PSTricks - 2008
new macros and bugfixes for the basic packages \pstricks, \pst-plot, \pst-tree, and \pst-node

Herbert Voß*

January 22, 2008

*Herbert.Voss@pstricks.de

Contents

Part I
pstricks – package

1 pstricks.sty

- Loading the package `pstricks` by LaTeX will now write a message into the file list of file version and date for the file `pstricks.pro`.
- A frequently done error is choosing a file name for the document, which is already a name of one PSTricks package, e. g. `pstricks.tex`. The error message in the log file was not really helpful. There is now an extended message:

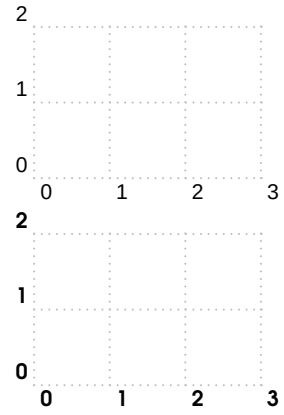
```
! LaTeX Error: Two \documentclass or \documentstyle commands
               or 'pstricks.tex' is a forbidden name for your document,
               it is already a name of a PSTricks package!
               Choose another name for your document!
```

2 pstricks.tex (1.20– 2008/01/01)

2.1 Option gridfont

By default the grid labels were printed always in Helvetica. With the new option `gridfont` one can define another PostScript Font. Available are

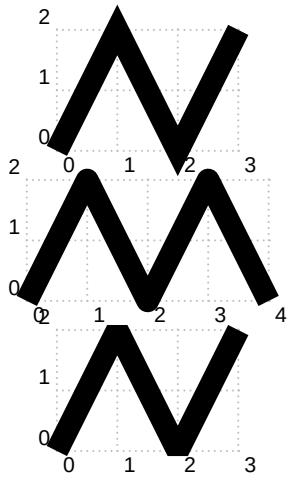
`Helvetica (default) - Helvetica-Narrow - Times-Roman - Courier - AvantGard -NewCenturySchlbk - Palatino-Roman - Bookman-Demi - Zapf-Dingbats - Symbol`




```
\usepackage{pstricks}
\begin{pspicture}[showgrid=true](3,2)
\end{pspicture}\\[20pt]
\begin{pspicture}(3,2)
  \psgrid[style=gridstyle,gridfont=
    AvantGard-Demi]
\end{pspicture}
```

2.2 `linejoin`

Connecting lines can be done in several ways and is controlled on PS level by the `setlinejoin` command. With this version of PSTricks it is possible to controll this by an optional argument, called `linejoin`. It is preset by 0 and can take values of 0,1,2. Other values jave no effect.




```
\psset{linewidth=3mm,unit=0.8}
\begin{pspicture}[showgrid=true](3,2)
  \psline(0,0)(1,2)(2,0)(3,2)
\end{pspicture}\\[10pt]
\begin{pspicture}[showgrid=true](4,2)
  \psline[linejoin=1](0,0)(1,2)(2,0)(3,2)
    (4,0)%
\end{pspicture}\\[10pt]
\begin{pspicture}[showgrid=true](3,2)
  \psline[linejoin=2](0,0)(1,2)(2,0)(3,2)%
\end{pspicture}
```

2.3 setlinecap

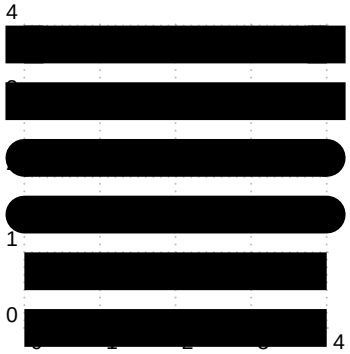
The value of `linecap` determines how the line ends are drawn:

0 lines are cut (default)

1 lines are ended by a filled semicircle of radius 0.5·`\pslinewidth`

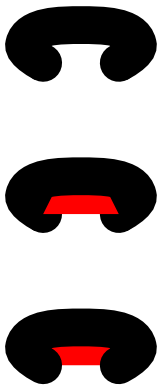
2 lines are ended by a filled half square of radius 0.5·`\pslinewidth`

