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

Herbert Voß[*]

September 12, 2020

---

[*] Herbert.Voss@pstricks.de;
thanks to Patrick Drechsler.

Contents

Part I

`pstricks` – package

1 `pstricks.sty`

1.1 Error messages

- 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 (example for a dcoument file called `pstricks.tex`):

```
! LaTeX Error: ‘pstricks.tex’ is a forbidden name for your document,
               it is already a name of a package.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.13 \documentclass
                   {article}
? H
Choose another name for your document
```

1.2 Optional arguments

`pstricks` supports transparent colors with Ghostscript’s `.setopacityalpha`, `.setblendmode`, and `.setshapealpha`. These functions are not known to VTEX or Adobes Distiller. The optional argument `vtex` disables transparencies and `distiller` overrides the Ghostscript functions with the ones from the Distiller.

2 `pstricks.tex` (2.97– 2019/05/11)

2.1 Makro `\psDEBUG`

`pstricks.tex` defines the option `PstDebug=0|1`, which can be used for debugging. The new macro `\psDEBUG` makes it easier to write some debugging information into the package files. The macro is only valid, if `PstDebug=1` is set, otherwise the macro does nothing.

`\psDEBUG[optional arg]{text}`

`\psDEBUG` writes the argument `text` into the log file. Without an optional argument the word `pstricks` is used. The following output of the log file

```
...
<key:xticksize>: setting ticksize to max
LaTeX Font Info: External font ‘cmex10’ loaded for size
(Font)          <7> on input line 26.
LaTeX Font Info: External font ‘cmex10’ loaded for size
(Font)          <5> on input line 26.
<pst@@hlabels>: xticksizeC=0.0pt
...
```

was possible with `\psset{PstDebug=1}`[1] and inside of `pstricks-add` with (only the first for example):

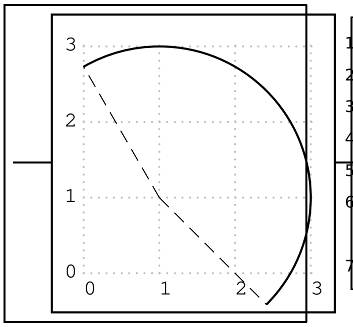
```
...
 \psDEBUG[key:ticksize]{setting ticksize}
...
```

2.2 Option `shift`

The optional argument `shift` can be used for a vertical alignment of the `pspicture` box. With `shift=*`, instead of a value or a length it is possible to center the `pspicture` box vertically to the baseline of the current line.

---

[1] Can also be used locally for a macro when used as optional argument in the usual way.

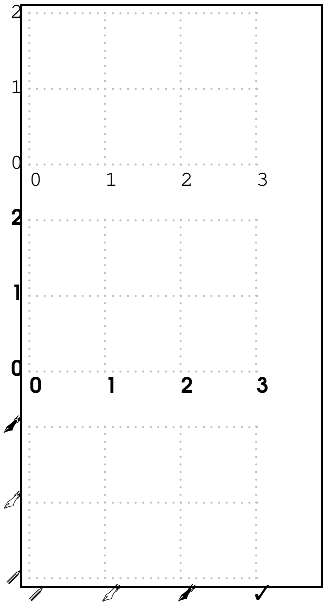


```
\usepackage{pstricks}
\rule{5mm}{0.5pt}%
\psframebox{%
\begin{pspicture}[showgrid=true,
  shift=*](-0.3,-0.4)(3.2,3.3)
 \psarc[showpoints=true](1,1)
    {2}{-45}{120}
\end{pspicture}}\rule{5mm}{0.5pt}
```

2.3 Option `gridfont`

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

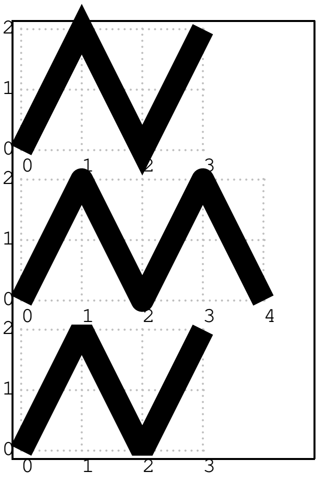
Helvetica (default) - Helvetica-Narrow - Times-Roman - Courier - AvantGard -NewCenturySchlbk - Palatino-Roman - Bookman-Demi - ZapfDingbats - Symbol



