texpic – Design and Implementation of a Picture Graphics Language in TEX à la pic

ROLF OLEJNICZAK-BURKERT

Eiselauer Weg 12
D-7901 Beimerstetten
West Germany

ABSTRACT

texpic is a TEX implementation of a graphics language similar to Kernighan's *troff* pre-processor *pic*.

Many features of the original *pic* are supported, including a variety of graphical objects (boxes, circles, ellipses, lines, arrows and others), directions of motion, controlling sizes of objects with variable and appropiate defaults, relative and absolute positioning of single objects or whole pictures (labels and corners are allowed), and much more.

There are two significant enhancements. Objects adapt to the size of their contents; that is, a circle may contain a table with mathematical equations, a box may contain the circle, etc. *texpic* objects and TEX or LATEX commands may be combined at will.

The implementation consists of two parts, a set of elaborate TEX macros and a post-processor for drawing (in the `dvi` file), written in C. It should be emphasized that *texpic* is fully portable, i.e., every TEX version, every preview and even every (correctly written) printer driver will work together with *texpic*.

1. Preface

Some years ago I attended a lecture on text processing. At that time I had just discovered TEX and was filled with enthusiasm, but unfortunately the lecture dealt mainly with another system: the *troff* typesetting software, widely used under UNIX.

There ensued a friendly competition between the lecturer and me — with the goal being to typeset things the other one couldn't do. One time he won, another time I made a point, so the race was rather even.

With introducing *pic* one day, a powerful, but easy to use language for drawing pictures, implemented as a pre-processor to *troff*, the tables turned. Because TEX has little to retort, I began to lose very often. To catch up, I decided to implement something similar in TEX, not knowing what frustration (and fun) this would be!

2. Boxes — The Cornerstone of TEX

Boxes are probably the only objects which are easy to implement *in* TEX. This is because TEX also uses a box concept which offers two possibilities. If we have specified width and height explicitly, we obtain just a box with these dimensions. Otherwise the smallest box is chosen which fits around its contents. For the frame of the box we need only horizontal and vertical lines — suitable commands already exist. Consequently we require the following:

- Boxes have a minimum size.
- Between contents and frame there is a certain amount of free space.
- Boxes adapt to the size of their contents.
- Boxes are centered perpendicular to the current direction of movement.
- Minimum size, free space and the thickness of the lines are locally or globally changeable.

The resulting TEX macros are relatively straightforward. Producing a box with *texpic*, the complete syntax of the corresponding command is:[1]

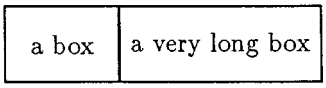
`\tpbox [`*attributes*`] [`*parameters*`] [`*contents*`];`

An *attribute* such as `invis` describes a quality and is typically one word, whereas a *parameter* such as `width 3cm` influences the size of an object and consists of several words. Finally, the *contents* begin behind the last *parameter* or *attribute*, stop at the next semicolon and are often ordinary text. Subsequent sections will illustrate this.

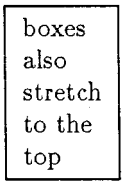
2.1 "Growing" Boxes with Minimum Size


```
\tpbox; a box
```

As we can see, the box is centered on an imaginary horizontal line.




```
\tpbox a box;
\tpbox a very long box;
```



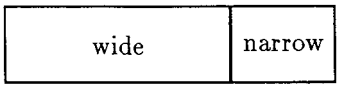

```
\tpbox \vbox{
   \hbox{boxes}
   \hbox{also}
   \hbox{stretch}
   \hbox{to the}
   \hbox{top}
};
```

In the next example the current direction of movement is vertical which changes the centering of the box:

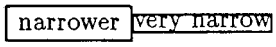
```
\par
\tpbox;
```

2.2 Local and Global Changes

With *parameters* we can change various sizes of *one* object:

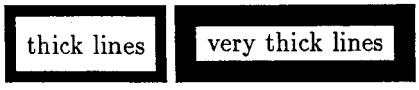



```
\tpbox width 3cm wide;
\tpbox width 1cm narrow;
```




```
\tpbox width 0cm height 0cm narrower;
\tpbox height 0cm width 0cm fill 0cm
   very narrow;
```

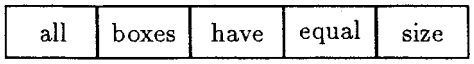
Parameters which control the size of an object, control only the minimum size, i.e. if the contents don't fit, the object will still grow. The space between frame and contents is changed through `fill`. The `thickness` of the lines is also adjustable:



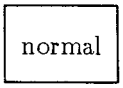

```
\tpbox thickness 4pt thick lines; \
\tpbox thickness 8pt very thick lines;
```

[1] "tp" as a prefix for all names relating to *texpic* and should avoid name conflicts.

