Figure'ing and Picture'ing LaTeX

Combining text, figures, and images in a LaTeX document

by

Anil Goel ©1993

`(akgoel@math.uwaterloo.ca)`

`Version 1.0`

last modified

July 5, 1993

Math Faculty Computing Facility
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

1 Introduction

LaTeX is based on TeX [9] and is one of the typesetting languages available at Waterloo. It is easy to learn, widely available, and has an informative and comprehensive manual (see [11, 12]). One weakness of LaTeX is graphics. Although it has built-in facilities for drawing figures they are extremely ugly and difficult to use.

EEPIC [10] is a set of LaTeX macros that rectifies this deficiency in LaTeX to a large extent. Even so, it is not always possible to typeset arbitrarily complex figures, pictures and images in a LaTeX document. Therefore, at times it is necessary to use some other language (such as PostScript) to define parts of a document otherwise typeset using LaTeX .

PostScript [4, 5, 6] is a page description language which has achieved widespread use. PostScript syntax is given in plain ASCII text. Although PostScript programs are usually generated by a program, complex graphical effects can be obtained by small manually written PostScript programs (see [7]).

`Xfig` [1] is a fig language editor which runs under X11. It provides a point and click interface similar to that of MacDraw. Once editing is complete fig files can be converted into PostScript, EEPIC or a variety of other languages.

For details on using these and other facilities mentioned in this document, see the corresponding man pages in addition to the references mentioned at the end.

For the purposes of this document a graphical entity consisting of geometrical objects and text will be defined as a *figure*, the term *picture* will be used for an arbitrary drawing and a pixel image (such as raster scan) will be referred to as an *image*. The term drawing will used to refer to any of the above.

2 Enhancing LaTeX to include figures, pictures and images

There are several tools for enhancing the basic LaTeX package. Most of these can be classified into three distinct categories as follows:

Macro Packages These are written in TeX or LaTeX and provide more powerful or higher-level commands for drawing figures and pictures. Examples of macro packages include EEPIC [10] and MusicTeX [16].

Typesetting Languages These are independent languages much more suited for picture drawing than LaTeX. Examples include PostScript and Fig. Programs written in these languages can be included in a LaTeX document, usually with the assistance of a macro package.

Interactive Applications These are applications and programs which help to hide the lower-level details of using macros and languages from the user by means of convenient user interfaces. Prime examples of such applications are graphical editors (e.g., `xfig`) and translators.

In the rest of this section the above three categories of tools are discussed in reverse order starting with interactive applications that provide convenient user interfaces for creating complex figures and pictures. For each category, an example is used to illustrate the basic techniques of preparing a figure or picture, optionally converting the resulting drawing into a more suitable format and including the drawing in a LaTeX document. The examples chosen are some of the most popular tools of those available for performing a particular task.

2.1 An Interactive Application - XFIG

2.1.1 Using `xfig` to create a figure

`Xfig` is a graphical editor that runs under the `X11` window system. It is a powerful facility for drawing figures that works on both monochrome and color display devices.

Figure 1 illustrates a sample `xfig` screen while being used to create a complex figure. Figures generated in this manner can be stored in `fig` format files for future editing with `xfig`. `Fig` is an independent language that supports a variety of figure drawing facilities.

2.1.2 Converting `Fig` format files

LaTeX does not understand `Fig` format files. Therefore, files created by using `xfig` need to be converted into a format suitable for including in a LaTeX document.

`Fig` format files can be converted into a variety of other formats using the `fig2dev` [1] command. Two of the more popular of these formats suitable for LaTeX documents are the PostScript language and the EEPIC macros. By default `xfig` runs in landscape (wider than tall) mode and `fig2dev` also defaults to landscape. Both take a `-P` option which causes them to operate in portrait (taller than wide) mode. If you use -P on one and not the other you are likely to get a sideways figure.

As an example:

```
% fig2dev -L eepic model.fig > model.tex
```

```
and,
```

```
% fig2dev -L ps model.fig > model.ps
```

produced the EEPIC (LaTeX) and PostScript files that resulted in the two drawings shown in Figure 2 when included appropriately in this document.

