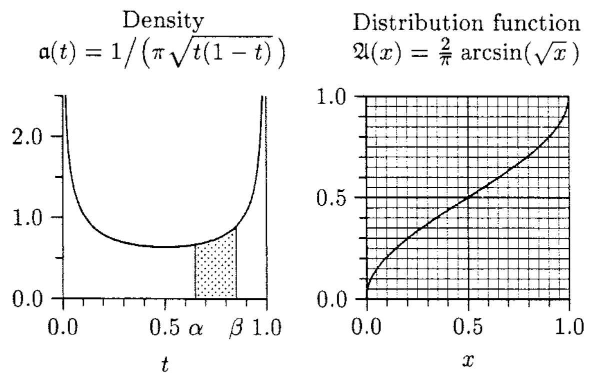
## P<sub></sub>ICTEX: Macros for Drawing PICtures

Michael J. Wichura
University of Chicago

### Overview

In the preface to *The TEXbook*, Knuth describes TEX as a "typesetting system intended for the creation of beautiful books — and especially for books that contain a lot of mathematics". PICTEX is a collection of TEX macros by means of which TEX users can easily instruct TEX to typeset beautiful pictures as a part of their books — and especially mathematical figures, such as the one below.

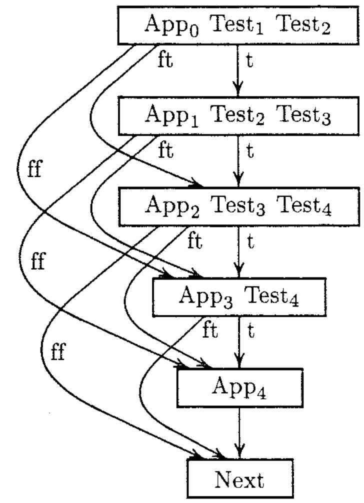
Figure 1. *Density $\mathfrak{a}$ and distribution function $\mathfrak{A}$ of the arc sine distribution. The shaded area under the graph of $\mathfrak{a}$ is $\mathfrak{A}(\beta) - \mathfrak{A}(\alpha)$.*




That figure and the others in this article illustrate the main things you can do with PICTEX: place text into a PICture; construct $x$ and $y$ axes with tick marks, tick labels, and axis labels; draw rectangles and other things made out of horizontal and vertical rules; draw straight lines and curves (without recourse to special fonts); use line fills that can be solid, dotted, dashed, or otherwise; and shade regions. In addition to these "primitive" graphics capabilities, PICTEX provides some "upper" level commands for drawing things like bar graphs, histograms, arrows, circles, and ellipses. Using TEX's powerful macro facilities, you can readily create other upper level commands that are tailored to your specific needs.

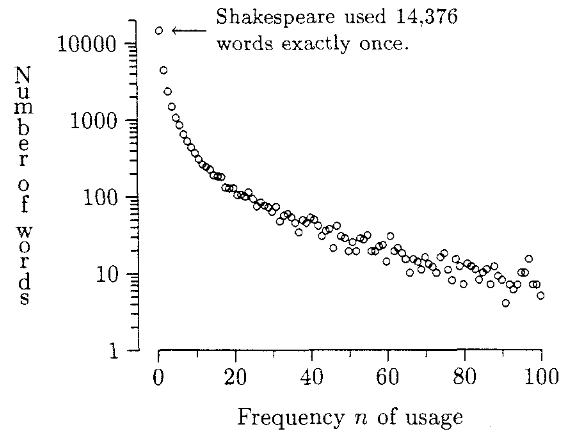
PICTEX has these advantages: (1) Figures become an integral part of the typesetting process. You can avoid having to leave the proper amount of space in your document for material that has to be created on some external device and later stripped into the finished product. (2) All of TEX's formatting capabilities are available for annotating your figures. In addition, that annotation will be done in the same fonts that you're using in the rest of your document. (3) Just as TEX is machine independent, so too is PICTEX. It doesn't matter whether you're working on a PC or mainframe computer. (4) Since typeset figures are embedded in the dvi file along with the rest of your document, all the advantages of TEX's device independent output accrue to them. In particular, you can revise away to your heart's content on your local system until your PICtures look just the way you want them to, and then you can have the final copy elegantly printed on a high resolution output device. (5) PICTEX can be extended using TEX's macro facilities, and can be used with LATEX.

