The current state of the PSTricks project, part II

Herbert Voß

Abstract

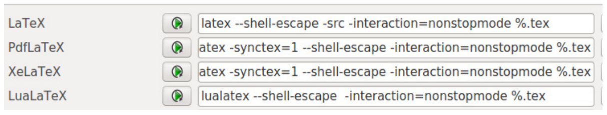
PSTricks is an abbreviation for PostScript Tricks, using the enormous graphical capabilities of the programming language PostScript, old as it may be. It is a so-called page description language, created in 1984 by Adobe Systems. In [4] we gave a report of what was possible with the different packages at that time. With this article we show what's new in the last seven years.

1 From PSTricks to PDF

The traditional route to create a PDF document with PostScript specials is still `latex` → `dvips` → `ps2pdf`. This sequence of commands can be put into a script or defined as a build-command for a GUI to make it only one mouse click. However, if one wants to use pdfLaTeX or XeLaTeX instead, this can also be done.

1.1 Using pdfLaTeX

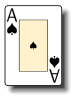
The package `auto-pst-pdf`[1] from Will Robertson works in the same way as `pst-pdf`[2], but doesn't need a script or the usual four runs by the user. Everything is done in one `pdflatex` run; this requires the `shell-escape` option to be enabled, to allow the running of external programs from within `pdflatex`. MiKTeX users have to enable the option `enable-write18`. This `shell-escape` option can be enabled in any GUI, for example TeXstudio:



```
LaTeX       latex --shell-escape -src -interaction=nonstopmode %.tex
PdfLaTeX    atex -synctex=1 --shell-escape -interaction=nonstopmode %.tex
XeLaTeX     atex -synctex=1 --shell-escape -interaction=nonstopmode %.tex
LuaLaTeX    lualatex --shell-escape  -interaction=nonstopmode %.tex
```

It is available from the GUI panel via Options → Configure TeXstudio → Commands.

`auto-pst-pdf` converts all `pspicture` environments into single images which replace the environment on the fly, in a second run. If there is no `pspicture` environment then the PSTricks-related code must be enclosed in a `postscript` environment.

For example: the poker card  internally uses the `pspicture` environment. With the `postscript` environment it will be converted by `auto-pst-pdf`, otherwise it will be missing in the PDF output. Here is the code for the above:

[1] ctan.org/pkg/auto-pst-pdf
[2] ctan.org/pkg/pst-pdf

```
... For example: the poker card
\begin{postscript}
\psset{unit=0.5}\crdAs
\end{postscript}
internally uses ...
```

The `postscript` environment can be used as its own paragraph or within a line.

1.2 Using XeLaTeX

XeLaTeX always creates an `.xdv` (extended DVI) file, which then is automatically converted into a PDF document. However, there are some cases where the program `xdvipdfmx` cannot create the correct PDF, e.g. nearly all examples from the old package `pst-light3d`.

1.3 LuaLaTeX

LuaLaTeX creates PDF directly, like pdfTeX. Therefore the first LuaLaTeX run needs to specify the option `--output-format=dvi`, which is not handled by the package `auto-pst-pdf`. However, using the package `dtk-extern` from `https://ctan.org/pkg/dtk` one can create any kind of TeX document inside a LuaLaTeX document.

2 Old packages with new macros

2.1 `pst-barcode`

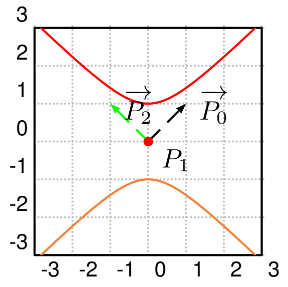
This package has existed for a long time, but now supports dozens of additional barcodes. Please refer to the documentation for the complete list. Here's an example of usage:

```
\begin{pspicture}(2in,0.5in)
\psbarcode{(00)030123456789012340|(02)130
  12345678909(37)24(10)1234567ABCDEFG}%
  {ccversion=c}{gs1-128composite}
\end{pspicture}
```

2.2 `pst-bezier` — Bézier curve with weighted points

A mass point is a weighted point $(P;\omega)$ with $\omega \neq 0$, or a vector $\left(\overrightarrow{P};0\right)$ with a weight equal to 0. A generic mass point is noted $(P;\omega)$. The package `pst-bezier` has a new macro `\psRQBCmasse`, which allows drawing a Bézier curve with such weighted points.

Conic	*Three weighted points*	*Points and vectors*
Parabola	$(P_0;1)$, $(P_1;\omega)$ $\left(P_2;\omega^2\right)$	$(P_0;1)$, $\left(\overrightarrow{P_1};0\right)$ $\left(\overrightarrow{P_2};0\right)$
Ellipse	$(P_0;1)$, $(P_1;\omega_1)$, $(P_2;\omega_2)$, $\omega_2 > \omega_1^2$	$(P_0;1)$, $\left(\overrightarrow{P_1};0\right)$ $(P_2;1)$
Hyperbola	$(P_0;1)$, $(P_1;\omega_1)$ $(P_2;\omega_2)$, $\omega_2 < \omega_1^2$	$(P_0;1)$, $\left(\overrightarrow{P_1};0\right)$ $(P_2;-1)$
		$\left(\overrightarrow{P_0};0\right)$, $(P_1;1)$ and $\left(\overrightarrow{P_2};0\right)$



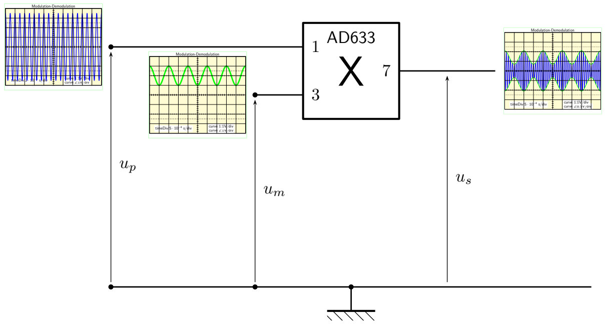
```
\begin{pspicture}[showgrid](-3,-3.4)(3,3)
\psclip{\psframe(-3,-3)(3,3)}
  \psRQBCmasse[linecolor=red,
    autoTrace](1,1)(0,0)(-1,1){0,1,0}
  \uput[r](P0){$\overrightarrow{P_0}$}
  \uput[r](0,-0.5){$P_1$}
  \uput[r](P2){$\overrightarrow{P_2}$}
  \psRQBCmasse[linecolor=orange,
    autoTrace=false](1,1)(0,0)(-1,1){0,-1,0}
\endpsclip
\end{pspicture}
```