Also note that `xfig` menu provides an *export* button which can be used by the user to directly convert and store the drawing being edited into a format of their choice. The use of *export* button in `xfig` eliminates the need for the user to explicitly convert the `fig` format files into a more suitable format by using `fig2dev`. When *export* option is used, the `fig2dev` command is invoked automatically and silently by `xfig` to carry out the relevant conversion of formats.

Once the drawing has been converted into PostScript or EEPIC, the resulting file needs to be included in a LaTeX document.

2.1.3 Including EEPIC figures in LaTeX

To include an EEPIC file in your LaTeX document you use the EEPIC and EPIC style files. To do this you must first input the relevant files:

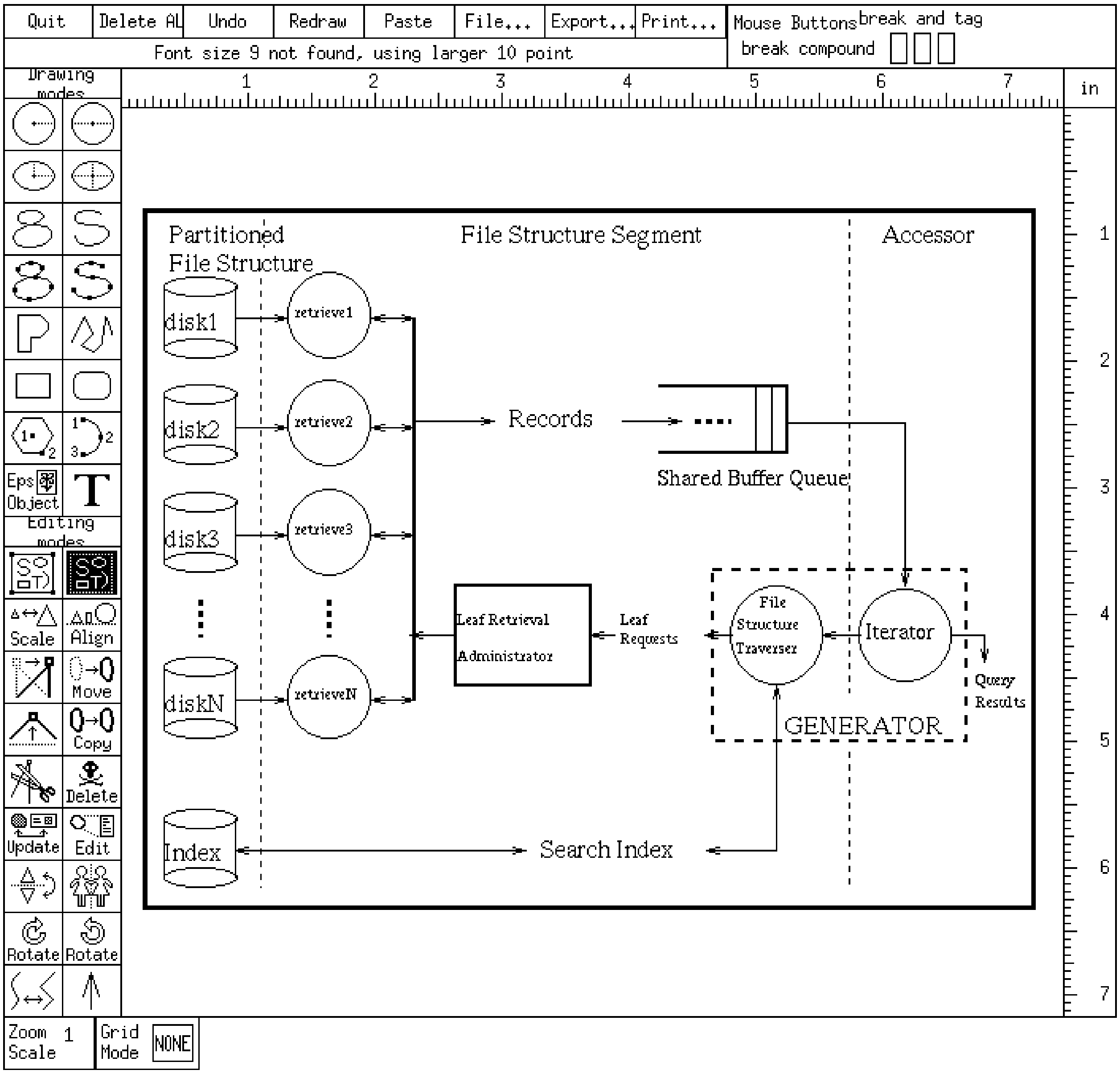
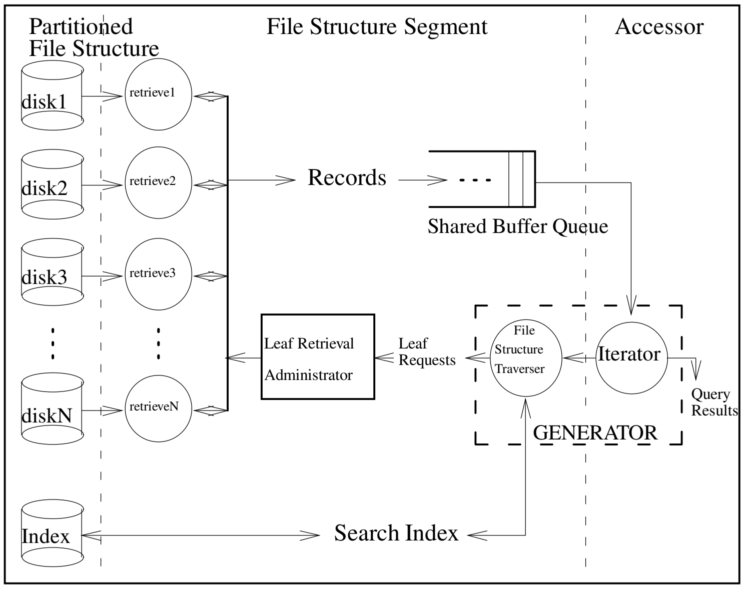
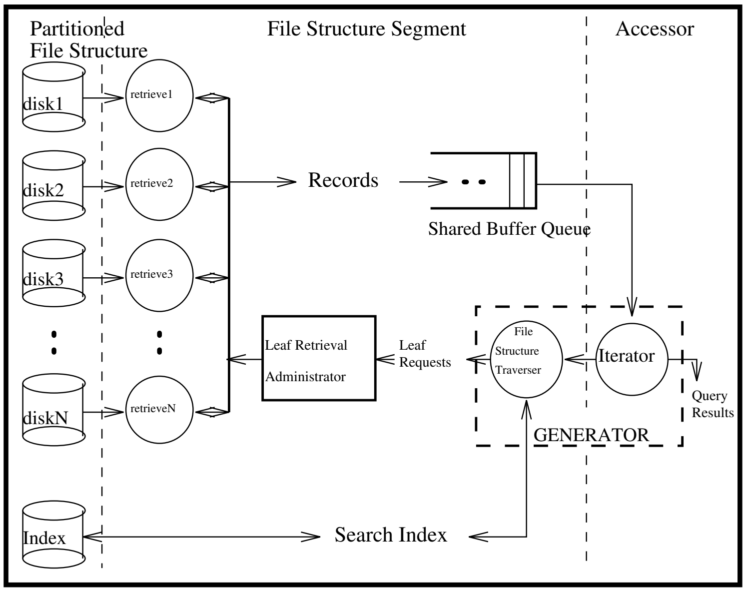


