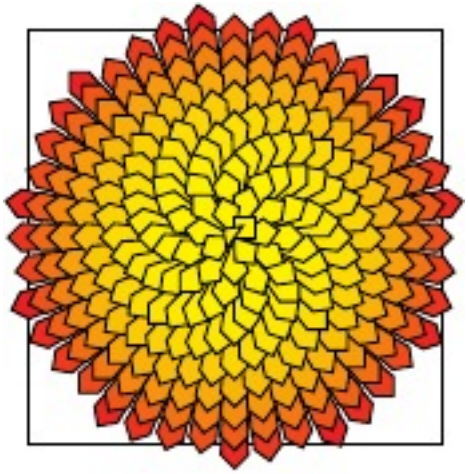


```
\begin{postscript}
\psframebox{\begin{pspicture}(-3,-3)(3,3)
 \psPhyllotaxis
\end{pspicture}}
\end{postscript}
```



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

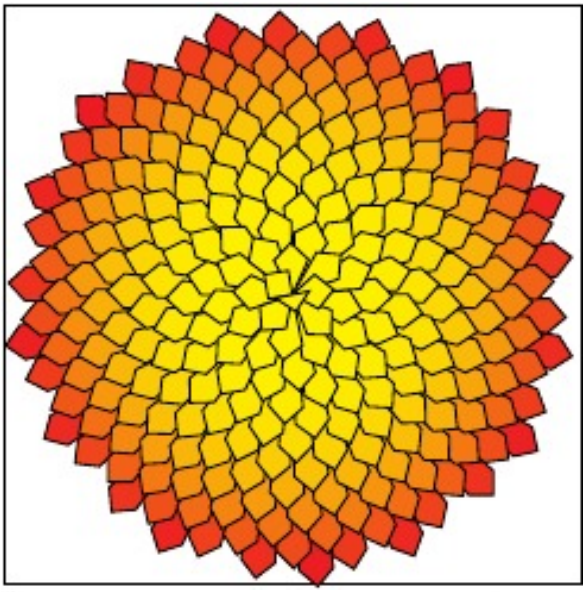
3.1 angle



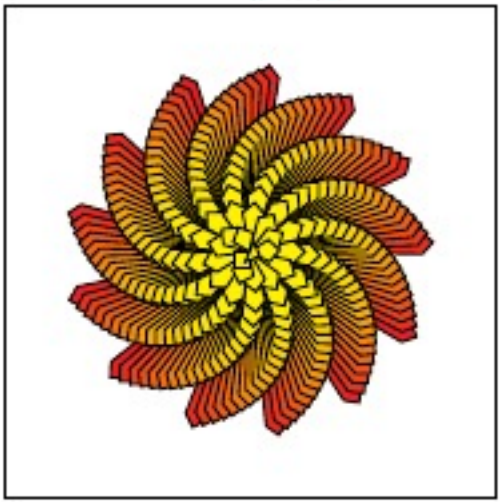
```
\begin{postscript}
\psframebox{\begin{pspicture}(-2.5,-2.5)
    (2.5,2.5)
 \psPhyllotaxis[angle=99]
\end{pspicture}}
\end{postscript}
```

3.2 c

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



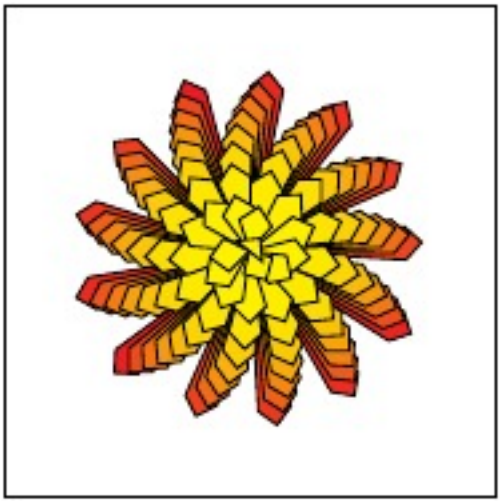
```
\begin{postscript}
\psframebox{\begin{pspicture}(8,8)
 \psPhyllotaxis[c=7](4,4)
\end{pspicture}}
\end{postscript}
```



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

3.3 maxIter

This is the number for the iterations.



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

4 Fern

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

The coordinates of the starting point are optional, if they are missing, then $(0,0)$ is assumed. The default `scale` is set to 10.

