Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

Trivial Experiments with psTricks manipulation

Radhakrishnan CV and Rajagopal CV
River Valley Technologies, Trivandrum, India
http://www.river-valley.com

Antoine Chambert-Loir
Ecole polytechnique, Palaiseau Cedex, France
http://www.math.polytechnique.fr/~chambert

August 10, 2003

1. Objectives

psTricks macros cannot be used with pdfTEX, since psTricks uses PostScript arithmetic, which isn't in PDF. As such, a package *viz.*, pdfTricks.sty has been written to circumvent this limitation, so that the extensive facilities offered by the powerful psTricks package can be made use of in a pdfTEX document. This is brought by making use of the shell escape function available in the web2c TEX compiler, while this package is of no use to other commercial implementations.

Shell escape provides us the facility to suspend a TEX compilation midway, perform the functions that we want to do with a shell command, complete those functions, return to the TEX compilation and finish the job. This facility can be invoked by using \write18{⟨*shell command*⟩}. So if you use a line:

```
write18{ls -l}
```

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

TEX will stop at the moment when it encounters the `\write18` command, escapes to shell, executes `ls -l` command and will return to the compilation process.

2. Shell escape

In all the web2c TEX implementations, shell escape is turned off by default. You can turn it on by changing the line in your `texmf.cnf` (configuration file of your TEX system) to:

```
shell_escape = t
```

which will be 'f' by default. You have to recreate all the formats like `pdflatex.fmt` or whatever you might need. This is not a wise step though, since it makes you open to the assault of Trojan macros without your knowledge.

The more elegant way will be to invoke shell escape with a switch, each time you run the compiler like:

```
pdflatex -shell-escape {⟨file name⟩}
```

This eliminates the vulnerability to macros/packages written by intelligent criminal minds.

3. Usage

The usage is very simple, as usual you have to load the package with a package loading line in the preamble of your document:

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements


```
\usepackage{pdftricks}
```

All the packages that are needed for the compilation of the `pstricks` code shall be enclosed within an environment `\begin{psinputs}` ...`\end{psinputs}` in order to make it available to all the figure documents during compilation. An example of this will look like

```
\begin{psinputs}
 \usepackage{pstricks}
 \usepackage{color}
 \usepackage{pstcol}
 \usepackage{pst-plot}
 \usepackage{pst-tree}
 \usepackage{pst-eps}
 \usepackage{multido}
 \usepackage{pst-node}
 \usepackage{pst-eps}
 ...
\end{psinputs}
```

Each `pspicture` environment along with any macro code shall be enclosed within an environment *viz.*, `pdfdisplay` or `pdfpic`. This will enable pdfLaTeX to write external files like `fig1.tex`, `fig2.tex`, ..., depending on the number of `psTricks` figures included in your document. `pdftricks` will intelligently find out whether your pdfTeX implementation has shell escape facility enabled or not. Depending on this, there are two ways of approaching the problem which are given in the succeeding sections.

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

3.1. With shell escape

With shell escape option invoked, one need not bother anything. The usual compilation with pdfLaTeX will result in a `pdf` document with all the figures appearing in the correct positions. pdfTeX will do the necessary job of writing all the `pstricks` code to self standing external `*.tex` documents, escapes to shell, compiles the `*.tex` document, converts to `*.eps`, translates to `*.pdf` and includes in the location where the `pstricks` code appeared.

3.2. Without shell escape

A special shell script viz., `pst2pdf` is also provided. Now as the second step, you have to execute this shell script, `pst2pdf` which will compile all the newly written `fig1.tex`,. . . , and respective `*.dvi` will be generated, which will be translated into respective `*.eps` files and finally translated to `*.pdf` by calling `epstopdf`. As a final step, you shall run pdfLaTeX again, so that the `*.pdf` figures will be included with the usual `\includegraphics` command automatically in the locations wherever the `pstricks` code appeared.