Figure 1: A Sample `xfig` Screen



(a) model.tex included via EEPIC (xdvi will display this)



(b) model.ps included via psfig (xdvi will *not* display this; use ghostview)

Figure 2: Sample outputs from `xfig`

```
\documentstyle[...,epic,eepic,nfssfontcompat]{...}

\begin{document}

⋮
```

Then you may include the EEPIC file using the `\input` macro:

```
\input model.tex
```

2.1.4 Including PostScript pictures in LaTeX

To include a PostScript file in your LaTeX document you use the psfig [3] macro. To do this you must first `\input` psfig:

```
\documentstyle[...]{...}
\input{psfig}

\begin{document}

⋮
```

Then you may use the `\psfig` macro to actually include a PostScript file:

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

Psfig depends upon the presence of a BoundingBox comment in the PostScript file being included. A BoundingBox statment essentially specifies the location and size of the included figure on the page. See [3] for a detailed explanation. In most cases the program generating the PostScript figure will automatically include a BoundingBox comment in the file and you won't have to worry about it. Psfig will use the BoundingBox comment in the PostScript file to scale the included figure to the given dimensions. You can change the *natural* scaling of the figure by specifying `height` or `width` or both in the `\psfig` statement above. If you only specify one, scaling will be the same in both directions in order to produce that one dimension. If you want to include drawings produced on a Mac you should see the `psfig` man page for special instructions.