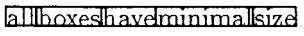
To achieve global changes we can simply change corresponding variables. As usual, braces control the scope:




```
{
   \tpboxwd=35pt \tpboxht=20pt
   \tpbox all; \tpbox boxes; \tpbox have;
   \tpbox equal; \tpbox size;
}
```



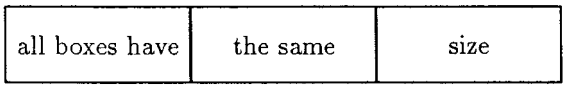

```
\tpbox normal;
```




```
{
   \tpboxwd=0pt \tpboxht=0pt \tpboxfill=0pt
   \tpbox all; \tpbox boxes; \tpbox have;
   \tpbox minimal; \tpbox size;
}
```

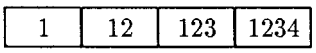
2.3 Sharing Attributes

With the attribute `same` we can make an object have the same size of the last one, provided that the contents fit:



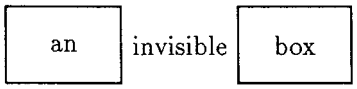

```
\tpbox all boxes have;
\tpbox same the same;
\tpbox same size;
```

The first `same` is the attribute, the second is ordinary text! `\tpphantom` can be used, if the biggest box is not the first one:




```
\tpboxwd=0in \tpboxht=0in
\tpphantom{\tpbox 1234;};
%
\tpbox same 1;    \tpbox same 12;
\tpbox same 123; \tpbox same 1234;
```

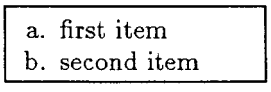
With `invis` we can make an object invisible, i.e. we suppress the frame. This attribute will prove useful later, when we want to position objects at different places:



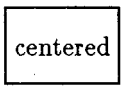

```
\tpbox an;
\tpbox invis invisible;
\tpbox box;
```

2.4 Boxes Around Other Objects

More complicated examples are possible — boxes are *bona fide* members of the TEX world:

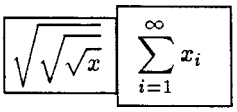



```
\tpbox
   \hbox to 3cm{\vbox{
      \item{a.} first item
      \item{b.} second item
   }\hss}
;
```




```
\centerline{
   \tpbox centered;
}
```

Notice, that \item would use the entire line, therefore \hbox is used to limit line length. Similarly, \centerline pushes the \tpbox to the middle.



```
\tpbox
   $\displaystyle\sqrt{\sqrt{\sqrt{x}}}$
;
\tpbox
   $\displaystyle\sum_{i=1}^{\infty}x_i$
;
```

```
\tpboxwd=0pt \tpboxht=0pt \tpboxfill=1pt
\tpbox{\tpbox{\tpbox{\tpbox{\tpbox{%
\tpbox{\tpbox{\tpbox;};};};};};};};
```

The last example shows a particularly valuable feature: nesting. Most LaTeX macros also work with *texpic* boxes. So a box around a **tabular** or a box inside a **tabular** can be used. This is very useful for positioning.

3. Circles — Do they have to be so special?

Now on to the circles which should provide exactly the same features as the boxes above. As we will see, however, circles are much more complicated than boxes.

3.1 Two Dead Ends

To draw circles there are two approaches which will not work, at least not very well or with considerable restrictions:

1. Drawing in TeX is possible, but this is very slow and there are also limitations regarding the number of circles, i.e., points, on the same page. See also the preface from the PiCTeX Manual.
2. Use of a printer language for drawing is possible with the \special command, though this means a commitment to one printer and therefore a loss of portability.

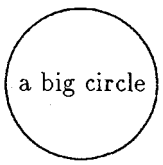
The second solution would be sufficient at the moment, but as in the original *pic*, references to objects should eventually be implemented. Because there is no way to get the current coordinates on the page in TeX, we could have transfered this problem to the printer language as well. However, this would certainly not improve any portability aspects.