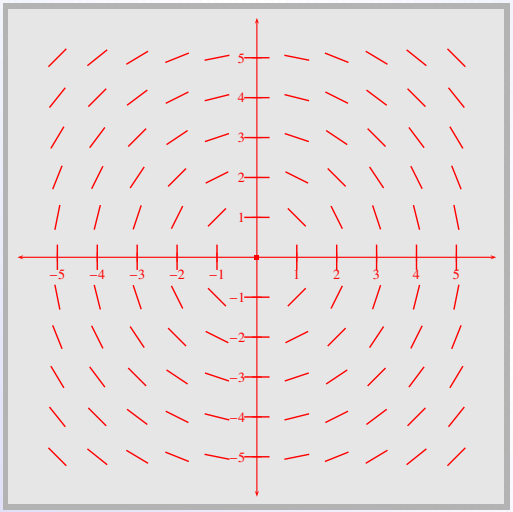
See an example below:

```
\begin{pdfdisplay}
\input pst-eps
\TeXtoEPS
\SpecialCoor
\begin{pspicture}(-6,-6)(6,6)
  \psaxes{<->}(0,0)(-6,-6)(6,6)
  \psset{arrows=->}
  \multido{\ia=-5+1}{11}{%
\multido{\ib=-5+1}{11}{%
 \pstVerb{/x \ia\space def
    /y \ib\space def
    y 0 eq
```

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements


```
      {/ValueTempA 0 def
       /ValueTempB 0.5 def}
      {/ValueTempZ 2 1 x x mul y y mul div add sqrt mul def
       /ValueTempA 1 ValueTempZ div def
       /ValueTempB x y ValueTempZ mul div def}
      ifelse}
 \psline(! x ValueTempA sub y ValueTempB sub)
        (! x ValueTempA add y ValueTempB add)}}
\end{pspicture}
\endTeXtoEPS
\end{pdfdisplay}
```

This psTricks code will result in the following graphic:



Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

The following are the steps in the sequential order:

1. `pdflatex` {⟨*your document*⟩}
2. `pst2pdf`
3. `pdflatex` {⟨*your document*⟩}

4. Options and Hooks

`shell` This option will invoke the shell escape functions.

`noshell` This option will suppress the shell escape functions and will assume that there is no shell escape facility.

`NoProcess` A command `\NoProcess` is available to facilitate suppressing of pdf generation of those figures whose pdf's are already available. This might prove helpful when you have more figures to process and many of them are perfected and don't need recompilation and translation everytime you run pdfLaTeX. The usage is:

```
\NoProcess[comma separated and/or hyphen separated ranges]
```

If you have ten figures and if you want to suppress the processing of the figure numbers 1, 2, 4 to 8 you can issue the command at the top of the document as:

```
\NoProcess[1,2,4-8]
```

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

`pdfdisplay` An environment to enclose the `pstricks` code, which will write out stand alone `*.tex` files and create `pdf` graphics. This environment will help to display the graphic.

`pdfpic` Does exactly the same as `pdfdisplay`, kept for backward compatibility.

`pdfinline` An environment for inline `pstricks` graphics.

There are few hooks to resize the graphic thus translated and included.

```
\configure[pdfgraphic][width=2in]
\configure[pdfgraphic][height=3in]
\configure[pdfgraphic][width=2in,height=3in]
\configure[pdfgraphic][scale=.5]
\configure[pdfgraphic][linecolor=red]
\configure[pdfgraphic][linewidth=1pt]
\configure[pdfgraphic][background=green]
\configure[pdfgraphic][rulesep=6pt]
```

the functionality is same as the `width`, `height` and `scale` options as in the `\includegraphics` command. But the graphic will be restricted to aspect ratio. `linecolor`, `linewidth`, `rulesep` and `background` corresponds to the box surrounding the graphic. The default values are white, 0pt, 0pt and white respectively.