An alternative set of macros that can be used to include Encapsulated PostScript files in a LaTeX document is called EPSF. Details on how to use the EPSF macros are contained in [15].

A special note about including PostScript files generated by Maple

Maple [2] generated PostScript plots need special mention because of a problem in the relevant driver of the current Maple release. This problem may be rectified in a future release of Maple. Until then, if you want to include a Maple generated PostScript plot in your LaTeX document you have to first modify the plot file by commenting out the following three lines.

```
% 540 82 translate
% 90 rotate
% 0.19 0.19 scale
```

Note that `%` is the comment character in PostScript. The plot can then be included in the LaTeX document by following the instructions given earlier in section 2.1.4. Another thing that will need to be done for getting the plot scaled properly is to specify the dimensions of the plot by using the height, width or both parameters in the `psfig` command, as in

```
\psfig{figure=maple_plot.ps,height=2.5in}
```

If this is not done the resulting plot may be too big since it will be printed in the default Maple size. The instructions described in this section were used to produce the 3-dimensional plot in Figure 3 from a file generated by maple for `plot3d(binomial,0..5,0..5)`.

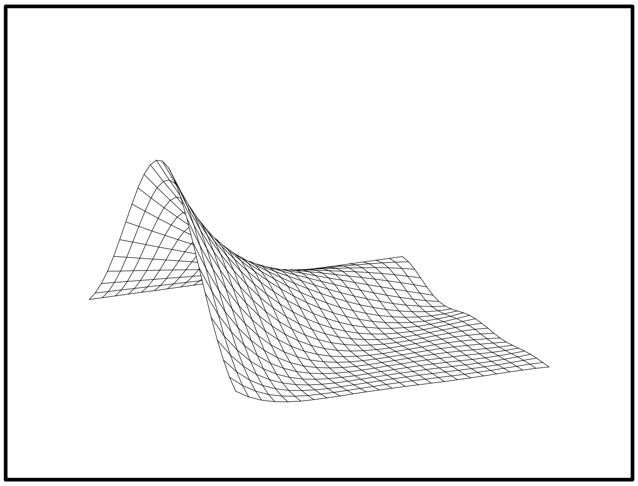


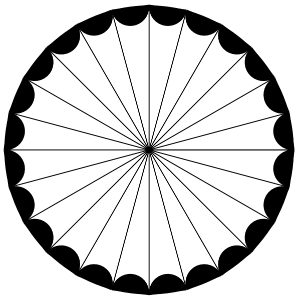
Figure 3: An example Maple plot

2.2 A Picture Drawing Language - PostScript

While it is true that most PostScript programs are generated by other programs, it is quite easy to generate complex pictures by writing programs in the PostScript language. Figure 4 gives an example of a small PostScript program alongwith the drawing it creats.

2.2.1 Converting and including PostScript drawings

There is no conversion required on a PostScript picture since it can be included directly with the assistance of the `psfig` macros. Section 2.1.4 gives the details on including a PostScript file in a LaTeX document.



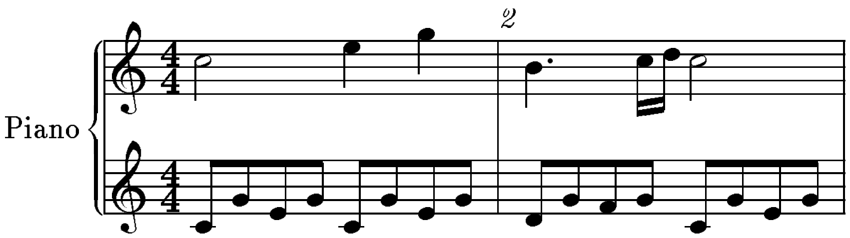
```
/drawwheel { % diameter drawwheel --
             % draws a wheel at currentpoint, of given diameter
    2 div /rad exch def % get radius
    gsave

    currentpoint translate
    0 0 rad 0 360 arc stroke        % draw the wheel outline
    % compute radius of the small circles hanging off the wheel..
    /smallrad rad 7.5 sin mul def
    /smallcx rad 7.5 cos mul def

    0 15 345 { % draw the 24 spokes
        gsave
        rotate % use the for loop variable value, 0..245
        0 0 moveto rad 0 lineto stroke % draw spoke
        smallcx smallrad -1 mul smallrad 82.5 262.5 arc fill
        grestore
    } for
    grestore
} def

% draw the wheel ...
200 200 moveto 300 drawwheel
```