3.2 A Post-Processor for Drawing

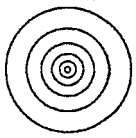
Looking for other ways to obtain the coordinates of an object we discover the **dvi** file which is absolutely device independent. Reading this file (and some **tfm** files to get the widths of single characters) we are able to track the current position.

The main point, however, is that we can draw in the **dvi** files. This is a bit subtle, since we must pay attention to some pointers. With the use of the \special command and a post-processor written in C, the same features as for boxes are possible:

```
\tpcircle;
```




```
\tpcircle a big circle;
```



```
\tpcircrad=0pt \tpcircfill=1pt
\tpcircle{\tpcircle{\tpcircle{%
\tpcircle{\tpcircle{\tpcircle;};};};};};
```

4. Directions and Movements — Not quite the same

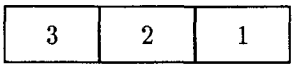
Changing the current direction of movement in *pic* is possible at any time — "north", "south", "west" and "east" are allowed. Besides that, we can change the current point by a movement. Both features can be implemented in TEX, however, with some restrictions.

4.1 Directions

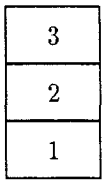
Because we want to allow arbitrary objects, we require the following points:

- All four points of the compass are allowed.
- Macros `\tp...begin` and `\tp...end` enclose the objects of one "row".
- Inside a "row" all arbitrary objects are possible.
- Every single object must be surrounded by braces.

In the following examples the default sizes have been decreased a little:

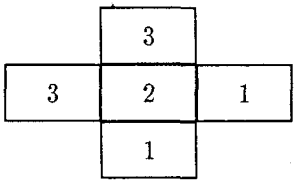



```
\tpleftbegin
   {\tpbox 1;} {\tpbox 2;} {\tpbox 3;}
\tpleftend
```

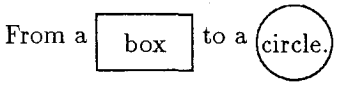



```
\tpupbegin
   {\tpbox 1;} {\tpbox 2;} {\tpbox 3;}
\tpupend
```

Directions can be combined with other objects as usual:




```
\tpleftbegin
   {\tpbox 1;}
   {\tpupbegin
      {\tpbox 1;} {\tpbox 2;} {\tpbox 3;}
   \tpupend}
   {\tpbox 3;}
\tpleftend
```




```
\tprightbegin
{From a } {\tpbox box;}
{ to a } {\tpcircle circle.;}
\tprightend
```

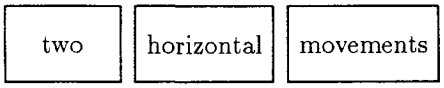
Especially for positioning objects these features are very useful.

4.2 Movements

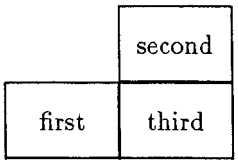
An arbitrary change of the current position is not possible in TEX, therefore the design of movements is poor and rather restricted:

- A single `\tpmove` changes the reference point for a default value in the current direction.
- Specifying an optional direction moves only the next object.
- All default values are changeable locally and globally.

The first example shows the "normal" use of `\tpmove`, the second moves only one object:




```
\tpbox two;
\tpmove width 5pt;
\tpbox horizontal;
\tpmove same;
\tpbox movements;
```




```
\tpbox first;
\tpmove up {\tpbox second;};
\tpbox third;
```

The `\tpmove` command in the last example does not change the reference point!

5. Arrows and Corners — Tying objects together

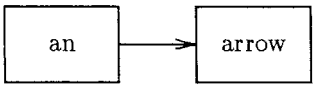
As said before, *pic* supports a "link" mechanism:

```
... to 3rd last circle ...
```

Since the actual position on a page is not available, this feature cannot be implemented in TEX. Because we are already using a post-processor for drawing circles, it is not very difficult to extend the C program to store the positions of the objects. The communication is done again with the `\special` command of TEX.

