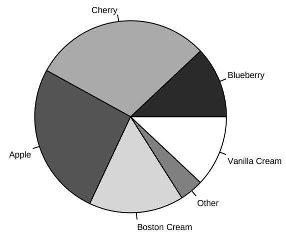
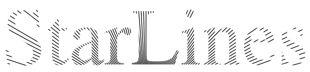
Psfig/TeX 1.4 Users Guide

Trevor Darrell

1 Introduction

Psfig/TeX is a macro package for TeX that facilitates the inclusion of PostScript figures into TeX documents. With the help of a compatible postprocessor, [1] PostScript figures are automatically scaled and positioned on the page, and the proper amount of space is reserved. The following are examples of included PostScript figures:





Custom characters such as ‘’ and ‘’ may be created and used freely throughout a document.

2 Simple figures

To include a PostScript figure with psfig, first load the psfig macros at the beginning of your document with

```
\input{psfig}
```

then, when you wish to include a figure, invoke the macro

`\psfig{figure=`*input*`}`

[1] The `dvips` program developed by T. Rokicki from `labrea.stanford.edu` has full Psfig support.

where *input* is the name of a PostScript file. Psfig will automatically position the figure at the current point on the page, and reserve the proper amount of space in TeX so that it doesn't block with any other objects on the page.

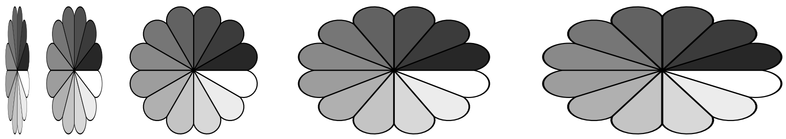
For example, if we have a file called 'piechart.ps' which contains the PostScript code to draw the chart in the introduction, we could use the commands

```
\par
\centerline{\psfig{figure=piechart.ps}}
\par
```

to include it as a centered paragraph. Since no mention of size was made in the above example, psfig draws the figure at its natural size (as if it was printed directly on a PostScript printer.) The pie's natural size is several inches across, which is a little large; the pie in the introduction was produced with:

```
\centerline{\psfig{figure=piechart.ps,height=1.5in}}
```

The `height` option specifies how tall the figure should be on the page. Since no `width` was specified, the figure was scaled equally in both dimensions. By listing both a `height` and `width`, figures can be scaled disproportionately, with interesting results:



This figure was produced with:

```
\centerline{\hbox{
\psfig{figure=rosette.ps,height=.8in,width=.15in}
\psfig{figure=rosette.ps,height=.8in,width=.35in}
\psfig{figure=rosette.ps,height=.8in}
\psfig{figure=rosette.ps,height=.8in,width=1.2in}
\psfig{figure=rosette.ps,height=.8in,width=1.5in}
}}
```

3 Sources of figures

Any program that produces PostScript as output can be used for psfig figures as long as it adheres to the bounding box comment convention (see below). For instance, the bitmap image of the author was included as a figure at left.

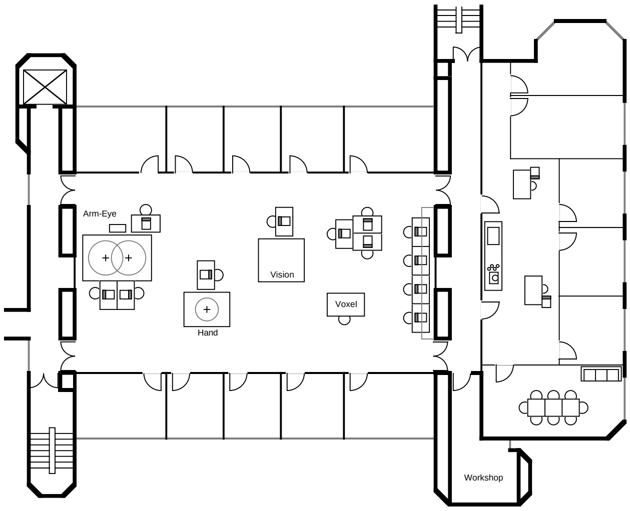


Figure 1: A figure produced by MacDraw

Since the Macintosh drawing applications produce PostScript, they can be used to create figures. Figure 1 shows a figure produced by MacDraw. More Mac figures are shown in the `macdemo.tex` document, and details on dealing with Macintosh postscript are covered in the `README file`.

4 Draft figures and Silent mode

Normally, psfig will print advisory messages to remind you that it is including figures as TeX processes a document. This behavior can be disabled with `\pssilent`, and re-enabled with `\psnoisy`.