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

2.4 `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 to 0 and can take values of 0,1,2. Other values will have 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}
```

2.5 `linecap`

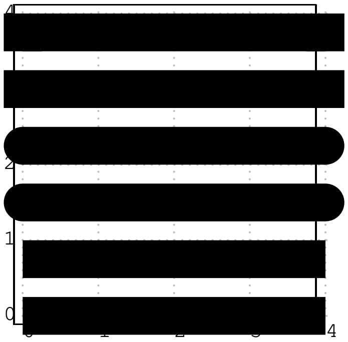
The value of `linecap` determines how the line ends are drawn:

0 lines are cut (default)

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

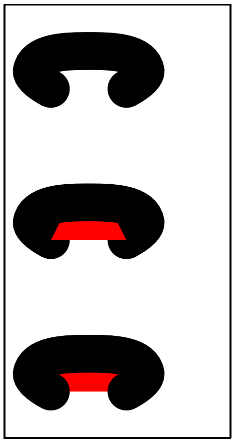
2 lines are ended by a filled half square of radius 0.5·`\pslinewidth`

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}
```

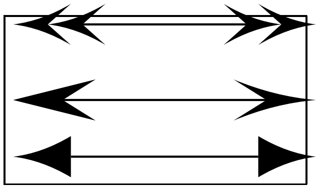
Using this optional argument makes only sense in some special cases, because it is the same as the arrow type `c-c`. But the arrows are not part of the current path and filling an open curve with the `linecap` option is different to a curve using the `c-c` arrow.



```
\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}
```

2.6 New arrowtype `D>` and `D>D>`

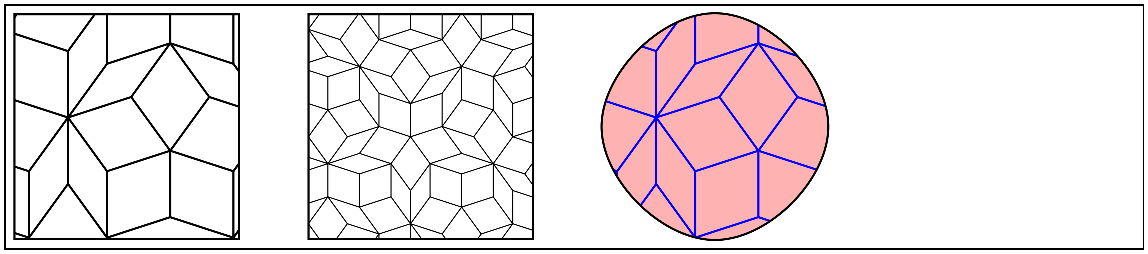
All arrows are drawn as polygons. The new arrow type `D>` or `<D` for the other way round, draws its lines as bezier curves, which looks nicer for big arrows.



```
\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}
```

2.7 Fill style `penrose`

The valid optional arguments are `penrose`, `penrose*`, and `hatchcolor`. The star version is only seen, if there is a fillcolor or a background different to white.



```
\begin{pspicture}(3,3)
\psframe[fillstyle=penrose](3,3)
\end{pspicture} \qquad
\begin{pspicture}(3,3)
\psframe[fillstyle=penrose,psscale=0.5](3,3)
\end{pspicture} \qquad
\begin{pspicture}(3,3)
\psccurve[fillstyle=penrose*,fillcolor=red!30,hatchcolor=blue](0,1.5)
   (1.5,3)(3,1.5)(1.5,0)
\end{pspicture}
```

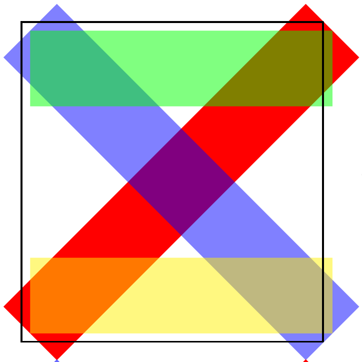
2.8 Transparent colors

The package `pstricks-add` already defined a fillstyle for transparency colors by using the Ghostscript's 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), as nearly all PostScript viewer cannot show transparencies.

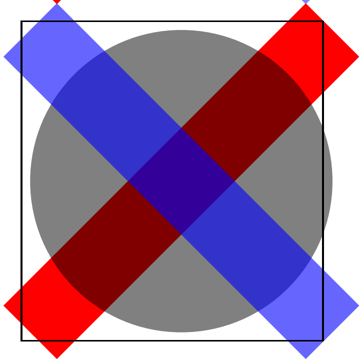
Loading the `pstricks` package with the option `vtex`, disables the transparency effekts and everything works as before.