3 The new packages

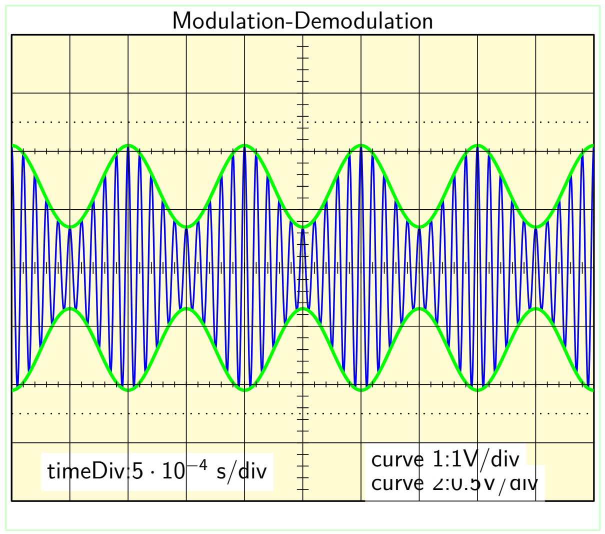
3.1 pst-am

`pst-am` allows the simulation of modulated and demodulated amplitude of radio waves. You can choose several possible parameters and plot the following curves:

- modulated signals;
- wave carrier;
- signal modulation;
- signal recovering;
- signal demodulation.



```
\def\noeud(#1){\qdisk(#1){1.5pt}}
\begin{pspicture}(-5,-1)(5,7)
\psline(-5,0)(5,0)\psline(-5,5)(-1,5)
\psline(-2,4)(-1,4)
\pnode(-5,5){E2}\noeud(E2)
\pnode(-2,4){E1}\noeud(E1)
\psline[arrowinset=0,arrowscale=2](1,4.5)(3,4.5)
\psframe[linewidth=1.5\pslinewidth](-1,3.5)(1,5.5)
\rput(0,4.5){\Huge\sffamily X}
\uput[270](0,5.5){\sffamily AD633}
\pnode(-5,0){M1}\pnode(-2,0){M2}
\pnode(0,0){O}\noeud(O)\noeud(M1)\noeud(M2)
\rput(O){\masse}
\uput[0](-1,5){1}
\uput[0](-1,4){3}
\uput[180](1,4.5){7}
\psset{linewidth=0.5\pslinewidth}
\psline{->}(-5,0.1)(-5,4.9)
\uput[0](-5,2.5){$u_p$}
\psline{->}(-2,0.1)(-2,3.9)
\uput[0](-2,2){$u_m$}
\psline{->}(2,0.1)(2,4.4)
\uput[0](2,2.25){$u_s$}
\psset[pst-am]{values=false}
\uput[0](3,4.5){\psscalebox{0.2}{\psAM[SignalModule,
  enveloppe,frequencePorteuse=1e4,
  voltDivY2=0.5,timeDiv=5e-4,linewidth=2\pslinewidth]}}
\uput[l](-2,4){\psscalebox{0.2}{\psAM[SignalModulant,
  timeDiv=5e-4,linewidth=5\pslinewidth]}}
\uput[l](-5,5){\psscalebox{0.2}{\psAM[SignalPorteuse,
  timeDiv=2e-4, frequencePorteuse=1e4,
  linewidth=5\pslinewidth]}}
\end{pspicture}
```



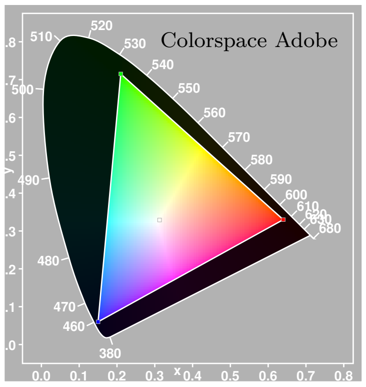
```
\psAM[SignalModule,enveloppe,frequencePorteuse=1e4,
      voltDivY2=0.5,timeDiv=5e-4]
```

3.2 pst-cie

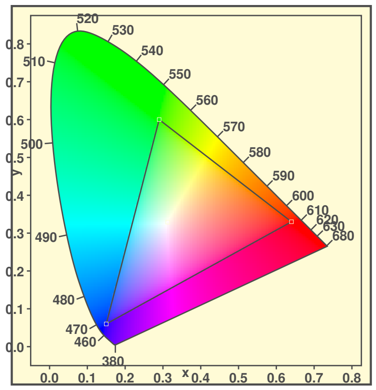
Using data (CIE XYZ 1931 and 1964) from the International Commission on Illumination (Commission internationale de l'éclairage) the package `pst-cie` proposes to represent the color table and/or the chromaticity diagram for different color spaces. The color spaces available are: `Adobe`, `CIE`, `ColorMatch`, `NTSC`, `Pal-Secam`, `ProPhoto`, `SMPTE` and `sRGB`.

It provides just one macro, which supports several optional arguments:

`\psChromaticityDiagram[`⟨*options*⟩`]`



```
\begin{pspicture}(-1,-1)(8.5,9.5)
\psChromaticityDiagram[datas=CIE1964,
  ColorSpace=Adobe,contrast=0.1,
  bgcolor=black!30]
\rput(5.5,8){\footnotesize Colorspace Adobe}
\end{pspicture}
```