The following example shows that using `linecap` for lines is the same than using the arrow option.




```
\begin{pspicture}[showgrid=true](4,4)%
  \psset{linewidth=5mm}
  \psline[arrows=C-C](0,3.75)(4,3.75)
  \psline[linecap=2](0,3)(4,3)
  \psline[arrows=c-c](0,2.25)(4,2.25)
  \psline[linecap=1](0,1.5)(4,1.5)
  \psline[arrows=-](0,0.75)(4,0.75)
  \psline(4,0)
\end{pspicture}
```

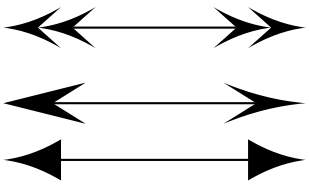
Using this optional argument makes only sense in some special cases, because it is the same as the arrow type `c-c`. But the arrows are not part of the current path and filling an open curve with the `linecap` option is different to a curve using the `c-c` arrow.



```
\psset{unit=5cm,linewidth=5mm}
\begin{pspicture}(-0.2,-0.6)(0.2,0.5)%
\def\curve{\pscurve(-.1,.1)(-.15,.15)(0,.2)(.15,.15)(.1,.1)}
\rput(0,.2){\psset{arrows=c-c}\curve}
\rput(0,-.2){%
  \psset{fillstyle=solid,fillcolor=red,arrows=c-c}
  \curve}
\rput(0,-.6){%
  \psset{fillstyle=solid,fillcolor=red,linecap=1}
  \curve}
\end{pspicture}
```

2.4 New arrowtype D> and D>D>

All arrows are drawn as polygons. The new arrow type D> or <D for the other way round, draws it lines as bezier curves, which looks nicer for big arrows.



```
\psset{arrowscale=5}
\begin{pspicture}(4,2)
\psline{<D<D-D>D>}(0,2)(4,2)
\psline[arrows=<-D>,arrowlength=2](0,1)(4,1)
\psline[arrowinset=0]{<D-D>}(0,0.25)(4,0.25)
\end{pspicture}
```

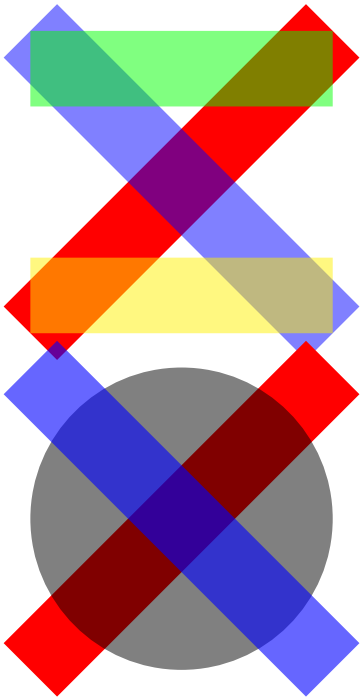
2.5 Transparent colors

The package `pstricks-add` already defined a fillstyle for transparency colors by using the GhostScripts blendmode. It now moves into the main `PSTricks` package, together with another possibility for creating transparent colors. Transparency is only seen with the PDF output (version 1.4 or greater), nearly all PostScript viewer cannot show transparencies.

Loading the `pstricks` package with the option `vtex`, disables the transparency effekts and everything works as before.

2.5.1 Options `strokeopacity` and `opacity`

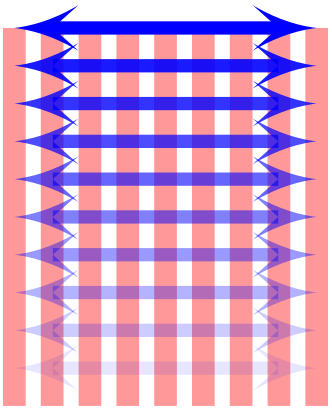
For the existing fill style `solid` the new option `opacity` can be used to get also transparent colors. It is predefined by 1 (0 . . . 1), which is the old behaviour, no transpareny. The option is valid only for PostScripts fill commands. Lines and curves can be trabsparent with setting the option `strokeopacity`, which can have a different value as for the `opacity` option.