Figure 4: A sample PostScript program and the wheel it draws

2.3 A Macro Package - MusicTEX

Macro packages are written in TEX or LATEX and are usually available in the form of style files that need to be included in a LATEX document before a drawing made by using the macros can be included. The use of `psfig` macro package has already been illustrated. This section illustrates the use of MusicTEX macros for drawing musical scores.

Here is a MusicTEX example taken from the MusicTEX manual [16]. Figure 5 is an example of the two first bars of the sonata in C-major K545 by MOZART :




```
The coding is set as follows :
\documentstyle[musictex,...]{...}

\begin{music}
\parindent 1cm
\def\nbinstruments{1}\relax % a single instrument
\def\instrumenti{Piano}%    % whose name is Piano
\nbporteesi=2\relax         % with two staffs
\generalmeter{\meterfrac{4}{4}}\relax % 4/4 meter chosen
\debutextrait               % starting real score
\normal                     % normal 12 pt note spacing
\temps\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\hl j\enotes
\temps\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\ql l\sk\ql n\enotes
\barre                      % bar
\Notes\ibu0f0\qh0{dgf}|\qlp i\enotes
\notes\tbu0\qh0g|\ibbl1j3\qb1j\tbl1\qb1k\enotes
\temps\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\hl j\enotes
\finextrait                 % terminate excerpt
\end{music}
```

Figure 5: A MusicTEX example

2.4 Other macro packages

Some other macro packages available at Waterloo include:

pstricks [19] TEX fonts and macros for fancy PostScript features.

nassflow [18, 17] LaTeX macros for flowcharts and Nassi-Schneiderman diagrams.

A list of other useful packages (e.g. bipartite graphs, trees, etc.) can be found in [8]. Not all of the packages mentioned in [8] are installed at Waterloo.

Before printing you will want to preview your document. Your choice of a previewer will depend on your choice of macro-language used to include the drawings. More on this later.

3 More About Macros

The examples in section 2 illustrated that some macro package is used to make LaTeX understand the type of drawing being included. Some of these macros process the input and convert it to LaTeX primitives. Others merely tell LaTeX about the type and size of the input drawing and assume that some output device driver (display device or printer) will be able to render it. LaTeX then leaves enough blank space for the figure to be drawn.

In the `xfig` example, note that the user has a choice of macro packages. There are several factors that affect the choice you should make: image quality, portability, size, processing time, etc. For example, `fig2dev` will attempt to render a figure in LaTeX as best as possible, but the quality of the resulting figure may be quite poor due to many LaTeX limitations such as restricted line orientations. On the other hand, such a document could be printed in any environment that supports LaTeX. A PostScript version may render the original figure accurately, but now the user is assuming that whoever wants to print the document has access to all the necessary macros and device drivers for processing and converting the PostScript input resulting in a potential lack of portability. An intermediate solution lies in using the EEPIC set of macros which will generate much better results than plain LaTeX and at the same time be extremely portable.

4 Viewing the Results

The output of LaTeX is in a format called *device independent* (DVI). In order to view (on a display device or on paper) a DVI document, one has to first convert the document to the device (terminal or printer) dependent language. The usual UNIX printer command `lpr -Fd`, where `-Fd` indicates that the input is a DVI file, converts the DVI specification to commands which the printer understands. Similarly, there are several commands for converting the DVI document into commands which render the document on a graphics terminal, in particular under the X11 window system.

Recall that when including a PostScript figure, LaTeX does not translate the input. Hence, in general, if you send the DVI file to a DVI processor without telling it about the postscript it will simply leave a blank space where the figure belongs. An extra conversion step is needed when including non-LaTeX input in your file.