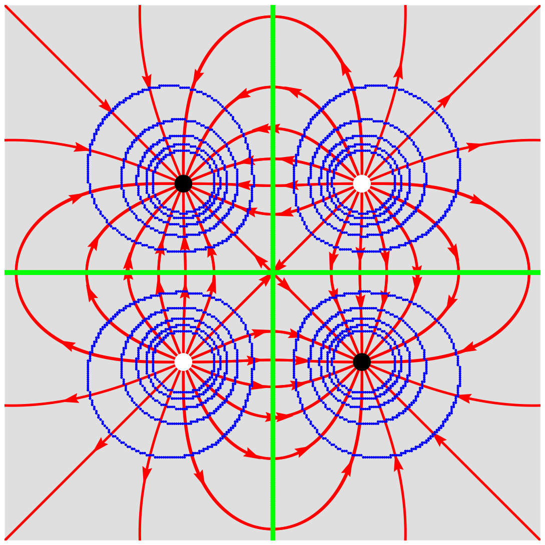
```
\begin{pspicture}(-1,-1)(8.5,9.5)
\psChromaticityDiagram[ColorSpace=Pal-Secam,
  bgcolor=yellow!100!black!20,
  textcolor=black!70]
\end{pspicture}
```

3.3 pst-electricfield

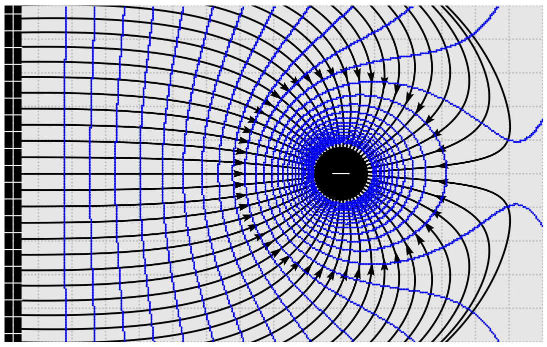
Equipotential surfaces and electric field lines can be drawn by using the package `pst-func` and the command `\psplotImp[options](x1,y1)(x2,y2)`. The Gauss theorem states that the electric flux across a closed surface S, defined by

$$\psi = \oiint_S \vec{D}\cdot\vec{u}_n \mathrm{d}S = Q \tag{1}$$

is equal to the real charge Q inside S. As a consequence, in places with no charge ($Q = 0$), the electric flux is a conserved quantity.



```
\begin{pspicture*}(-6,-6)(6,6)
\psframe*[linecolor=lightgray!50](-6,-6)(6,6)
%\psgrid[subgriddiv=0,gridcolor=gray,griddots=10]
\psElectricfield[Q={[-1 -2 2][1 2 2]
  [-1 2 -2][1 -2 -2]},linecolor=red]
\psEquipotential[Q={[-1 -2 2][1 2 2]
  [-1 2 -2][1 -2 -2]},
  linecolor=blue](-6.1,-6.1)(6.1,6.1)
\psEquipotential[Q={[-1 -2 2][1 2 2]
  [-1 2 -2][1 -2 -2]},linecolor=green,
  linewidth=2\pslinewidth,
  Vmax=0,Vmin=0](-6.1,-6.1)(6.1,6.1)
\end{pspicture*}
```



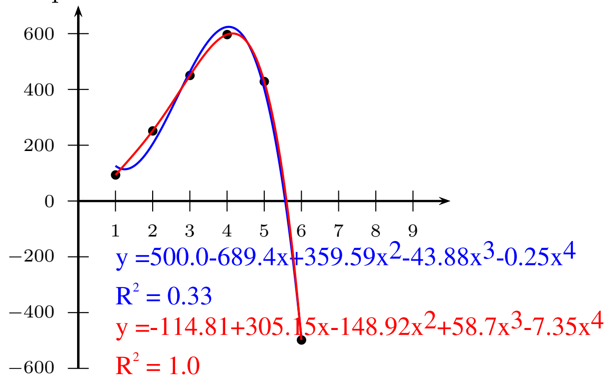
```
\begin{pspicture*}(-10,-5)(6,5)
\psframe*[linecolor=lightgray!40](-10,-5)(6,5)
\psgrid[subgriddiv=0,gridcolor=lightgray,griddots=10]
\psElectricfield[
  Q={[600 -60 0 false][-4 0 0] },
  N=50,points=500,runit=0.8]
\psEquipotential[
  Q={[600 -60 0 false][-4 0 0]},
  linecolor=blue,Vmax=100,Vmin=50,
  stepV=2](-10,-5)(6,5)
\psframe*(-10,-5)(-9.5,5)
\rput(0,0){\textcolor{white}{\large$-$}}
\multido{\rA=4.75+-0.5}{20}{%
  \rput(-9.75,\rA){\textcolor{white}{\large$+$}}}
\end{pspicture*}
```

3.4 pst-fit

Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points, possibly subject to constraints. The package `pst-fit` has many optional arguments to help achieve the desired interpolated curve. The following example shows six points of the polynomial

$$-109 + 294.53x - 142.94x^2 + 57.4x^3 - 7.26x^4$$

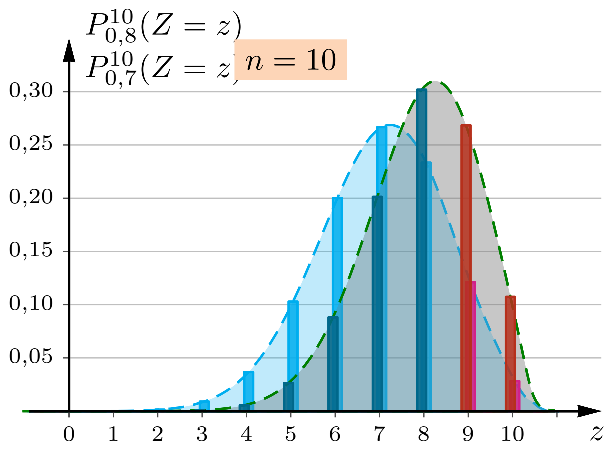
which are marked in the example as dots. The red and blue lines are two different solutions for a polynomial of 4th order. The internally calculated equations are plotted.



```
%Poly(-7.26*x^4+57.4*x^3-142.94*x^2+294.53*x-109)
\def\poly{1 93   2 251  3 450  4 597  5 428  6 -498}
\begin{pspicture}(-0.5,-6.5)(13,7.5)
\psset{yunit=0.0075}
\psaxes[arrows=->,Dx=1,Dy=200,
  labelFontSize=\scriptstyle,
  xsubticks=1,ysubticks=1](0,0)(0,-600)(10,700)
\listplot[plotstyle=dots]{\poly}
\listplot[valuewidth=20,
  decimals=2,EqPos=1 -200,
  plotstyle=GLLSR,PolyOrder=4,
  plotpoints=400,Yint=500,
  linecolor=blue]{\poly}
\listplot[linecolor=red,
  decimals=2,EqPos=1 -400,
  plotstyle=GLLSR,PolyOrder=4,
  plotpoints=400]{\poly}
\end{pspicture}
```