2.8.1 Options `strokeopacity` and `opacity`

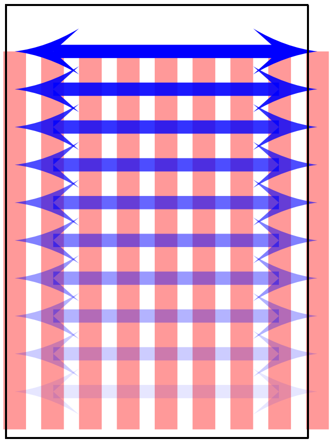
For the existing fill style `solid` the new option `opacity` can be used to get also transparent colors. It is predefined by 1 $(0\dots1)$, which is the old behaviour, no transpareny. The option is valid only for PostScripts fill commands. Lines and curves can be transparent with setting the option `strokeopacity`, which can have a different value than the `opacity` option.



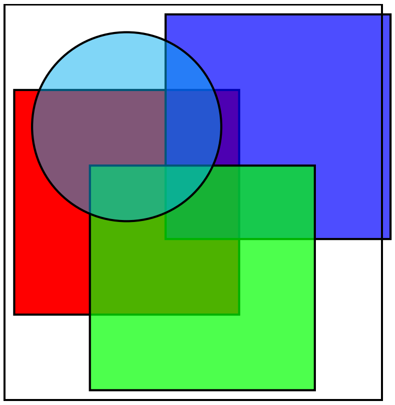
```
\begin{pspicture}[linewidth=1cm](4,4)
  \psline[linecolor=red](0,0)(4,4)
  \psline[linecolor=blue,strokeopacity=0.5](0,4)
      (4,0)
  \psline[linecolor=green,strokeopacity
      =0.5](0,3.5)(4,3.5)
  \psline[linecolor=yellow,strokeopacity
      =0.5](0,0.5)(4,0.5)
\end{pspicture}
```



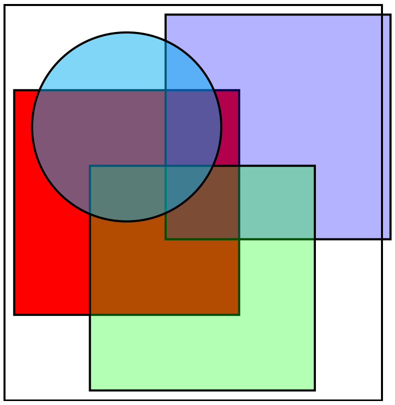
```
\begin{pspicture}[linewidth=1cm](4,4)
\psline[linecolor=red](0,0)(4,4)
\pscircle*[opacity=0.5](2,2){2}
\psline[linecolor=blue,strokeopacity=0.6](0,4)
    (4,0)
\end{pspicture}
```



```
\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}
```



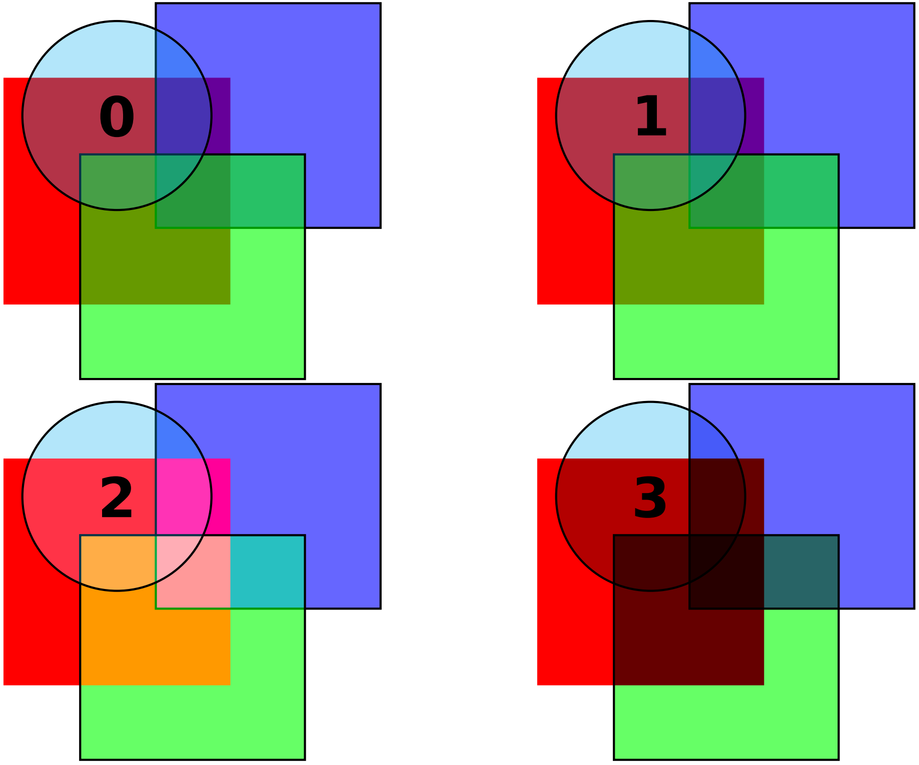
```
\begin{pspicture}(5,5)
 \psset{fillstyle=solid}
 \psframe[fillcolor=red](0,1)(3,4)
 \psframe[fillcolor=blue,opacity=0.3](2,2)
    (5,5)
 \psframe[fillcolor=green,opacity=0.3](1,0)
    (4,3)
 \pscircle[fillcolor=cyan,
  opacity=0.5](1.5,3.5){1.25}
\end{pspicture}
```