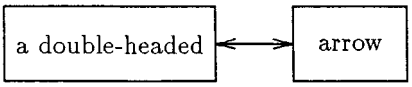
5.1 Arrows

To work not only with lines we implement arrows:

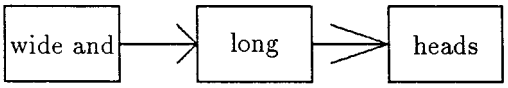



```
\tpbox an;
\tparrow;
\tpbox arrow;
```

TEX does not support slanted lines and LATEX does not permit arbitrary slopes. Therefore, the arrowhead is drawn by the post-processor. There are two new *parameters* and one new *attribute* relating to the arrowheads:




```
\tpbox a double-headed;
\tparrow double;
\tpbox arrow;
```




```
\tpbox wide and;
\tparrow headwidth 0.2in; \tpbox long;
\tparrow same headheight 0.3in; \tpbox heads;
```

5.2 Links

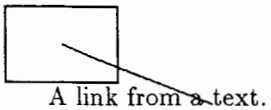
Links to objects are much better than coordinates for connecting objects with lines or arrows. Because the original syntax of *pic* is not ideal for scanning, I changed the syntax slightly from `line from 2nd box to 3rd circle` to `\tpline from 2.box to 3.circle;`:

```
\tpcircle; \par \hskip 3cm \tpcircle;
\tpline from 1.circle to 2.circle;
```

Counting up results in absolute links. Relative links are constructed by counting backwards:

```
\tpcircle; \par \hskip 3cm \tpcircle;
\tparrow from 2.circle to 3.circle;
\tparrow from -2.circle to -1.circle;
```

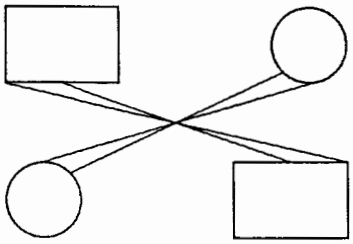
If one link is missing the current position is used:




```
\tpbox; \par
A link from a
\tpline from 1.box;
text.
```

5.3 Corners

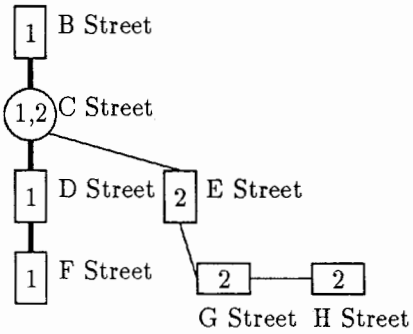
Links can even refer to eight compass points on the perimeter of an object:

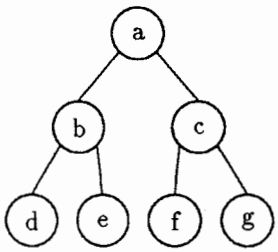


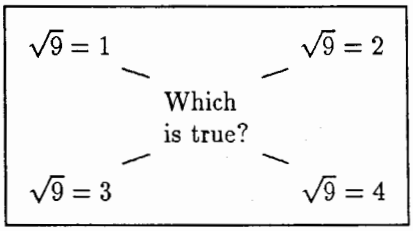
```
\tpbox; \hskip 2cm \tpcircle;
\par \vskip 1cm
\noindent \tpcircle; \hskip 2cm \tpbox;

\tpline from -1.box.n to -2.box.s;
\tpline from -1.box.ne to -2.box.sw;
\tpline from -1.circle.n to -2.circle.s;
\tpline from -1.circle.ne to -2.circle.sw;
```

As you can see, circles also have "corners". With these features fancy pictures become possible. However, they require too much code to be shown here:

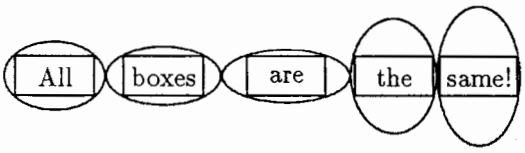




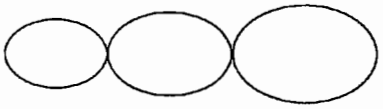


6. Ellipses — Circles with a catch

Unfortunately, ellipses differ considerably from circles since there is not just *one* smallest suitable ellipse around an object:

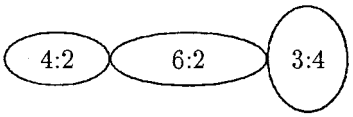


Because an ellipse has *two* major axes, it seems reasonable to require a fixed ratio for them:



```
\tpboxwd=30pt \tpboxht=15pt
%
\tpellipse {\tpbox invis;};
\tpellipse {\tpbox invis width 40pt;};
\tpellipse {\tpbox invis height 30pt;};
```