3.5 pst-func

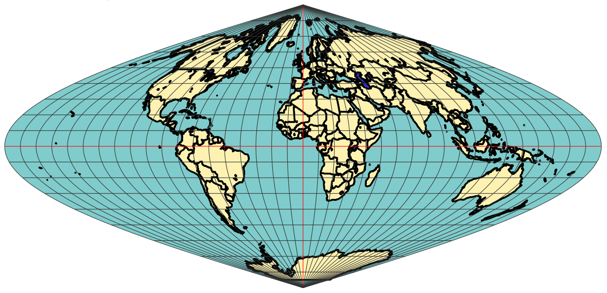
This package has some new macros for distributions, especially binomial distributions.



```
\begin{pspicture}(-.75,-1.8)(13.2,4.7)%
\psset{yunit=12cm}%
\psset{plotpoints=500,arrowscale=1.3,
  arrowinset=0.05,arrowlength=1.9,comma}
\psaxes[labelFontSize=\scriptstyle,xticksize=0 0,
 yticksize=0 12,tickcolor=gray!50,Dy=0.05,dy=0.05
]{-}(0,0)(-0.9,0)(10.8,0.34)
\uput[-90](11.9,0){$z$}
\uput[0](0,0.36){$P_{0,8}^{10}(Z=z)$}
\uput[0](0,0.32){$P_{0,7}^{10}(Z=z)$}
\rput(-0.05,0){\psBinomialC[linecolor=Green,
     fillstyle=solid,fillcolor=gray,opacity=0.25,
     plotstyle=curve,linestyle=dashed]{10}{0.8}}
\rput(0.05,0){\psBinomialC[linecolor=cyan,
     fillstyle=solid,fillcolor=cyan,opacity=0.25,
     plotstyle=curve,linestyle=dashed]{10}{0.7}%
  \psBinomial[markZeros,linecolor=cyan,
    fillstyle=solid,fillcolor=cyan,barwidth=0.2,
    opacity=0.85]{0,8,10}{0.7}
  \psBinomial[markZeros,linecolor=magenta,
    fillstyle=solid,fillcolor=magenta,barwidth=0.2,
    opacity=0.85]{9,10,10}{0.7}}
\rput(-0.05,0){%
  \psBinomialC[linecolor=Green,fillstyle=solid,
     fillcolor=gray,opacity=0.25,plotstyle=curve,
     linestyle=dashed]{10}{0.8}
  \psBinomial[markZeros,linecolor=DeepSkyBlue4,
    fillstyle=solid,fillcolor=DeepSkyBlue4,
    barwidth=0.2,opacity=0.85]{0,8,10}{0.8}
  \psBinomial[markZeros,linecolor=BrickRed,
    fillstyle=solid,fillcolor=BrickRed,barwidth=0.2,
    opacity=0.85]{9,10,10}{0.8}}
\psaxes[labels=none,xticksize=-2pt 0,yticksize=-2pt 0,
  tickcolor=black!70,Dy=0.05,dy=0.05\psyunit,Dx=1,
  dx=1\psxunit]{->}(0,0)(-0.9,0)(12,0.35)
\rput(5,0.33){\psframebox[fillstyle=solid,
  fillcolor=orange!30,
  linestyle=none]{$n=10$}}
\end{pspicture}
```

3.6 pst-geo

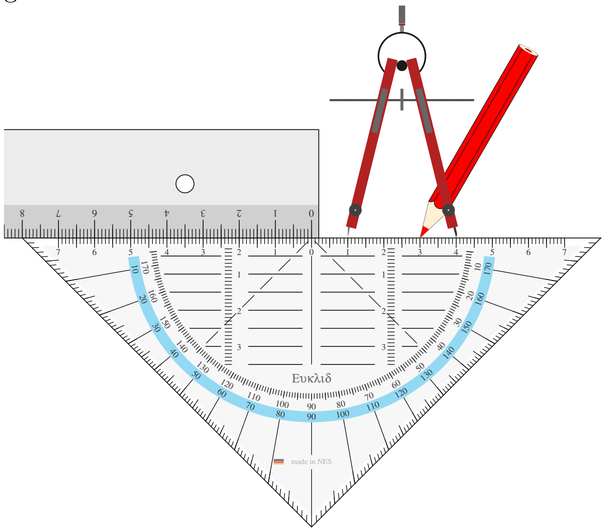
In the past the user had to load four different packages for the different geographical macros. After rearranging the code, there is now only one package: pst-geo with only one .sty file and one .pro file (PostScript code). The Sanson-Flamsted projection of the world is a *sinusoidal* projection which is a pseudo-cylindrical equal area-map:



```
\begin{pspicture}(-5,-5)(8,5)
\WorldMap[type=4]% Sanson-Flamsted
\end{pspicture}
```

3.7 pst-geometrictools

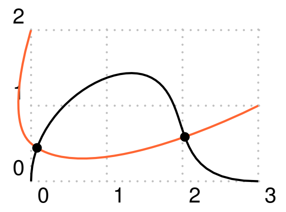
For mathematical worksheets in schools this package provides four macros for the tools which are used for geometrical constructions.



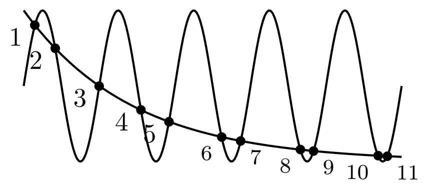
```
\begin{pspicture*}(-17,-17)(17,17)
\psProtractor{0}(0,0)% origin of the protractor
\psRuler{0}(0,0)% origin of the ruler
\psPencil{-30}(6,0)% origin of the pencil
\psCompass{3}(2,0)% origin of the compass
\end{pspicture*}
```

3.8 pst-intersec

This package calculates the intersection points of Bézier curves and/or arbitrary PostScript paths.