Figure 2. *A segment of a complex flow chart.*




On the other hand, PICTEX has several limitations: (1) PICTEX was expressly designed to facilitate the construction of pictures such as Figure 1. It simply is not the right tool for producing illustrations such as the lions that grace the title pages of *The TEXbook*. (2) Within the realm of mathematical figures, PICTEX itself doesn't make 3D pictures or other complex things. Considering that TEX provides fewer arithmetic capabilities than the simplest pocket calculator, that would be asking for too much. However PICTEX can be used as an interface between TEX and a sophisticated graphics program. For example, Figure 2 was set using Fig and a Fig-to-PICTEX converter developed by Micah Beck. (3) PICTEX takes a while to draw a PICture. Figure 1 initially took 30 seconds on a Sun 3/60, the bulk of the time going into producing the two curves. In subsequent drafts PICTEX used a special routine to replot the curves and did the whole PICture in 10 seconds. (By contrast, running on a Sun 3/60, TEX processes a page of "straight

Figure 3. *Number of words Shakespeare used exactly n times, for n = 1 to 100 by 1.*




text" in about 1 second.) (4) PICtures take up a sizable amount of computer memory. A large PICture with several curves will exceed the capacity of a standard version of TeX. This difficulty can be circumvented by using a larger version of TeX, such as that developed by Bart Childs et al. (See page 122.)

The PICTeX manual contains detailed instructions on the use of the PICTeX macros, with many examples and exercises, along with a command summary and index. The manual is about 90 pages long. Both the macros and the manual may be obtained from TUG.

The next section of this article has some examples showing how one goes about drawing a PICture. The final section has an example showing how PICTeX can be extended via TeX's macro facilities.

## Drawing PICtures

Table 1 lists the code that was used to construct Figure 3, in which the common logarithm of the number of words Shakespeare used exactly $n$ times is graphed versus $n$ for $n = 1, 2, \ldots, 100$. In tabular form the data are as follows:

| $n$ | $\nu_n =$ *number of words used n times* | $\lambda_n = \log_{10}(\nu_n)$ |
|---|---|---|
| 1 | 14,376 | 4.1576 |
| 2 | 4,343 | 3.6378 |
| 3 | 2,292 | 3.3602 |
| $\vdots$ | $\vdots$ | $\vdots$ |
| 99 | 7 | 1.1761 |
| 100 | 5 | 0.6990 |

Figure 3 has $n$ on the horizontal ($x$) axis, and $\lambda_n$ on the vertical ($y$) axis.

Table 1. *PICTeX commands for Figure 3. (The line numbers are not part of the commands.)*

```
1 \beginpicture
2 \setcoordinatesystem units <.02in,.4in>
3 \multiput {$\scriptstyle\circ$} at
4    1 4.1576     2 3.6378     3 3.3602
5    4 3.1652     5 3.0183     6 2.9227
   (additional coordinates omitted here)
6   95 1.0000    96 1.0000    97 1.1761
7   98 0.8451    99 0.8451   100 0.6990 /
8 \put {$\longleftarrow
9   \vcenter{\hsize=100pt \raggedright
10  \eightpoint \noindent
11  Shakespeare used 14,376 words exactly
12  once.}$} [l] <4pt,0pt> at 1 4.1576
13 \setplotarea x from 0 to 100,
14   y from 0 to 4.301
15 \axis bottom
16  label {Frequency $n$ of usage}
17  ticks numbered from 0 to 100 by 20
18        short unlabeled quantity 11 /
19 \axis left shiftedto x=-5
20  label {\stack
21    {N,u,m,b,e,r,,,o,f,,,w,o,r,d,s}}
22  ticks logged
23    numbered at 1 10 100 1000 10000 /
24    unlabeled short from 2 to 9 by 1
25      from 20 to 90 by 10
26      from 200 to 900 by 100
27      from 2000 to 9000 by 1000
28      at 20000 / /
29 \endpicture
```