If width or height are specified, the shape of the ellipse will change:

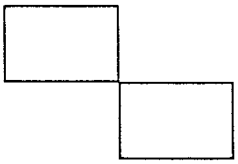



```
\tpelliwd=30pt \tpelliht=20pt \tpellifill=0pt
%
\tpellipse width 40pt 4:2;
\tpellipse width 60pt 6:2;
\tpellipse height 40pt 3:4;
```

Within each ellipse the ratio of its axes is displayed.

7. Shifted Objects — With and without size

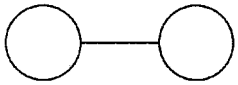
Sometimes it is useful to move whole objects. To do this, there are two new attributes: `with` and `at`. Unlike `at`, `from`, and `to`, `with` does not permit a link:



```
\tpbox;
\tpbox with .nw at -1.box.se;
```

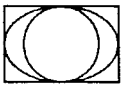
Again with the C post-processor, the implementation is simple: only one change in position has to be made. But there is a problem: objects which are to be moved must be set without any size; otherwise, they will need some place on the page and after being moved this place would be empty! So the user has to worry about the space.

The "corners" of lines and arrows are abbreviated with 's' for 'start' and 'e' for 'end':



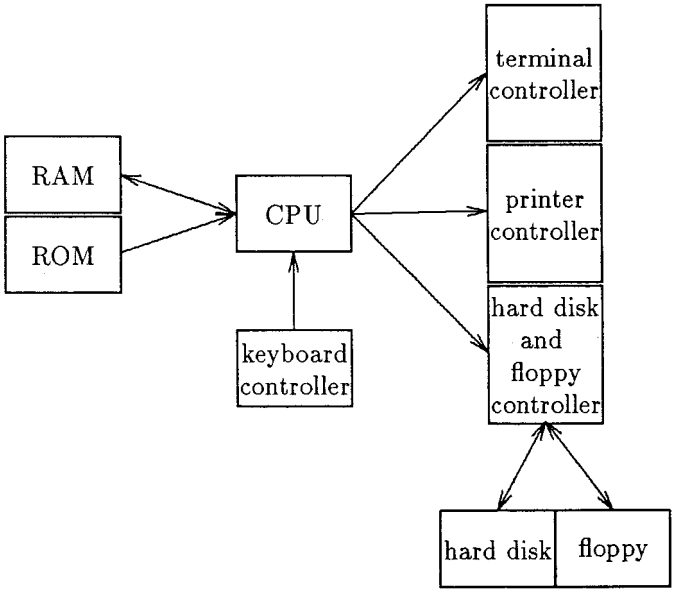
```
\tpbox invis;
\tpline;
\tpcircle with .e at -1.line.s;
\tpcircle with .w at -1.line.e;
```

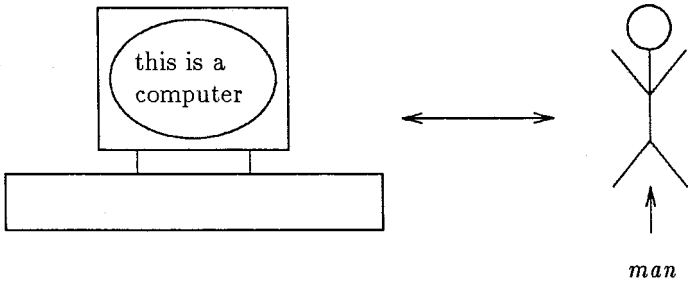
Without the `with` attribute, the center of the desired object is used:



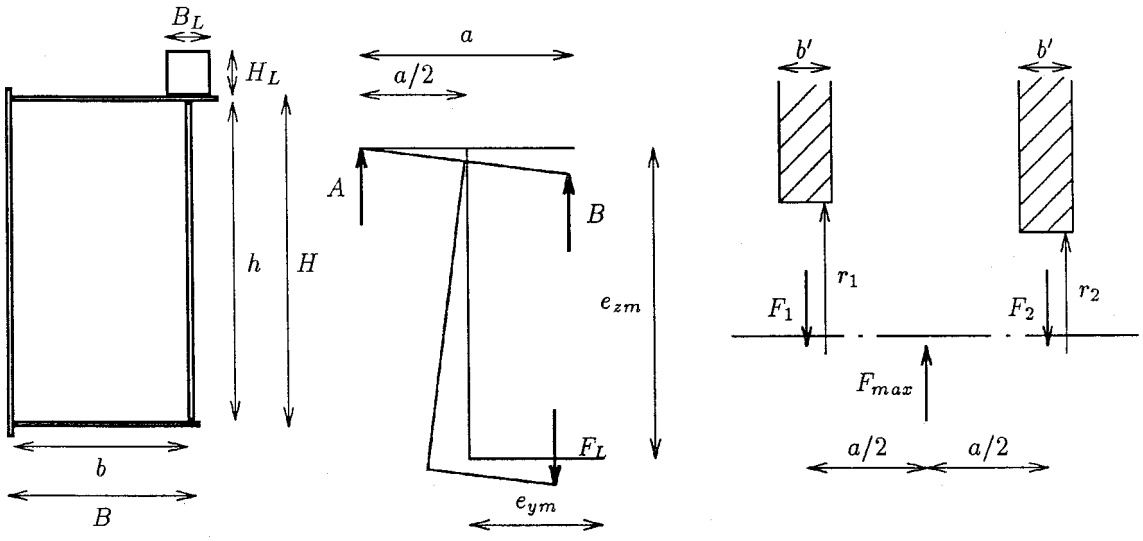
```
\tpbox;
\tpcircle at -1.box;
\tpellipse at -1.box;
```

Once again, more elaborated pictures are possible:





A friend of mine, a mechanical engineer, supplied the following pictures, which I would never have thought possible:



8. Conclusion

8.1 What has been done

Using *texpic* simple pictures in the style of *pic* can be drawn within a TEX document. Graphical objects have been implemented which may be used with several attributes and positioned in different ways. *pic* syntax was modified slightly to accomodate TEX conventions. Furthermore, there are two significant enhancements:

- Objects adapt to the size of their contents.
- The contents may be almost anything.

This results in a very smooth integration of text and line drawings. Through a C program as a post-processor operating on the `dvi` file, we achieved a very portable and absolutely device independent solution. Some points of the original *pic* were not implemented:

- *pic* itself serves as a target language for other pre-processors (*grap*, *chem*, etc.) New features in *texpic*, however, will most likely have to be constructed within TEX as well.
- In *pic* a picture can be scaled to near arbitrary dimensions. I see no way to do this in TEX.

8.2 What can be done

A few more features are probably practical:

- Generalization of the corners, for example `nnw` or `sssee`. This requires only a little bit of mathematics.
- Arcs of a circle and splines. This is possible with some mathematics and the C program.

- References to the dimensions of an object, e.g.:

```
... -1.box.ht ...
```

- Local scopes for objects. This requires an extension to the management of object stacks.
- Labels for objects or whole pictures, as in:

```
\tpbox name ellipsoid;
\tpline from ellipsoid.w ...
```

- Coordinates with addition and subtraction. This is very simple, because we already have the coordinates in the C program. The only thing to do is to build an interface to TEX, e.g.:

```
\tpbox with .n at -1.box.s minus (12,15);
```

- Interpolating a point, e.g.:

```
\tparrow from <1/3,-1.box.n,-2.box.s> ...
```

  There is a syntactical problem: a link must consist of one word.
- Projecting object coordinates, e.g.:

```
\tparrow from (1.box.s,-2.ellipse.n) ...
```

- Printing and positioning text. The ideal would be along the lines of "printf", because this is simple to implement in C.

8.3 What might be done

I am afraid the following features would be rather difficult to implement:

- Grids with automatic scaling. There is a question: what should a good grid look like?
- Drawing arbitrary functions. This requires all sorts of mathematical and syntactical support.
- Simple graphics in the style of *grap*, a pre-processor of *pic*.
- Rotation of objects. This would result in substantial changes since then every object must be drawn by the post-processor.

It is interesting to note that further refinement of features appears to shift more and more responsibity out of TEX and on to the post-processor. Is the ideal solution a graphical co-processor to TEX?

Bibliography

Adobe Systems Incorporated. *PostScript Language — Tutorial and Cookbook*. Reading, Mass.: Addison-Wesley, 1985.