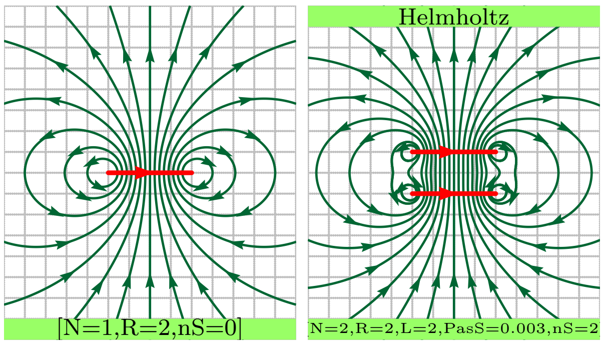
```
\begin{pspicture}(3,2)
  \pssavepath[linecolor=DOrange]%
     {MyPath}{\pscurve(0,2)(0,0.5)(3,1)}
  \pssavebezier{MyBez}(0,0)(0,1)(1,2)(3,2)(1,0)(3,0)
  \psintersect[showpoints]{MyPath}{MyBez}
\end{pspicture}
```



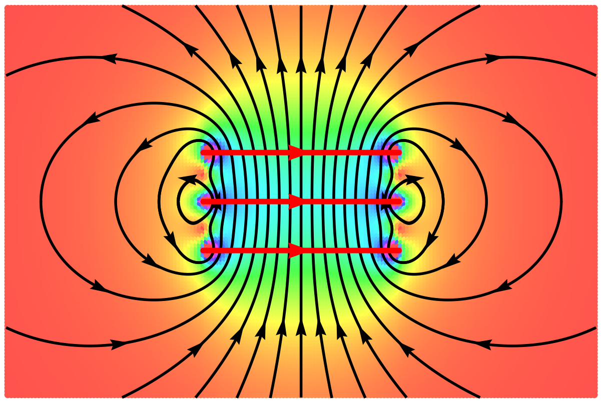
```
\begin{pspicture}(10,4.4)
\pssavepath{A}{%
  \psplot[plotpoints=200]%
     {0}{10}{x 180 mul sin 1 add 2 mul}}
\pssavepath{B}{%
  \psplot[plotpoints=50]%
     {0}{10}{2 x neg 0.5 mul exp 4 mul}}
\psintersect[name=C, showpoints]{A}{B}
\multido{\i=1+1}{5}{\uput[210](C\i){\i}}
\multido{\i=6+2,\ii=7+2}{3}{%
  \uput[225](C\i){\footnotesize\i}
  \uput[-45](C\ii){\footnotesize\ii}}
\end{pspicture}
```

3.9 pst-magneticfield

This package is similar to pst-electricfield. It supports the default two-dimensional magnetic field, density plots and a three-dimensional view of the field.



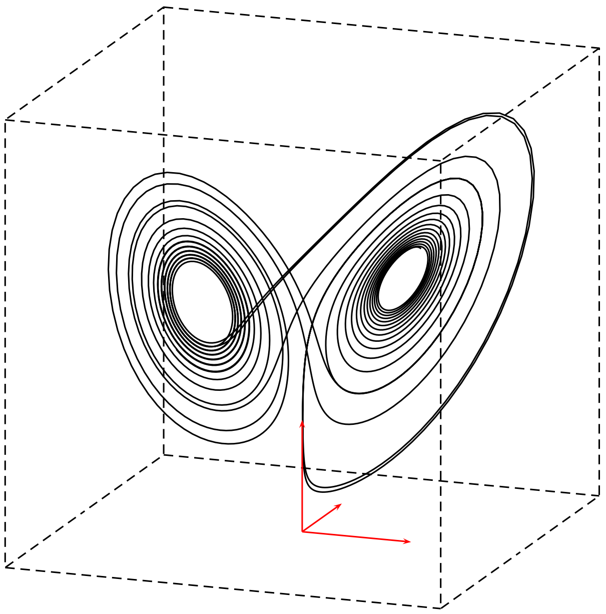
```
\psset{unit=0.5}
\begin{pspicture*}[showgrid](-7,-8)(7,8)
\psmagneticfield[linecolor={[HTML]{006633}},
  N=1,R=2,nS=0](-7,-8)(7,8)
\psframe*[linecolor={[HTML]{99FF66}}](-7,-8)(7,-7)
\rput(0,-7.5){\footnotesize[N=1,R=2,nS=0]}
\end{pspicture*}
\begin{pspicture*}[showgrid](-7,-8)(7,8)
\psmagneticfield[linecolor={[HTML]{006633}},
  N=2,R=2,L=2,PasS=0.003,nS=2](-7,-8)(7,8)
\psframe*[linecolor={[HTML]{99FF66}}](-7,7)(7,8)
\rput(0,7.5){\footnotesize Helmholtz}
\psframe*[linecolor={[HTML]{99FF66}}](-7,-8)(7,-7)
\rput(0,-7.5){\tiny[N=2,R=2,L=2,PasS=0.003,nS=2]}
\end{pspicture*}
```



```
\begin{pspicture}(-6,-4)(6,4)
\psmagneticfield[N=3,R=2,L=2,
  StreamDensityPlot](-6,-4)(6,4)
\end{pspicture}
```