In the `xfig` example in section 2.1.3, EEPIC was used to describe the image. Hence, `xdvi`, a DVI previewer for X11 window system is able to display the figures. Similarly, the musicTeX example in section 2.3 uses TeX fonts and macros so `xdvi` is able to display the musical scores on an X11 terminal [1]. On the other hand, in the PostScript example in section 2.1.4, LaTeX has left the figure in its native PostScript. `xdvi` does not understand PostScript so it just leaves enough blank space for the figure.

In order to view a LaTeX document with postscript figures the DVI file must be translated into a postscript file. Now the whole document is in PostScript and can be understood by any device driver which handles PostScript input.

At Waterloo `dvips` [15] is the standard dvi to postscript converter. See `man dvips` for details on how to convert a dvi file into PostScript.

Also note that most of the existing laser printers at Waterloo are PostScript printers. This means that any dvi file produced by LaTeX has to be converted to PostScript before it can be printed (even if the LaTeX document did not contain any PostScript). In most cases, this conversion is performed automatically by the `lpr` command when used with `-Fd` option. The conversion is done by silently invoking `dvips`.

4.1 Previewing Files

As mentioned before, the `xdvi` command can be used to look at dvi files generated by LaTeX as long as they don't contain any PostScript (EEPIC is okay). If xdvi fails to display the file, you will need to convert the dvi file into PostScript (by using `dvips`) and then use the `ghostview` command to look at the resulting PostScript file. Ghostview has a very nice user interface and has a number of features including viewing and printing of randomly selected pages. If you only want to look at the document once in order (no random selection of pages), you can avoid creating the intermediate PostScript file by typing the following:

```
dvips yourfile.dvi | ghostview -
```

In the absence of `ghostview` on your machine, you can use `gs` instead. It is not a very user friendly program. Type `^C` to escape.

5 Other Useful Applications

5.1 Images

A paper image can be digitized and stored in PostScript format by using the scanner available in the DCS I/O room (MC 1063). The image on the title page was produced in this way.

5.2 Window dumps

Window dumps produced using `xwd` can be converted to PostScript using the `xpr` command. The window dump of the `xfig` window in Figure 1 was produced using the command:

```
% xpr -device ps -portrait -psfig -compact xfig.xwd > xfig.ps
```

but in most cases you can save some disk space by piping `xwd` to `xpr` directly:

[1] assuming `xdvi` knows where to find the fonts

```
% xwd | xpr -device ps -portrait -psfig -compact > xfig.ps
```

Note that the `-psfig` option is necessary in order to have the figure center correctly in your LaTeX document. The `-compact` option saves some disk space in most cases.

5.3 Special effects

5.3.1 Scaling and rotating

Postscript input can be arbitrarily scaled and rotated using the `\psfig` macros. See the `psfig` documentation for complete details.


Figure'ing and Picture'ing LaTeX
Combining text, figures, and images in a LaTeX document
by
Anil Goel ©1993
(akgoel@math.uwaterloo.ca)
Version 1.0
last modified
July 5, 1993
Math Faculty Computing Facility
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Figure 6: The Title Page of this Document

For example, you can include a page of one of your LaTeX documents in another document with any desired scaling and rotation. The title page of this document is reproduced in Figure 6. The `dviselect` command was used to extract the one page and dvips was used to convert it to PostScript. The resulting PostScript file was then included:

```
\centerline{\hbox{\psfig{figure=title.ps,height=3.0in,angle=45}}}
```

5.3.2 Using PostScript fonts

You may have noticed by now that this document has been typeset with PostScript fonts as opposed to the Computer Modern family of fonts usually employed by LaTeX . The recommended mechanism for using PostScript fonts is the `psnfss` macro package available at Waterloo. See `man psnfss` and [14] for details.

6 Using `Make` to Make Life Easy

This document was produced using a Makefile which takes care of converting the various files into one single document. The Makefile is reproduced in Appendix A.

7 Acknowledgements

This document has borrowed liberally from [13], a smaller document Lindsay Patten wrote for the PAMI Lab users at University of Waterloo. The PostScript example in section 2.2 is taken from one of the example programs packaged with the `itrans` package for Indian languages.

Thanks are due to Bill Ince and Peter Yamamoto who read the first draft and offered several valuable suggestions. Peter even keyed in many of his suggested additions which were gratefully accepted for inclusion in this document.