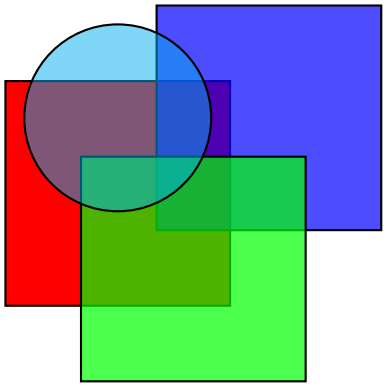
```
\begin{pspicture}[linewidth=1cm](4,4)
  \psline[linecolor=red](0,0)(4,4)
  \psline[linecolor=blue,strokeopacity=0.5](0,4)(4,0)
  \psline[linecolor=green,strokeopacity=0.5](0,3.5)
    (4,3.5)
  \psline[linecolor=yellow,strokeopacity=0.5](0,0.5)
    (4,0.5)
\end{pspicture}
```



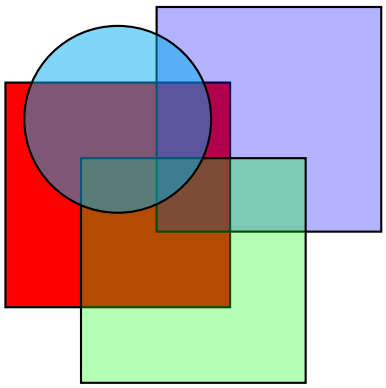
```
\begin{pspicture}[linewidth=1cm](4,4)
\psline[linecolor=red](0,0)(4,4)
\pscircle*[opacity=0.5](2,2){2}
\psline[linecolor=blue,strokeopacity=0.6](0,4)(4,0)
\end{pspicture}
```



```
\begin{pspicture}[linewidth=3mm](4,5.5)
  \multido{\rA=0.0+0.5}{9}{%
    \psline[linecolor=red!40](\rA,0)(\rA,5)}
  \multido{\rA=0.0+0.5,\rB=0.0+0.1}{11}{%
    \psline[arrows=<D-D>,linecolor=blue,
      linewidth=5pt,arrowscale=1.5,
       strokeopacity=\rB](0,\rA)(4,\rA)}
\end{pspicture}
```



```
\begin{pspicture}(5,5)
  \psset{fillstyle=solid}
  \psframe[fillcolor=red](0,1)(3,4)
  \psframe[fillcolor=blue,opacity=0.7](2,2)(5,5)
  \psframe[fillcolor=green,opacity=0.7](1,0)(4,3)
  \pscircle[fillcolor=cyan,
    opacity=0.5](1.5,3.5){1.25}
\end{pspicture}
```