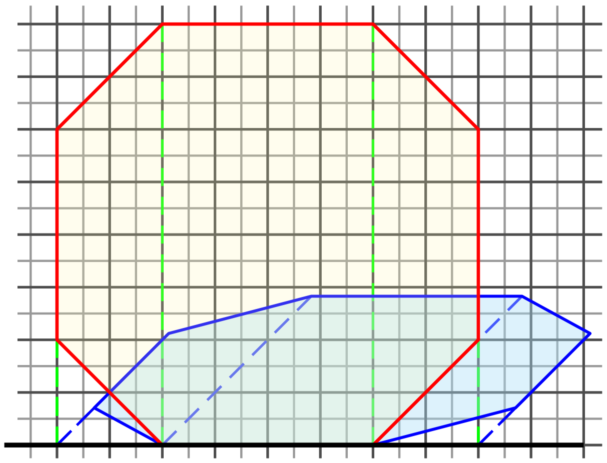
3.10 pst-ode

This package integrates differential equations using the Runge-Kutta-Fehlberg (RKF45) method with automatic step size control. Thus, the precision of the result does not depend on the number of plot points specified, as would be the case with the classical Runge-Kutta (RK4) method.



```
\begin{pspicture}(-8,-4)(6,12)
\pstVerb{ /alpha 10 def /beta 28 def
  /gamma 8 3 div def }%
\pstODEsolve[algebraic]{lorenzXYZ}%
  {0 1 2}{0}{25}{2501}{10 10 30}%
  { alpha*(x[1]-x[0])        |% x
    x[0]*(beta-x[2]) - x[1]  |% y
    x[0]*x[1] - gamma*x[2]    % z
  }
\psset{unit=0.17cm,Alpha=160,Beta=15}
\listplotThreeD{lorenzXYZ}% plot the ode-data
\psset{unit=0.425cm,linestyle=dashed}
\pstThreeDNode(0,0,0){O}\pstThreeDNode(0,0,5){Z}
\pstThreeDNode(5,0,0){X}\pstThreeDNode(0,5,0){Y}
\pstThreeDNode(-10,-10,0){A}\pstThreeDNode(-10,-10,20){B}
\pstThreeDNode(-10,10,20){C}\pstThreeDNode(-10,10,0){D}
\pstThreeDNode(10,-10,0){E}\pstThreeDNode(10,-10,20){F}
\pstThreeDNode(10,10,20){G}\pstThreeDNode(10,10,0){H}
\pspolygon(A)(B)(C)(D)\pspolygon(E)(F)(G)(H)
\psline(A)(E)\psline(B)(F)\psline(D)(H)\psline(C)(G)
\psset{linestyle=solid,linecolor=red}
\psline{->}(O)(X)\psline{->}(O)(Y)\psline{->}(O)(Z)
\end{pspicture}
```

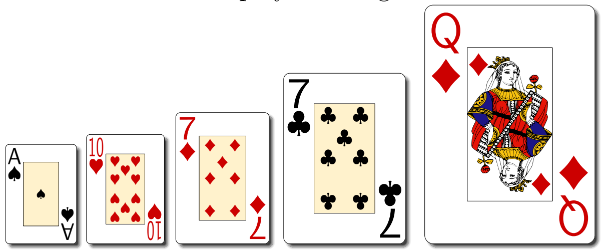
3.11 pst-perspective



```
\begin{pspicture}(0.5,-0.5)(11.5,8.5)
\begin{psclip}%
  {\psframe[linestyle=none](0.25,-0.25)(11.35,8.35)}
  \psgrid[subgriddiv=2,gridlabels=0,gridwidth=0.7pt,
     gridcolor=black!70,subgridwidth=0.6pt,
     subgridcolor=black!40](-1,-1)(13,10)
\end{psclip}
{\psset{translineA=true,translineB=true,
 linestyle=dashed,dash=5pt 3pt,linecolor=blue,
 linejoin=2}
%------ create octogon -------------
\pstransTS(3,0){A}{A'}\pstransTS(7,0){B}{B'}
\pstransTS(9,2){C}{C'}\pstransTS(9,6){D}{D'}
\pstransTS(7,8){E}{E'}\pstransTS(3,8){F}{F'}
\pstransTS(1,6){G}{G'}\pstransTS(1,2){H}{H'}}
\pspolygon[fillstyle=solid,fillcolor=cyan!30,
  opacity=0.4, linecolor=blue]%
  (A')(B')(C')(D')(E')(F')(G')(H')
\pspolygon[fillstyle=solid,fillcolor=yellow!40,
  opacity=0.2,linewidth=0.9pt,
  linecolor=red](A)(B)(C)(D)(E)(F)(G)(H)
\pcline[linewidth=1.3pt](0,0|O)(11,0|O)
\end{pspicture}
```

3.12 pst-poker

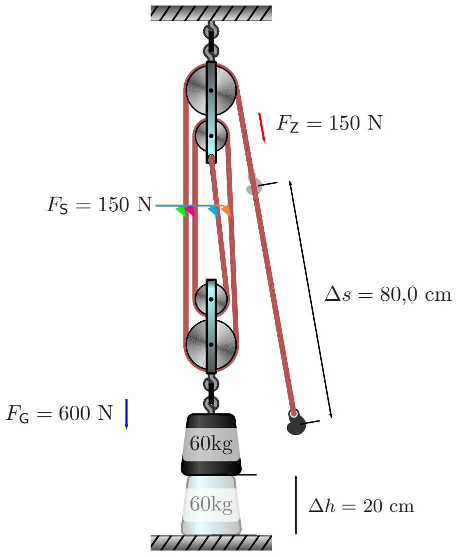
This is mostly a package for fun: it draws single or a group of poker cards. It can be displayed inline, like this: or displayed as big cards:



```
\crdAs
\psset{unit=1.1}\crdtenh \psset{unit=1.2}\crdsevd
\psset{unit=1.3}\crdsevc \psset{unit=1.4}\crdQd
```

3.13 pst-pulley

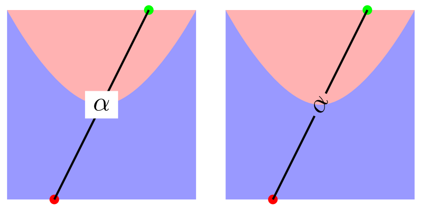
This package draws a nice view of a pulley, which may be of help to physics teachers in schools. There is only one macro which takes up to four optional parameters: `N=1...6` gives the number of wheels of the pulley; `M=...` gives the mass of the weight in `kg`; `h=...` gives the height of the weight in `cm` from the bottom.



```
\pspulleys[pulleyGrid=false,N=4,M=60,h=20]
```