Please send comments to akgoel@math.uwaterloo.ca.

References

[1] Micah Beck. *TransFig: Portable Figures for* LaTeX. In `/software/fig/doc/transfig.dvi.Z`.

[2] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, and S.M.Watt. *Maple V Language Reference Manual*. Springer-Verlag and Waterloo Maple Publishing, 1991.

[3] Trevor Darrell. *Psfig/TeX 1.8 Users Guide*. In `/software/tex_documents/doc/formatted/psfig-doc.ps.Z`.

[4] Adobe Systems Inc. *Encapsulated PostScript*. In `/software/postscript/doc/encapsulated.ps`.

[5] Adobe Systems Inc. *Summary of PostScript statements*. In `/software/postscript/doc/pssummary.ps`.

[6] Adobe Systems Inc. *PostScript Reference Manual*. Addison Wesley, 1985.

[7] Adobe Systems Inc. *PostScript Tutorial and Cookbook*. Addison Wesley, 1985.

[8] David M. Jones. *An index of TeX macros*. In /software/tex_documents/doc/source/tex-index.Z.

[9] Donald E. Knuth. *The TeXbook*. Addison Wesley, 1987.

[10] Conrad Kwok. *EEPIC: Extensions to epic and* LaTeX *Picture Environment*. In `/software/tex_documents/doc/formatted/eepic.dvi.Z`.

[11] Leslie Lamport. LaTeX *User's Guide and Reference Manual*. Addison Wesley, 1986.

[12] Anil Goel (original by Leslie Lamport). *Using LaTeX at Waterloo*.
In `/software/tex_documents/doc/formatted/LocalGuide.dvi.Z`.

[13] Lindsay Patten. *Using PostScript within LaTeX*. In
`watnow:/system/watnext/software/share/pami/contrib/psdvidocs`.

[14] Sebastian Rahtz. *Notes on setup of the NFSS to use PostScript fonts*.
In `/software/tex_documents/doc/formatted/psnfss.dvi.Z`.

[15] Tomas Rokicki. *DVIPS: A TeX Driver*.
In `/software/tex_network/doc/dvips.dvi.Z`.

[16] Daniel Taupin. *musictex: Using TeX to write polyphonic or instrumental music*.
In `/software/tex_documents/doc/formatted/musicdoc.dvi.Z`.

[17] Marion van Geest. *FLOW: Typeset Flow diagrams in LaTeX*.
In `/software/tex_documents/doc/formatted/flow_man.dvi.Z`.

[18] Marion van Geest. *NASSI: Typeset Nassi-Schneiderman diagrams in LaTeX*.
In `/software/tex_documents/doc/formatted/nass_man.dvi.Z`.

[19] Timothy Van Zandt. *PSTricks documentation and examples*.
In `/software/tex_documents/doc/source/pstricks`.

A Makefile

```
MAINFILE = figsInLatex
FIGS_TO_MAKE = xfig.ps model.ps model.tex
PSFILES = face.ps wheel.ps

# Everything below this line can stay the same, customize above

.SUFFIXES: .fig .tex .dvi .ps .xwd .bib .bbl

.dvi.ps:
	dvips $*.dvi > $*.ps

.fig.ps:
	fig2dev -L ps $*.fig > $*.ps

.fig.tex:
	fig2dev -L eepic $*.fig > $*.tex

.bib.bbl:
	if [ -r $*.aux ] ; then . ; else latex $*.tex ; fi
	bibtex $*

.xwd.ps:
	xpr -device ps -portrait -psfig -compact $*.xwd > $*.ps


all: $(MAINFILE).ps

preview: $(MAINFILE).ps
	ghostview $(MAINFILE).ps

printout: $(MAINFILE).ps
	lpr -Fl -Pljp_mfcf $(MAINFILE).ps

$(MAINFILE).dvi: $(MAINFILE).tex $(FIGS_TO_MAKE) $(PSFILES) \
	$(MAINFILE).bbl
	latex $(MAINFILE)
	dviselect 0 $(MAINFILE).dvi > title.dvi
	dvips title.dvi > title.ps
	latex $(MAINFILE)

clean:
	/bin/rm -f $(MAINFILE).ps $(FIGS_TO_MAKE)
	/bin/rm -f *.aux *.dvi *.bbl *.blg *.log
```