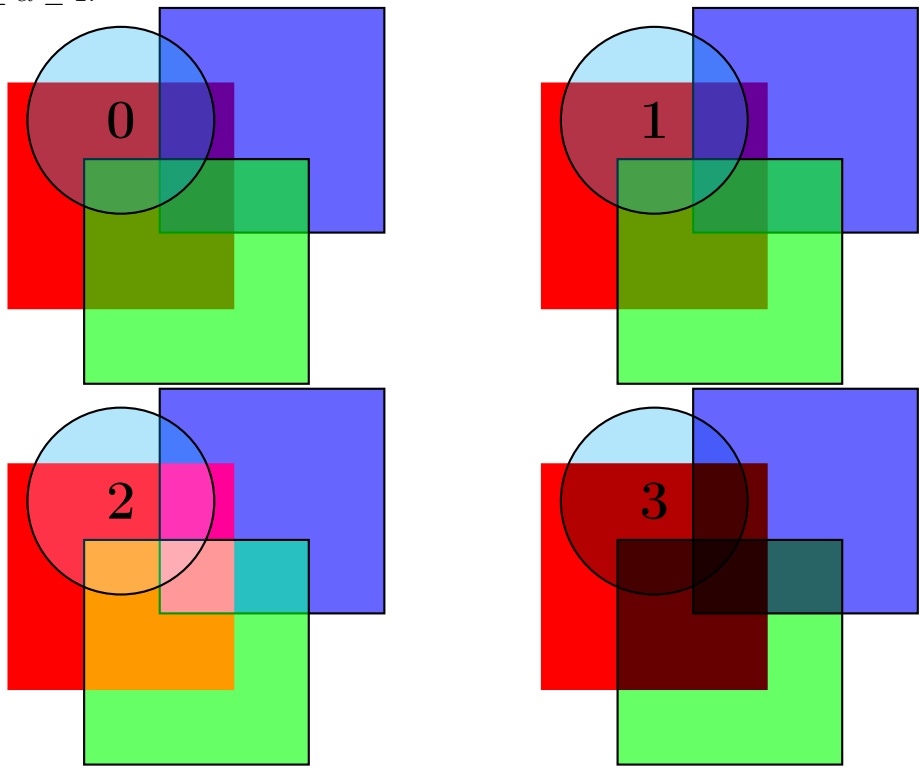
```
\begin{pspicture}(5,5)
  \psset{fillstyle=solid}
  \psframe[fillcolor=red](0,1)(3,4)
  \psframe[fillcolor=blue,opacity=0.3](2,2)(5,5)
  \psframe[fillcolor=green,opacity=0.3](1,0)(4,3)
  \pscircle[fillcolor=cyan,
    opacity=0.5](1.5,3.5){1.25}
\end{pspicture}
```

2.5.2 Fill style `shape`

There is now one more fill style for transparent colors: `shape` with using the `shapealpha` value and one of the possible blendmodes:

```
/Normal     ->0
/Compatible ->1
/Screen     ->2
/Multiply   ->3
```

The fill style `solid` uses GhostScripts `.setopacityalpha` function and the new style `shape` the `blendmode` together with `.setshapealpha`. `shapealpha` is predefined with 0.6 and both alpha values can be chosen from the range $0 \leq \alpha \leq 1$.




```
\begin{pspicture}(5,5)% default blendmode
  \psframe*[linecolor=red](0,1)(3,4)
  \psframe[fillcolor=blue,fillstyle=shape](2,2)(5,5)
  \psframe[fillcolor=green,fillstyle=shape](1,0)(4,3)
  \pscircle[fillcolor=cyan,fillstyle=shape,
    shapealpha=0.3](1.5,3.5){1.25}
  \rput(1.5,3.5){\huge\textbf{0}}
```

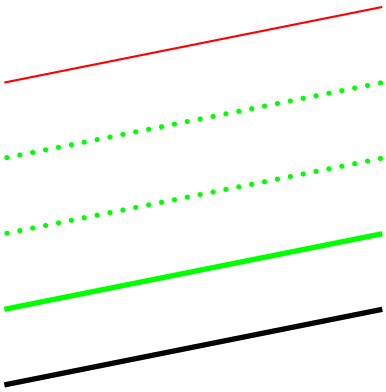
```
\end{pspicture}
\hfill
\begin{pspicture}(5,5)
  \psset{blendmode=1}% type /Compatible
  \psframe*[linecolor=red](0,1)(3,4)
  \psframe[fillcolor=blue,fillstyle=shape](2,2)(5,5)
  \psframe[fillcolor=green,fillstyle=shape](1,0)(4,3)
  \pscircle[fillcolor=cyan,fillstyle=shape,
    shapealpha=0.3](1.5,3.5){1.25}
  \rput(1.5,3.5){\huge\textbf{1}}
\end{pspicture}

\begin{pspicture}(5,5)
  \psset{blendmode=2}% type /Screen
  \psframe*[linecolor=red](0,1)(3,4)
  \psframe[fillcolor=blue,fillstyle=shape](2,2)(5,5)
  \psframe[fillcolor=green,fillstyle=shape](1,0)(4,3)
  \pscircle[fillcolor=cyan,fillstyle=shape,
    shapealpha=0.3](1.5,3.5){1.25}
  \rput(1.5,3.5){\huge\textbf{2}}
\end{pspicture}
\hfill
\begin{pspicture}(5,5)
  \psset{blendmode=3}% type /Multiply
  \psframe*[linecolor=red](0,1)(3,4)
  \psframe[fillcolor=blue,fillstyle=shape](2,2)(5,5)
  \psframe[fillcolor=green,fillstyle=shape](1,0)(4,3)
  \pscircle[fillcolor=cyan,fillstyle=shape,
    shapealpha=0.3](1.5,3.5){1.25}
  \rput(1.5,3.5){\huge\textbf{3}}
\end{pspicture}
```