3.14 pst-rputover

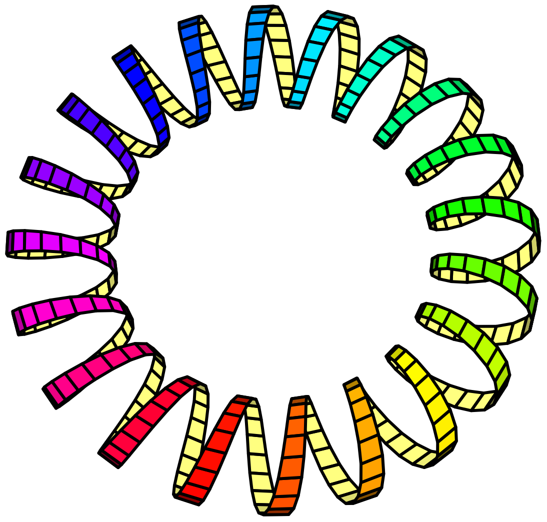
The macro `\ncput*` places an object at the middle of two given nodes. It is a general method to mark lines. With a background color, it doesn't look especially good, as can be seen in the left example with the default behaviour. The package `pst-rputover` has the same effect but without using its own background color.



```
\begin{pspicture}(2,2)
\psframe*[linecolor=blue!40](0,0)(2,2)
\pscurve*[linecolor=red!30](0,2)(1,1)(2,2)
\pnode(.5,0){A}\psdot[linecolor=red](A)
\pnode(1.5,2){B}\psdot[linecolor=green](B)
\pcline(A)(B)\ncput*{$\alpha$}
\end{pspicture}\quad
\begin{pspicture}(2,2)
\psframe*[linecolor=blue!40](0,0)(2,2)
\pscurve*[linecolor=red!30](0,2)(1,1)(2,2)
\pnode(.5,0){A}\psdot[linecolor=red](A)
\pnode(1.5,2){B}\psdot[linecolor=green](B)
\pclineover(A)(B){$\alpha$}
\end{pspicture}
```

3.15 pst-ruban

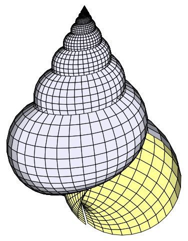
This package draws ribbons (instead of lines) on three dimensional objects. It is an extension of the package `pst-solides3d` allowing you to draw ribbons on certain solids of revolution: cylinder, torus, sphere, paraboloid and cone. The width of the ribbon, the number of turns, the color of the external face as well as that of the inner face can be optionally specified.



```
\psset{viewpoint=20 20 70 rtp2xyz,Decran=20,
       lightsrc=viewpoint,
       resolution=360,unit=0.6}
\begin{pspicture}(-5,-5)(5,5)
\psSpiralRing[incolor=yellow!50,r1=4,r0=1,hue=0 1]
\end{pspicture}
```

3.16 pst-shell

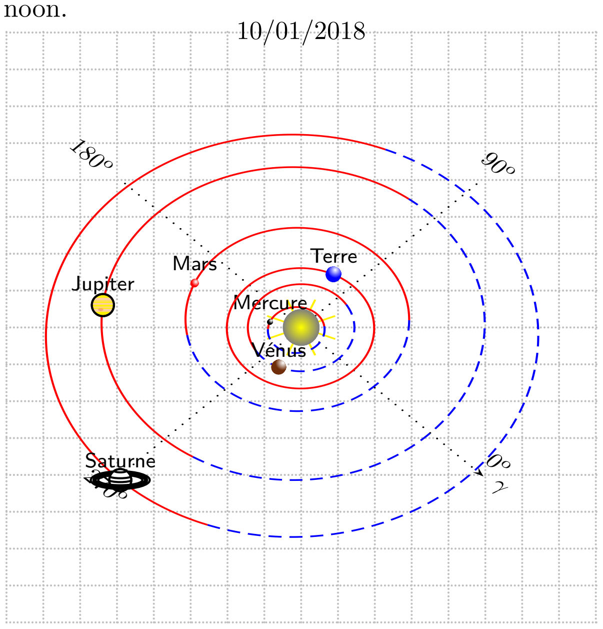
Geometric modeling of shellfish was carried out by Michael B. Cortie. In the "Digital Seashells" document he gives the parametric equations which are a function of 14 parameters, in order to allow modeling of a very large number of shells (`researchgate.net/publication/223141547_Digital_seashells`).



```
\begin{pspicture}(-3,-7)(3,0)
\psset{lightsrc=viewpoint,
  viewpoint=800 -90 20 rtp2xyz,Decran=50}
\psShell[style=Escalaria,base=0 -7200 -180 180,
  ngrid=720 30,incolor=yellow!40,
  fillcolor=yellow!20!blue!10,linewidth=0.01pt]
\end{pspicture}
```

3.17 `pst-solarsystem`

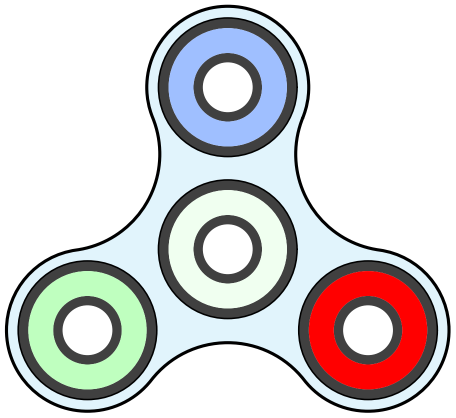
Position of the visible planets, projected on the plane of the ecliptic. The following example shows the solar system on Don Knuth's next magic birthday, at high noon.



```
\SolarSystem[Day=10,Month=01,Year=2018,Hour=12,
  Minute=0,Second=0,viewpoint=1 -1 2,solarValues=false]
```

3.18 `pst-spinner`

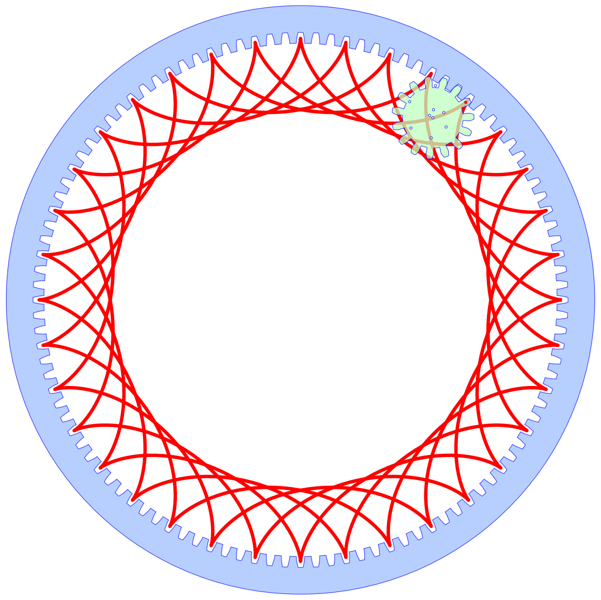
This package is just for fun. A fidget spinner is a type of stress-relieving toy.