5. Limitations/Dependencies

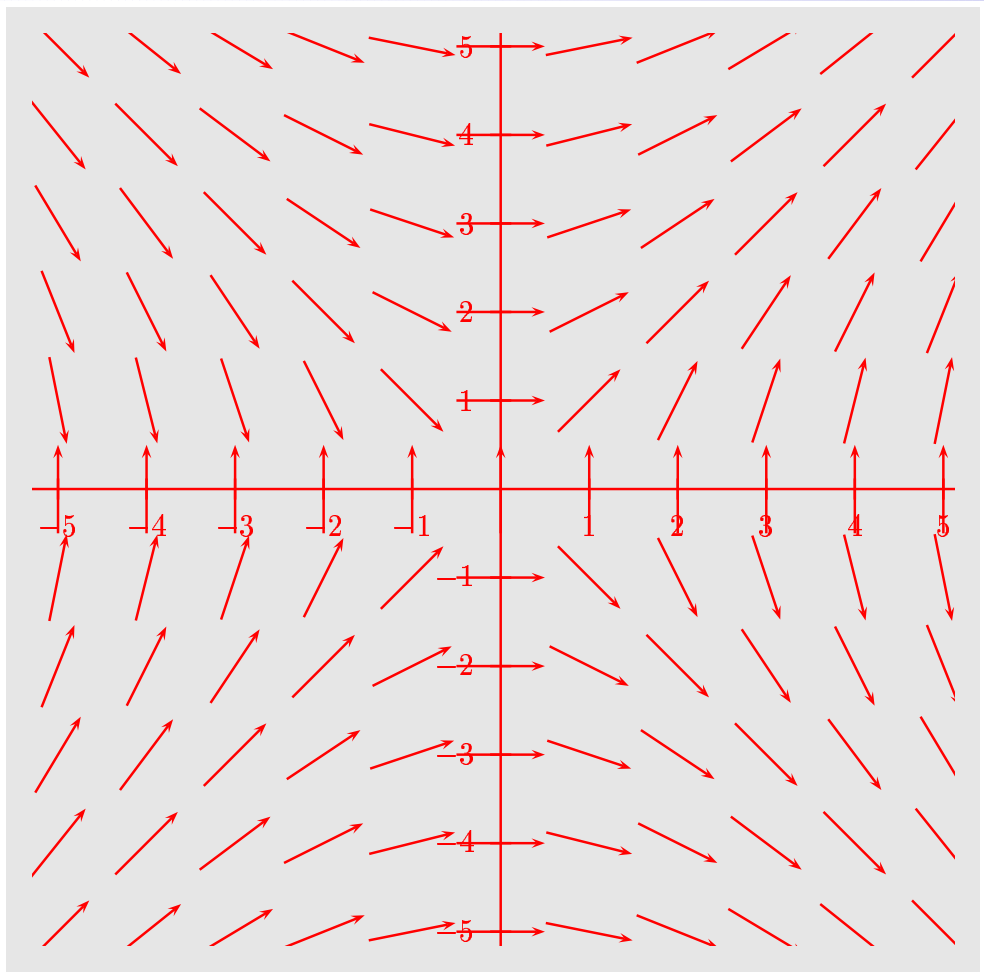
- Usage is limited to `web2c` TEX implementations.
- The shell escape commands used in this package are typical UNIX commands and as such it works best in GNU LINUX/UNIX flavours.
- `pdfTricks` needs, `graphicx` and `keyval` packages; `epstopdf`, the Perl script that translates `eps` files into `pdf`.