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

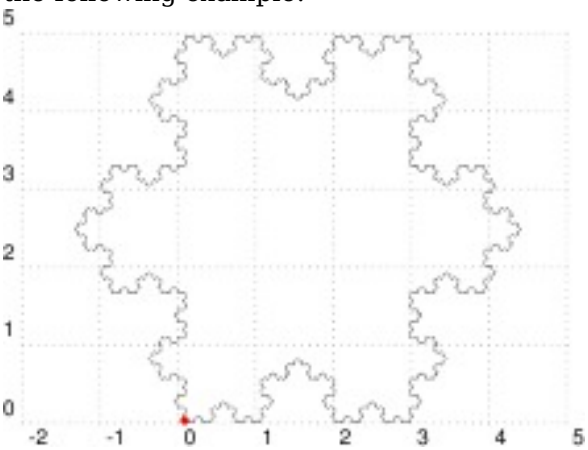
```
\begin{postscript}
\psframebox{\begin{pspicture}(-1,0)(2,5)
 \psFern(1,1)
\end{pspicture}}
\end{postscript}
```

```
\begin{postscript}
\psframebox{\begin{pspicture}(-3,0)(3,11)
  \psFern[scale=30,maxIter=100000,
      linecolor=green]
\end{pspicture}}
\end{postscript}
```

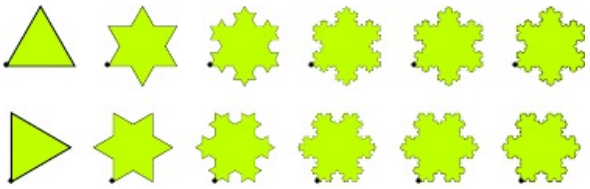
5 Koch flake

`\psKochflake[Options](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}
```



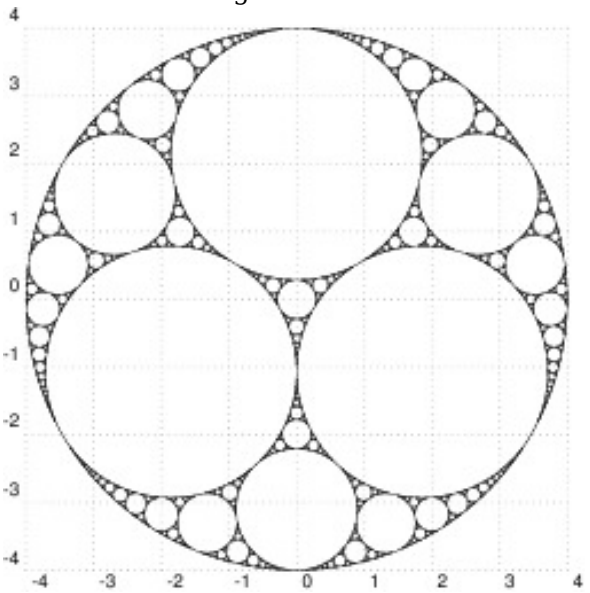
```
\begin{pspicture}(-0.4,-0.4)(12,4)
 \psset{fillcolor=lime,fillstyle=solid}
 \multido{\iA=0+1,\iB=0+2}{6}{%
  \psKochflake[angle=-30,scale=3,maxIter
      =\iA](\iB,2.5)\psdot*(\iB,2.5)
  \psKochflake[scale=3,maxIter=\iA](\iB
      ,0)\psdot*(\iB,0)}
\end{pspicture}
```

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

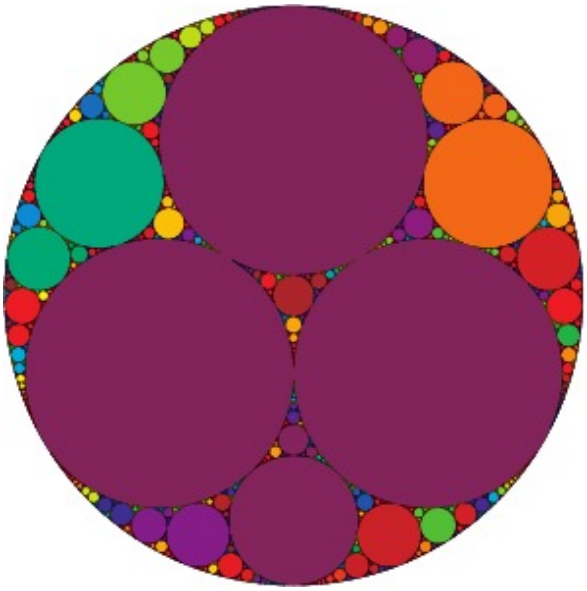
6 Apollonius circles

`\psAppolonius[Options]` (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}
```