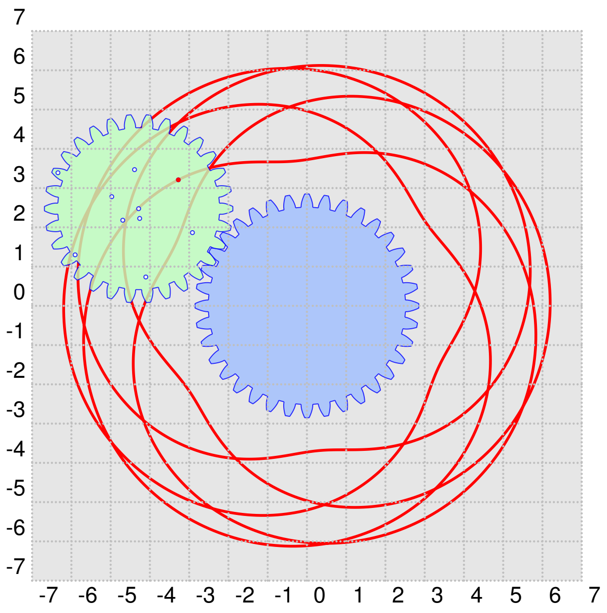
```
\begin{pspicture}(-4,-4)(5,4)
\psFidgetSpinner[fillcolor=cyan!10,linewidth=0.05,
  mask=false](0,0)
\end{pspicture}
```

3.19 `pst-spirograph`

A spirograph is a geometric drawing toy that produces mathematical roulette curves that are technically known as hypotrochoids and epitrochoids. It is possible to draw inner and outer curves.



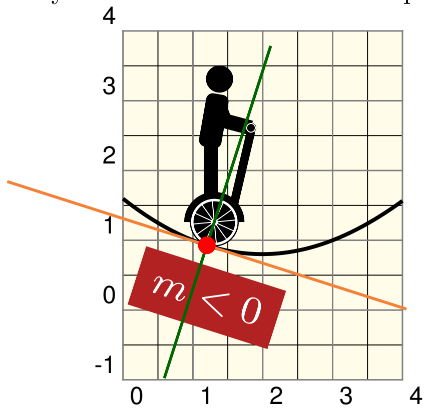
```
\begin{pspicture}(-7,-7)(7,7)
\psset{unit=0.5}
\psSpirograph[thetamax=-1800,Z1=108,Z2=15,
  m=0.2,linewidth=0.025,ap=10,
  fillstyle=solid,polarangle=54,
  linecolor=blue,holenumber=0,
  opacity=0.75]
\end{pspicture}
```



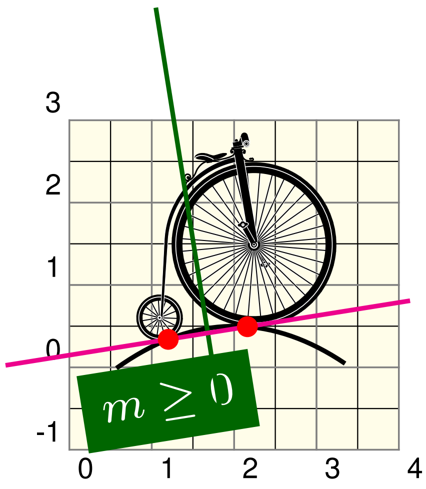
```
\begin{pspicture}[showgrid=top](-7,-7)(7,7)
\psframe*[linecolor=gray!20](-7,-7)(7,7)
\psSpirograph[thetamax=1800,Z1=36,Z2=30,m=0.15,
  linewidth=0.025,ap=20,inner=false,
  fillstyle=solid,polarangle=150,
  linecolor=blue,holenumber=4,opacity=0.8]
\end{pspicture}
```

3.20 pst-vehicle

This package provides slipping/rolling vehicles on curves of any kind of mathematical function, especially for animations in math and physics.



```
\def\FuncA{(x-3)*sin(0.2*(x-1))+1}
\begin{pspicture}(0,-1)(4,4)
\psframe*[linecolor=yellow!10](0,-1)(4,4)
\psgrid[style=quadrillage](0,-1)(4,4)
\psplot{0}{4}{\FuncA}
\psVehicle[vehicle=\Segway,
  style=segway]{0.25}{1.2}{\FuncA}
\end{pspicture}
```



```
\def\FuncA{-0.25*(x-2)^2+0.5}
\begin{pspicture}(0,-1)(4,3)
\psframe*[linecolor=yellow!10](0,-1)(4,3)
\psgrid[style=quadrillage](0,-1)(4,3)
\psplot[yMinValue=0]{0}{4}{\FuncA}
\psVehicle[vehicle=\HighWheeler]%
  {0.25}{1.2}{\FuncA}
\end{pspicture}
```

4 Summary

The Turing-complete PostScript programming language is old in computer terms, but provides many nice and useful graphical features. More information and many examples of PSTricks can be found on the following websites:

- http://pstricks.tug.org
- http://pstricks.blogspot.de

References

[1] Bill Casselman. *Mathematical Illustrations — A manual of geometry and PostScript.* Cambridge University Press, Cambridge, first edition, 2005.

[2] Denis Girou. Présentation de PSTricks. *Cahier GUTenberg*, 16:21–70, February 1994.

[3] Frank Mittelbach, Michel Goossens, Sebastian Rahtz, Denis Roegel, and Herbert Voß. *The LaTeX Graphics Companion.* Addison-Wesley, Boston, 2nd edition, 2006.

[4] Herbert Voß. The current state of the PSTricks project. *TUGboat*, 31(1):36–49, 2010. tug.org/TUGboat/tb31-1/tb97voss.pdf.

[5] Timothy Van Zandt and Denis Girou. Inside PSTricks. *TUGboat*, 15(3):239–248, September 1994. tug.org/TUGboat/tb15-3/tb44tvz.pdf.

◇ Herbert Voß
Herbert.Voss (at) fu-berlin dot de