2.6 \addtopsstyle

`\addtopsstyle{style-name}{settings}`

This macro allows to add some more settings to an existing style. If the style is not defined, then `\addtopsstyle` behaves like the already defined `\newpsstyle` macro.

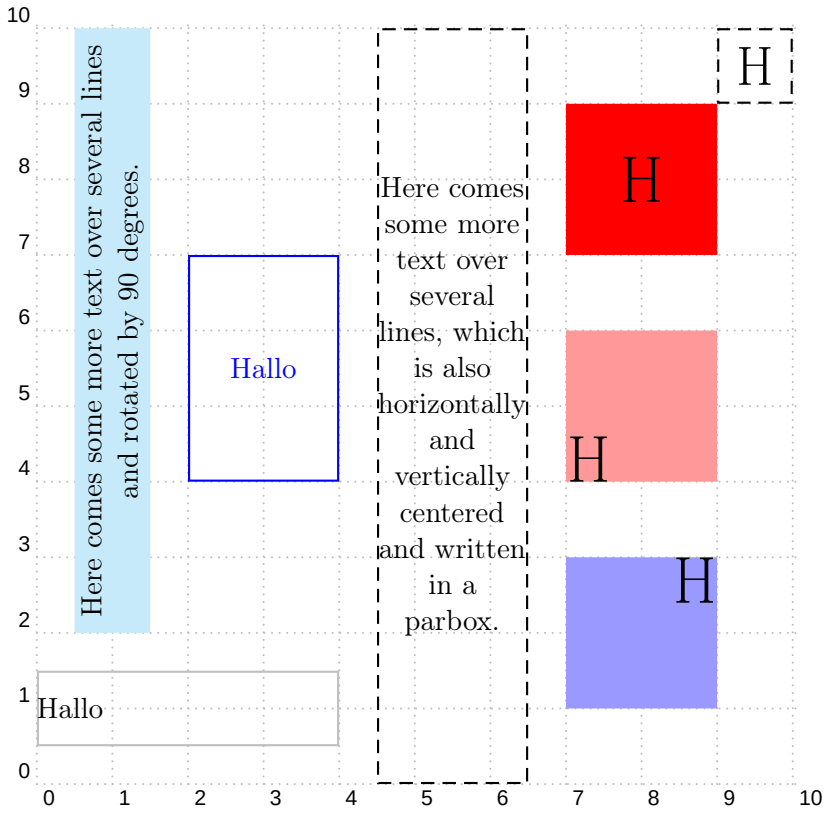


```
\newpsstyle{Fiber}{linewidth=2pt}
\begin{pspicture}(5,5)
   \psline[style=Fiber](0,0)(5,1)
   \addtopsstyle{Fiber}{linecolor=green}
   \psline[style=Fiber](0,1)(5,2)
   \addtopsstyle{Fiber}{linestyle=dotted}
   \psline[style=Fiber](0,2)(5,3)
   \addtopsstyle{Fiber}{}
   \psline[style=Fiber](0,3)(5,4)
   \addtopsstyle{Fibber}{linecolor=red}
   \psline[style=Fibber](0,4)(5,5)
\end{pspicture}
```