2.8.2 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 Ghostscript's `.setopacityalpha` function and the new style `shape` and the `blendmode` together with `.setshapealpha`. `shapealpha` is predefined with 0.6 and both alpha values can be chosen from the range $0 \le \alpha \le 1$.



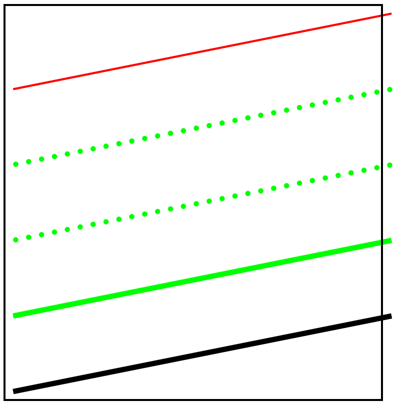
```
\begin{pspicture}(5,5)% default blendmode
 \psframe*[linecolor=red](0,1)(3,4)
 \psframe[fillcolor=blue,fillstyle=shape](2,2)(5,5)
 \psframe[fillcolor=green,fillstyle=shape](1,0)(4,3)
 \pscircle[fillcolor=cyan,fillstyle=shape,
  shapealpha=0.3](1.5,3.5){1.25}
  \rput(1.5,3.5){\huge\textbf{0}}
\end{pspicture}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

2.9 `\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.

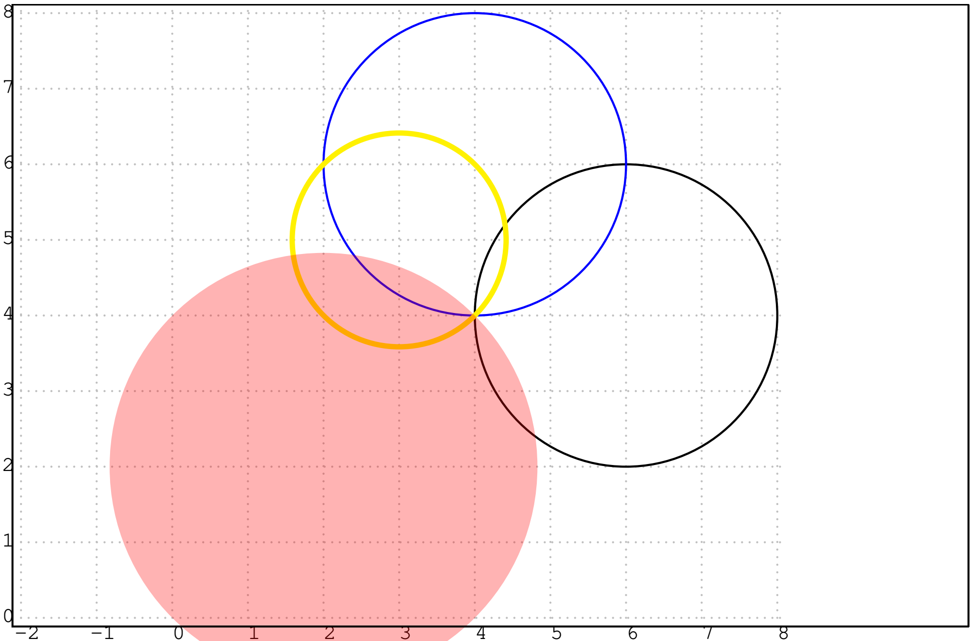