Some PostScript figures can take quite a long time to transmit and print; for these figures a draft mode is available to speed printing intermediate versions of the document. A figure printed in draft mode will occupy the proper space as far as TeX is concerned but no PostScript code will be executed to print it. If psfig is not in silent mode, it will also print the name of the PostScript file in the space reserved for the figure. No `\special`'s are used in draft mode; thus any dvi processor can be used.

The macro `\psdraft` will switch into draft mode, and all subsequent psfig macros will produce draft figures until reaching the macro `\psfull`, which switches out of draft mode.

5 Bounding boxes

To properly translate and scale a figure psfig must know its 'natural' position on the page; this information is present in what is called the *bounding box* of a PostScript program. The bounding box is an outer limit to the marks created by a program, and is specified as four coordinates of a rectangle: the lower-left x coordinate (bbllx), the lower-left y coordinate (bblly), the upper-right x coordinate (bburx), and the upper-right y coordinate (bbury). Adobe has defined a convention whereby the bounding box of a program is contained in a 'bounding box comment'. [2] This comment, which must be present in any file to be used as a psfig figure, is a line of the form

`%%BoundingBox:` *bbllx bblly bburx bbury*

All values are in PostScript points, relative to the *default* transformation matrix. The only mandatory PostScript convention is that the first line of the file should begin with the characters '`%!`' (a '`%`' begins a comment in PostScript.) A good place for the bounding box comment is as the second line of the file.

The bounding box values may be specified directly on the psfig macro, using clauses `bbllx=`*bbllx*,`bblly=`*bblly*,..., in which case the file is not searched for the bounding box. Note that the `\psfig` macro will be confused by white space or newlines in its argument.

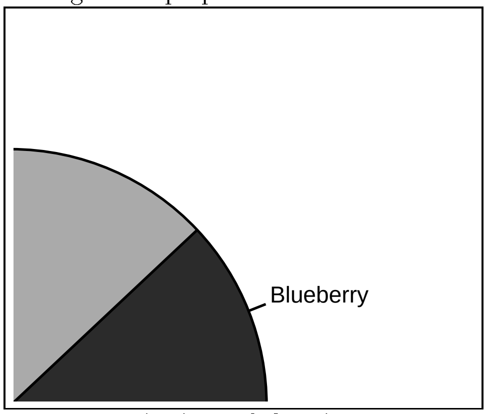
[2] See 'Appendix J: PostScript File Structuring Conventions' in *The PostScript Language Reference Manual*

6 Reserved size

There are two sizes associated with each psfig figure: the size at which it is to be printed on the page and the size it reserves in TEX. This latter size is appropriately termed the *reserved size*, and is expressed as clauses of the form "`rheight=`*dimen*" and "`rwidth=`*dimen*". If ommited, the reserved size defaults to the real size. Some special effects need to be transparent to TEX and thus have a zero reserved size, such as the grey box enclosing this paragraph.

7 Clipping

Normally a PostScript program can be expected to not mark the page outside its bounding box. If this is not the case, or if you want to specify a bounding box so as to isolate part of a larger figure, there is an option that sets the PostScript clip path so that no marks will show up outside the declared bounding box. Currently this is invoked by adding a clause of the form "`clip=`". Here a slice has been taken out of the pie chart in the example by specifying a smaller bounding box and adding the clip option.



A piece of the pie.

Some PostScript programs use the clipping path to position their output on the page; if a figure is being drawn at its natural size and position despite psfig commands to the contrary, it may need the clip option.

8 PostScript environment

The PostScript environment within psfig is fairly robust. All of the usual PostScript operators will operate as desired; in particular the operators 'showpage', 'initgraphics', and 'defaultmatrix' will all behave consistently inside a figure, except that 'showpage' will only do an 'initgraphics' and will *not* print or erase the current page. [3]

It is very possible, however, for a PostScript program to bypass the psfig environment and disrupt a document.

9 Acknowledgements

This work was done while the author was with the Department of Computer and Information Science, University of Pennsylvania. Ned Batchelder co-developed the original *troff* version of this program with the author, and was responsible for much of the overall design. Greg Hager provided an initial TEX implementation. The three broken out figures in the introduction were taken from examples in *The PostScript Language Tutorial and Cookbook.*

Version 1.4 incororates enhancements by J. Daniel Smith to accept the BoundingBox comment with or without a space after the colon. These enhancements use file reading code from Tom Rokicki's dvips program.

[3]For more detailed information about PostScript issues in the internals of psfig, see *Psfig – A Ditroff Preprocessor for PostScript Figures* in the USENIX 87 proceedings, or *Bringing troff up to speed* in the July 1987 issue of Unix Review.