2.7 \psTextFrame

`\psTextFrame[settings](x1,y1)(x2,y2){Text}`

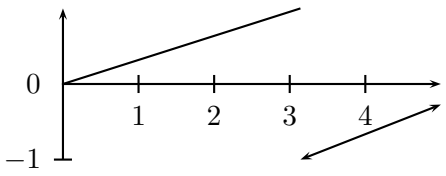
The *Text* cannot have a linebreak. In case it is needed, put the *Text* into a `minipage` or `\parbox`, as seen in the following example. The `ref`-option allows different placing and the `rot`-option allows the rotating of the *Text*. The macro itself first uses the `\psframe` and then `\rput` with calculated coordinates.




```
\begin{pspicture}[showgrid=true](0,-0.5)(10,10)
 \psTextFrame[linecolor=lightgray,ref=l](0,0.5)(4,1.5){Hallo}
 \psTextFrame[linecolor=blue](2,4)(4,7){\color{blue}Hallo}
 \psTextFrame[linestyle=dashed](9,9)(10,10){\huge H}
 \psTextFrame*[linecolor=red,linestyle=dashed](7,7)(9,9){\Huge H}
 \psTextFrame*[linecolor=red!40,ref=lB](7,4)(9,6){\Huge H}
 \psTextFrame*[linecolor=blue!40,ref=rt](7,1)(9,3){\Huge H}
 \psTextFrame[linestyle=dashed](4.5,0)(6.5,10){%
  \parbox{2cm}{\centering Here comes some more text over several
    lines, which is also horizontally and vertically centered and
    written in a parbox.}}
 \psTextFrame*[linecolor=cyan!20,rot=90](.5,2)(1.5,10){%
  \parbox{8cm}{\centering Here comes some more text over several
    lines and rotated by 90 degrees.}}
\end{pspicture}
```

2.8 Special coordinates

Additionally to the existing !-operator for PostScript coordinates, there is new a *-operator, which invokes the algebraic parser before the coordinates are passed to the default !-operator. The syntax is pretty easy: `(<value> {f(x)})`. In the following example the predefined value of Pi from pstricks.pro is used. The function must be enclosed in braces when itself contains round braces.




```
\SpecialCoor
\begin{pspicture}(0,-1)(5,1)
  \psaxes{->}(0,0)(0,-1)(5,1)
  \psline(0,0)(*Pi {sqrt(abs(cos(x)))})
  \psline{<->}(*Pi {cos(x)})(*5 {sin(x)*cos(x)})
\end{pspicture}
```

2.9 Code changes

```
% hv 2007-10-16 to fix the bug in pst-node with \\[name=...]
\def\ps@ifnextchar#1#2#3{%
  \let\reserved@d= #1%
  \def\reserved@a{#2}\def\reserved@b{#3}%
  \futurelet\@let@token\ps@ifnch}
\def\ps@ifnch{%
  \ifx\@let@token\reserved@d \let\reserved@b\reserved@a \fi
  \reserved@b
}
```

3 pstricks.pro

```
/Pyth2 { % Pythagoras, xA yA xB yB
  3 -1 roll  % xA xB yB yA
  sub % xA xB yB-yA
  3 1 roll  % yB-yA xA xB
  sub % yB-yA xA-xB
  Pyth } def
```

This new PostScript function allows to calculate the distance between two points, given by their coordinates whereas the existing /Pyth does ist for two values.

Part II
pst-node – package

4 pst-node.tex (1.00– 2007/10/16)

4.1 Bugfix for psmatrix

A long standing bug with psmatrix and using the `name` option is now fixed. The following now works as expected:

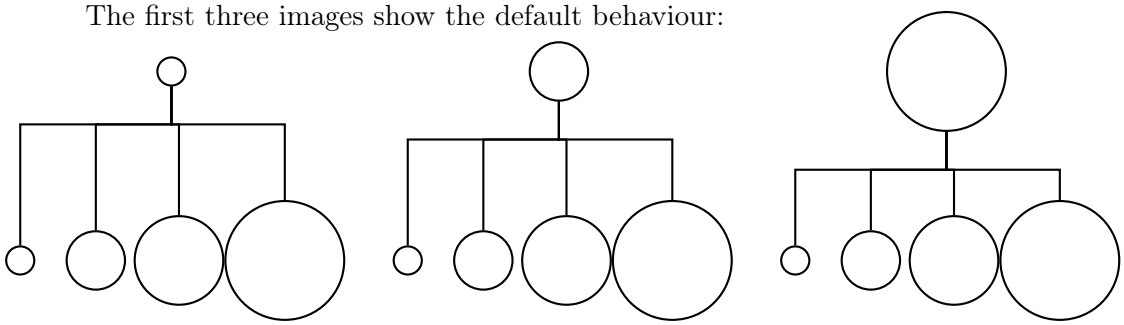

```
\begin{psmatrix}[rowsep=5mm]
[name=a]a\\
[name=b]b\\[1cm]
[name=c]c\\
\end{psmatrix}
\ncline{a}{b}
\ncarc{a}{c}
```

An optional argument after \\ is now scanned in the correct way.