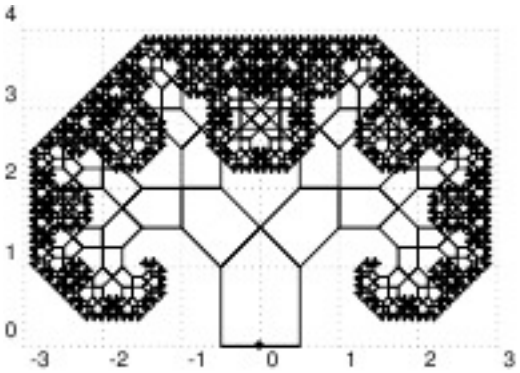
```
\begin{pspicture}(-5,-5)(5,5)
 \psAppolonius[Radius=5cm,Color]
\end{pspicture}
```

7 Trees

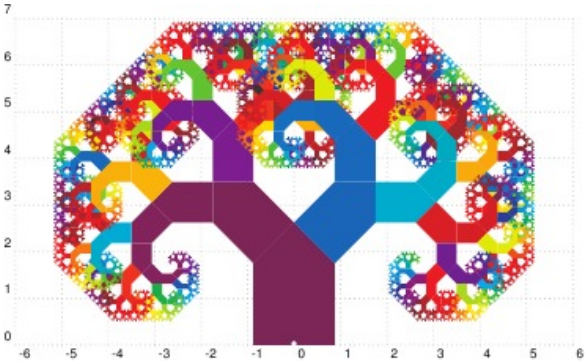
`\psPTree[Options](x,y) \psFArrow[Options](x,y){fraction}`

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

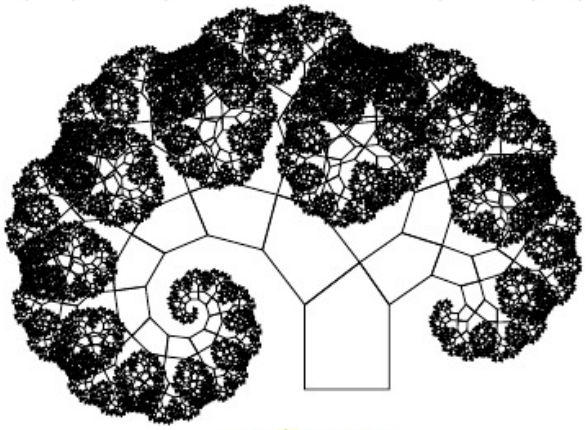
Name	*Meaning*	*default*
`xWidth`	first base width	1cm
`minWidth`	last base width	1pt
`c`	factor for unbalanced trees (0<c<1)	0.5
`Color`	colored tree	fasle




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

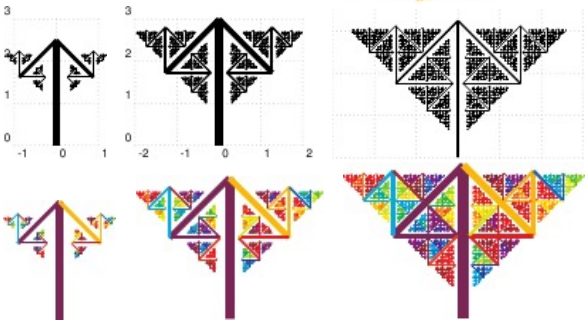


```
\begin{pspicture}[showgrid=true](-6,0)
    (6,7)
 \psPTree[xWidth=1.75cm,Color=true]
 \psdot*[linecolor=white](0,0)
\end{pspicture}
```



```
\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](-3,0)
    (3,3.5)
 \psFArrow[linewidth=3pt]{0.65}
\end{pspicture*}
```

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

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

8 List of all optional arguments for `pst-fractal`

Key	Type	Default
`xWidth`	`ordinary`	`1cm`
`yWidth`	`ordinary`	`1cm`
`type`	`ordinary`	`Julia`
`baseColor`	`ordinary`	`white`
`cx`	`ordinary`	`0`
`cy`	`ordinary`	`0`
`dIter`	`ordinary`	`1`
`maxIter`	`ordinary`	`255`
`maxRadius`	`ordinary`	`100`
`plotpoints`	`ordinary`	`2000`
`angle`	`ordinary`	`0`
`c`	`ordinary`	`5`
`minWidth`	`ordinary`	`1pt`
`scale`	`ordinary`	`1`
`Radius`	`ordinary`	`5cm`
`Color`	`boolean`	`true`

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.

Index