```
\newpsstyle{Fiber}{linewidth=2pt}
\begin{pspicture}(5,5)
  \psline[style=Fiber](0,0)(5,1)
  \addtopsstyle{Fiber}{linecolor=green}
  \psline[style=Fiber](0,1)(5,2)
  \addtopsstyle{Fiber}{linestyle=dotted}
  \psline[style=Fiber](0,2)(5,3)
  \addtopsstyle{Fiber}{}
  \psline[style=Fiber](0,3)(5,4)
 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

2.10 `\pscircleOA`

`\pscircleOA[settings](x0,y0)(xA,yA)`

(x_0, y_0) is the center and (x_A, y_A) a given point of the circle. The radius is calculated by TeX.

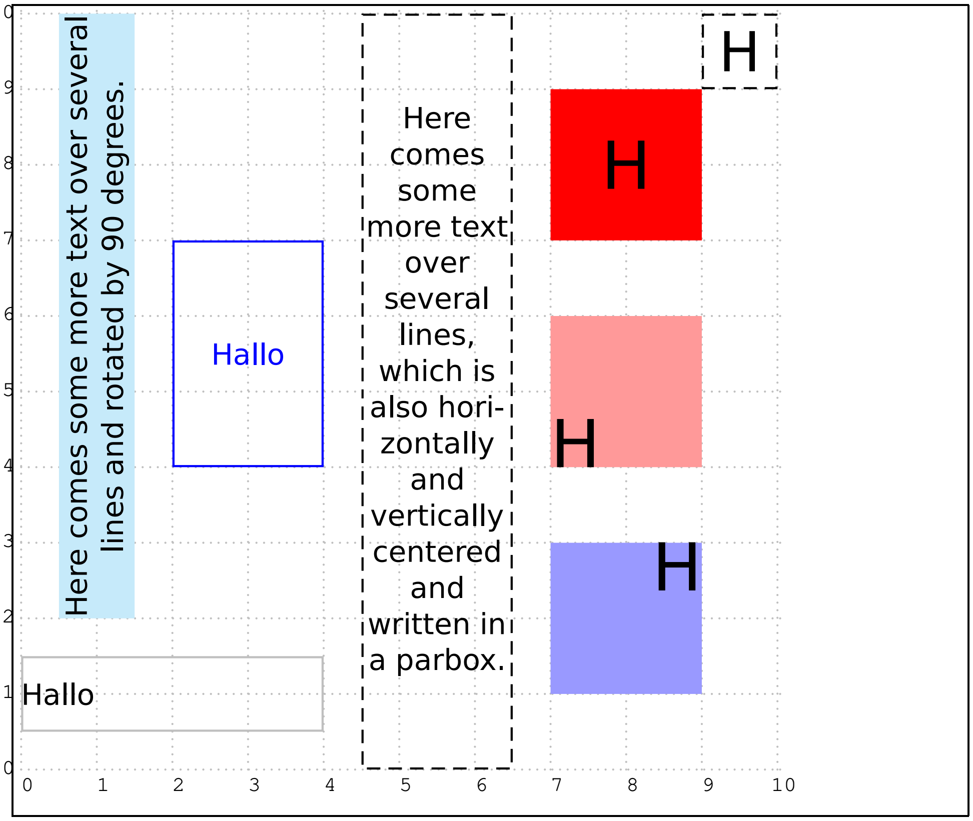


```
\begin{pspicture}[showgrid=true](-2,0)(8,8)
\pscircleOA(6,4)(4,4)
\pscircleOA[linecolor=blue](4,6)(4,4)
\pscircleOA[linewidth=2pt,linecolor=yellow](3,5)(4,4)
\pscircleOA*[opacity=0.3,linecolor=red](2,2)(4,4)
\end{pspicture}
```

2.11 `\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 the `\rput` macro with calculated coordinates.


10
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10
Here comes some more text over several lines and rotated by 90 degrees.
Hallo
Hallo
Here comes some more text over several lines, which is also horizontally and vertically centered and written in a parbox.
H
H
H
H

```
\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}
```