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

6. Licence/Acknowledgements

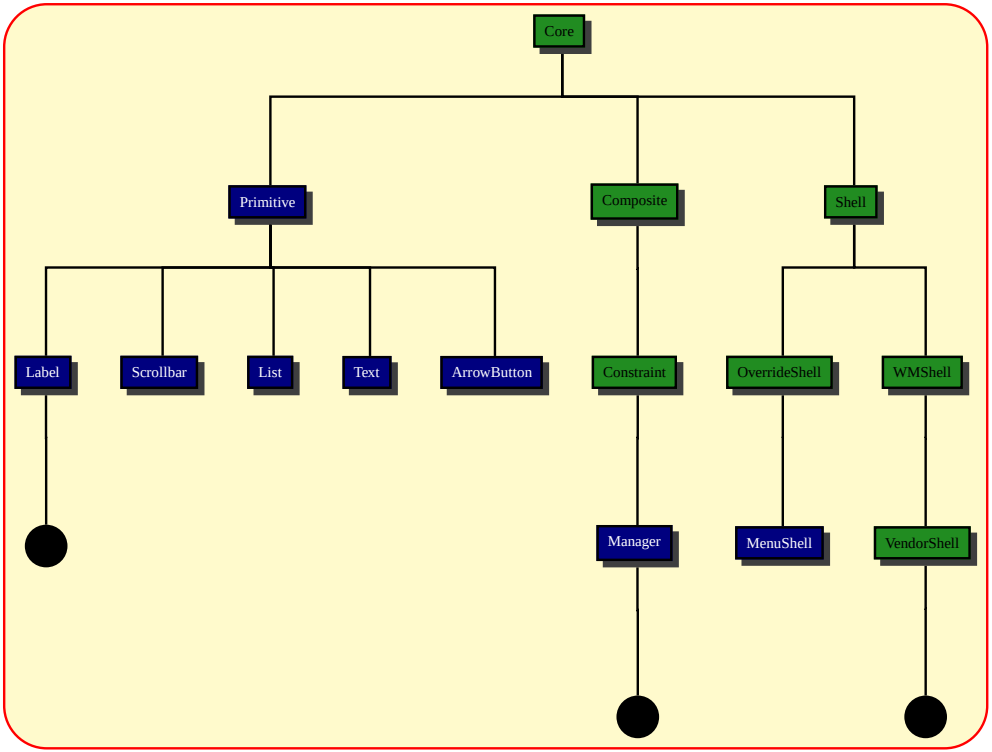
The package is distributed under GNU General Public Licence. See some more examples in the following pages. An example `test.tex` and `test.pdf` are included for you to play with. The authors gratefully acknowledge the encouragement provided by Sebastian Rahtz, Oxford University Computing Services; KSS Nambooripad and Mohit Agarwal of Indian TEX Users Group, Trivandrum. Our special thanks are due to E. Krishnan of Free Software Foundation of India at whose goading that we wrote this package and to Thierry Bouche who has kindly provided the hack to find out the shell escape facility on the fly, in the pdfTEX implementation.

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

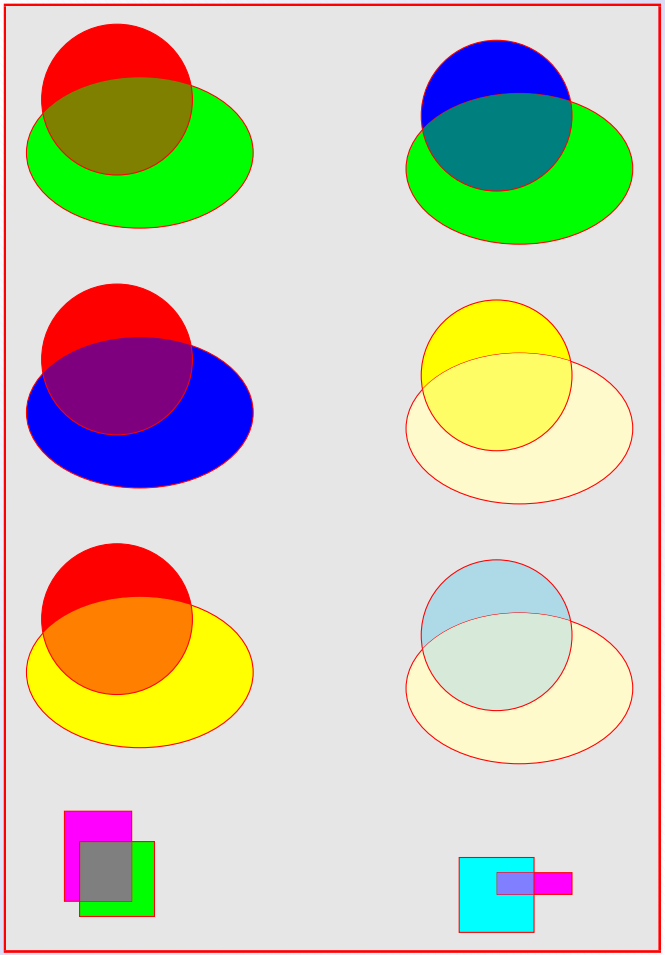


Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements

Set of Motif widgets classes



Core Xt Class

List Motif Class

Objectives
Shell escape
Usage
Options and Hooks
Limitations/Dependencies
Licence/Acknowledgements