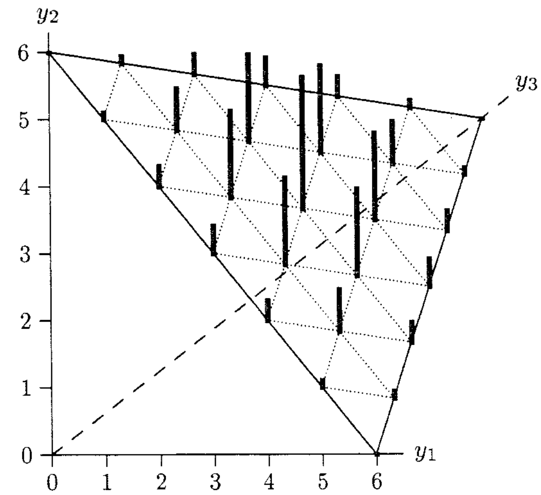
Table 1 is largely self-explanatory, since much of the code reads almost like English. However, some remarks are in order: (1) The command on line 2 tells PICTeX to set up a rectangular coordinate system in which one unit on the $x$ axis has length of .02 inches, and one unit on the $y$ axis has a length of .4 inches. The value .02 was chosen so that a run of 100 units on the $x$ axis would amount to $.02 \times 100 = 2$ inches, leaving plenty of room in *TUGboat*'s 3 inch column for the $y$ axis structure. (2) The command on lines 3–7 places small circles at the coordinate points $(1, 4.1576)$, ..., $(100, 0.6990)$. The trailing '/' on line 7 informs PICTeX that the list of coordinate points is exhausted. (3) The option '`[l] <4pt,0pt>`' on line 12 to the `\put` command on line 8 instructs PICTeX to place the associated text (i.e., the arrow and the `\vcentered` paragraph) into the PICture with its left edge 4 points to the right of the the designated coordinate $(1, 4.1576)$.

(4) PiCTeX draws coordinate axes along the edges of a prespecified rectangular "plot area". The command on lines 13 – 14 sets up such a plot area, running from 0 to 100 along the $x$ axis and from 0 to $4.301 = \log_{10}(20{,}000)$ along the $y$ axis. The command on line 15 draws an axis along the bottom edge of this plot area. The analogous command on line 19 creates an axis along the left edge of the plot area; however this axis is subsequently shifted to lie along the line $x = -5$.

Some features of PiCTeX that emerged in the preceding example merit further discussion. The coordinate system is a key element of any PiCture, since it governs the placement of everything that's put into that PiCture. You can set and reset the coordinate system at will, moving the location of the origin and changing the lengths of the $x$ and $y$ units; this facilitates work on PiCtures like Figure 1 having several components. PiCTeX's `\put` command and its relatives have options that make it easy to specify exactly how an object should be positioned relative to a given coordinate. You can choose between various horizontal (centered, left, right) and vertical (centered, above, below, baseline) orientations, and you can offset objects horizontally and vertically by amounts that don't depend on the current coordinate system. `\axis` is PiCTeX's most versatile command. You can: choose between bottom, left, top, and right axes; freely specify where ticks are to be placed; use any combination of unlabeled, numbered, or user-labeled ticks; have the ticks marks point out from the plot area or into it, or even extend all the way across it; specify axis labels and a plot heading; govern the length and thickness of axes and tick marks; and adjust the spacing between the various components of the graph framework. In general you can fine tune any part of a PiCture; a few minor adjustments may make the difference between a figure that is merely presentable and one that is a work of art. One last point: you don't have to prespecify to TeX how much room a PiCture will take up. PiCTeX automatically determines the size of a PiCture and passes this information on to TeX so that the PiCture can be positioned appropriately in the page layout.