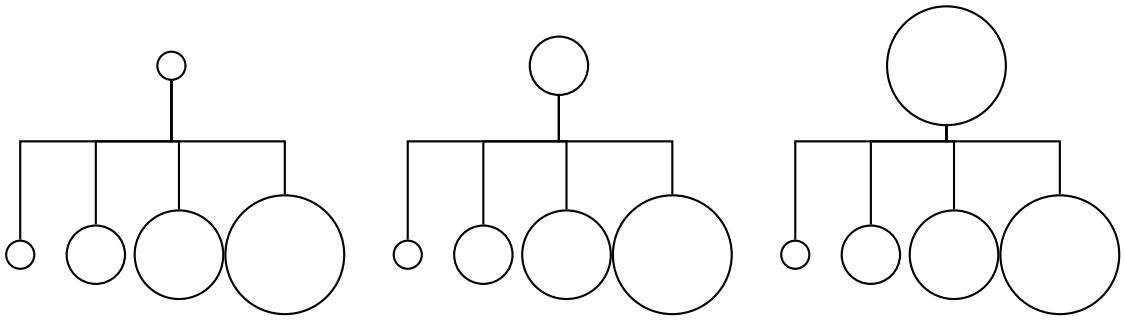
4.2 New option pcRef

There is a new option `pcRef` for the `\ncangles` connection. By default, the reference point for the `armA` option is the border of the node. This makes it difficult, to get horizontically aligned lines for different node images. With `pcRef=true` the node center is the reference point and the connection is still drawn from the border of the node.

The first three images show the default behaviour:



The next three one the influence of `pcRef=true`; the horizontal line for the three examples is on the same height:



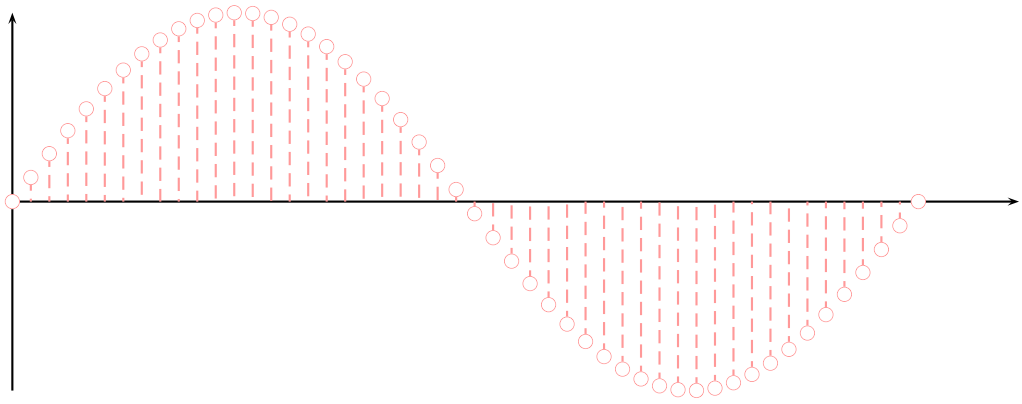
```
\begin{pspicture}(5,4)
  \cnode(2.5,3.5){0.2}{A}
  \cnode(0.5,1){0.2}{B1}
  \cnode(1.5,1){0.4}{B2}
  \cnode(2.6,1){0.6}{B3}
  \cnode(4,1){0.8}{B4}
  \psset{angleB=90,angleA=-90,armA=1cm}
  \ncangles[pcRef=true]{A}{B1}
  \ncangles[pcRef=true]{A}{B2}
  \ncangles[pcRef=true]{A}{B3}
  \ncangles[pcRef=true]{A}{B4}
\end{pspicture}
%
\begin{pspicture}(5,4)
  \cnode(2.5,3.5){0.4}{A}
  \cnode(0.5,1){0.2}{B1}
  \cnode(1.5,1){0.4}{B2}
  \cnode(2.6,1){0.6}{B3}
  \cnode(4,1){0.8}{B4}
  \psset{angleB=90,angleA=-90,armA=1cm}
  \ncangles[pcRef=true]{A}{B1}
  \ncangles[pcRef=true]{A}{B2}
  \ncangles[pcRef=true]{A}{B3}
  \ncangles[pcRef=true]{A}{B4}
\end{pspicture}
%
\begin{pspicture}(5,4)
  \cnode(2.5,3.5){0.8}{A}
  \cnode(0.5,1){0.2}{B1}
  \cnode(1.5,1){0.4}{B2}
  \cnode(2.6,1){0.6}{B3}
  \cnode(4,1){0.8}{B4}
  \psset{angleB=90,angleA=-90,armA=1cm}
  \ncangles[pcRef=true]{A}{B1}
  \ncangles[pcRef=true]{A}{B2}
```

```
  \ncangles[pcRef=true]{A}{B3}
  \ncangles[pcRef=true]{A}{B4}
\end{pspicture}
```