Adobe Systems Incorporated. *PostScript Language — Reference Manual*. Reading, Mass.: Addison-Wesley, 1985.

Aho, Alfred V., Brian W. Kernighan, and Peter J. Weinberger. *The AWK Programming Language*. Reading, Mass.: Addison-Wesley, 1988.

Appelt, Wolfgang. *TEX für Fortgeschrittene*. Bonn: Addison-Wesley, 1988.

Bentley, J.L. and Brian W. Kernighan. "grap — A Language for Typesetting Graphs." *CACM*. August 1986.

Elan Computer Group. "pic — Reference Manual".

Foley, J.D. and A. Van Dam. *Fundamentals of Interactive Computer Graphics*. Reading, Mass.: Addison-Wesley, 1982.

Hearn, D. and M.P. Baker. *Computer Graphics*. Reading, Mass.: Addison-Wesley, 1986.

Jordan, B.W., W.J. Lennon and B.D. Holm. "An Improved Algorithm for the Generation of Nonparametric Curves." *IEEE Transactions on Computers*. December 1973.

Kernighan, Brian W. "*pic* — A Language for Typesetting Graphics." *Software — Practice and Experience*. January 1982.

Knuth, Donald E. *The TEXbook*. Reading, Mass.: Addison-Wesley, 1986.

Knuth, Donald E. *TEX: The Program*. Reading, Mass.: Addison-Wesley, 1986.

Kopka, Helmut. *LaTeX — Eine Einführung*. Bonn: Addison-Wesley, 1988.

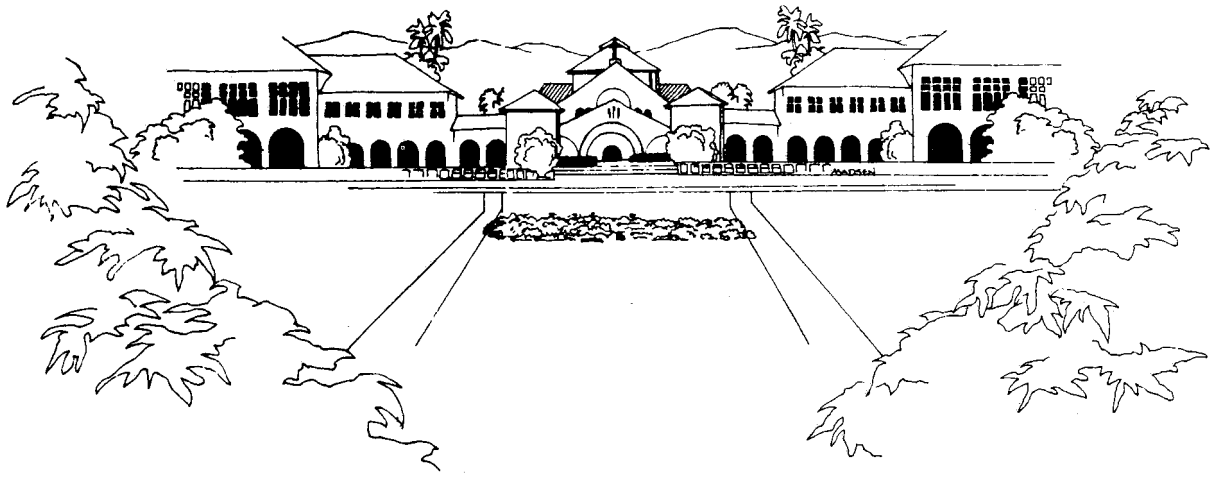
Lamport, Leslie. *LaTeX — User's Guide & Reference Manual*. Reading, Mass.: Addison-Wesley, 1986.

Schreiner, Axel T. "Lecture on Text Processing." Given at the University of Ulm, Dept. of Computer Science, 1987/88.

Schwarz, Norbert. *Einführung in TeX*. Bonn: Addison-Wesley, 1988.

Wichura, Michael J. *The PiCTeX Manual*. Providence, Rhode Island: TeX Users Group, 1986.

Wonneberger, Reinhard. *Kompaktführer LaTeX*. Bonn: Addison-Wesley, 1987.



AMS-TEX82 Users Course and TEX Users Group Meeting

Stanford University, July 11–15, 1983

Terman Engineering Center Auditorium