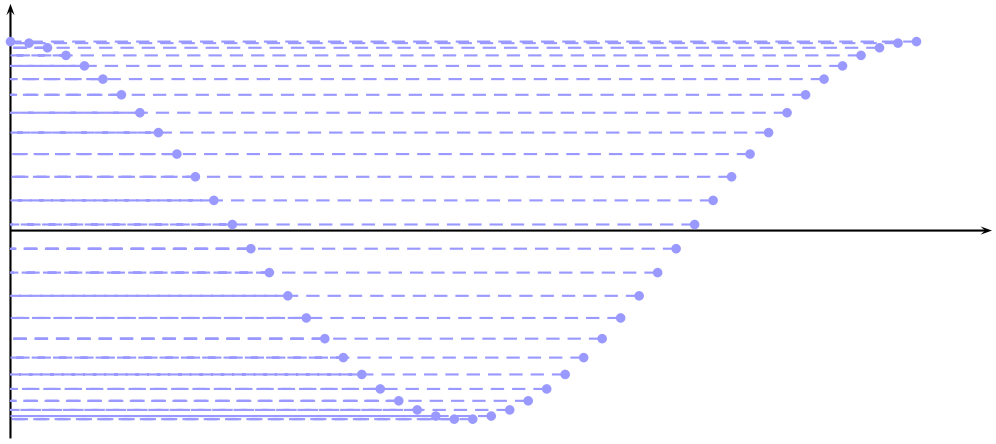
Part III
pst-plot – package

5 pst-plot.tex (1.00– 2007/06/26)

5.1 New options LineToXAxis and LineToYAxis



```
\psset{xunit=0.0333cm,yunit=2.5cm}
\begin{pspicture}(0,-1)(400,1)
  \psline{->}(0,0)(400,0)
  \psline{->}(0,-1)(0,1)
  \psplot[plotstyle=LineToXAxis,linestyle=dashed,plotpoints=50,
       linecolor=red!40,
           showpoints=true,dotstyle=o,dotsize=0.2]{0}{360}{x sin}
\end{pspicture}
```



```
\psset{xunit=0.0333cm,yunit=2.5cm}
\begin{pspicture}(0,-1.2)(400,1.4)
  \psline{->}(0,0)(390,0)
  \psline{->}(0,-1.1)(0,1.2)
  \psplot[plotstyle=LineToYAxis,linestyle=dashed,plotpoints=50,
       linecolor=blue!40,
           showpoints=true]{0}{360}{x cos}
\end{pspicture}
```

Part IV
pst-tree – package

6 pst-tree.tex (1.01– 2007/06/26)

There was a bug with `style=...` in fact of a missing `\use@par` in `\pstree@ii`.