What about lines and curves? PiCTeX has four interpolation modes, two of which are piecewise linear interpolation and piecewise quadratic interpolation. (The other two modes generate histograms and bar graphs.) Moreover, PiCTeX has four modes for line fill: solid, dotted, dashed, and user-specified. Given a list of coordinate points, PiCTeX's `\plot` command connects those points

Figure 4. *Trinomial sample space and probabilities for $m = 6$ and $\pi = (1/3, 1/3, 1/3)$. For nonnegative integers $y_1$, $y_2$, and $y_3$ summing to $m$, the probability at the point $(y_1, y_2, y_3)$ is $\frac{m!}{y_1!y_2!y_3!}\pi_1^{y_1}\pi_2^{y_2}\pi_3^{y_3}$. (The $y_3$ axis recedes into the plane of the page.)*




using the current interpolation and line fill modes. Every option in PiCTeX has a default; the defaults for `\plot`ting are piecewise linear interpolation with solid line fill.

For example, in Figure 4 the edges of the simplex were drawn simply with

```
\plot   0 6   6 0   8 5   0 6   /
```

while the $y_3$ axis was created with

```
\setdashes
\plot   0 0   8.5 5.3125   /
```

In Figure 1 the left-half of the arc sine density $\mathfrak{a}$ was created by first placing the origin of the coordinate system at the point $t = 0.5$ on the horizontal axis, and by then entering

```
\setquadratic
\inboundscheckon
\plot     -.485 2.6187    -.475 2.0388
          -.465 1.7320    -.465 1.7320
          -.44  1.3403    -.40  1.0610
          -.36  0.9174    -.32  0.8285
          -.27  0.7564    -.22  0.7089
          -.12  0.6558      0   0.6366 /
```

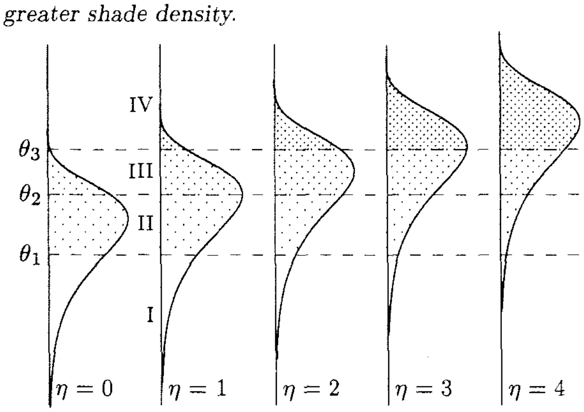
Note that $\mathfrak{a}(.5{-}.485) = 2.6187$, $\mathfrak{a}(.5{-}.475) = 2.0388$, and so on. The "in bounds check" feature kept PiCTeX from plotting any points with vertical coordinates greater than 2.5. To plot the right half of $\mathfrak{a}$ the same commands were used, but with the signs of the horizontal coordinates changed to '+'.

In general, to `\plot` any given curve you have to provide PiCTeX with a list of coordinate values, as opposed to, say, some mathematical formula.

However, you can create such a list using Fortran or C or whatever, store it in a file, and have `\plot` take its input from that file.

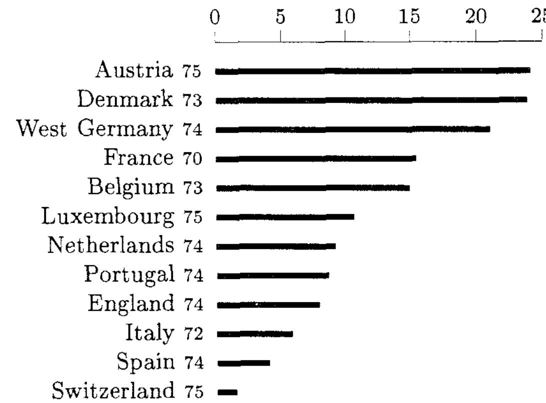
PiCTeX draws lines and curves by placing lots of periods close together. This takes a lot of time, because PiCTeX repeatedly has to lead TeX step by simple arithmetic step through complex calculations, like finding the distance between two points in a coordinate plane, for which it has no primitives. (TeX can only do fixed point addition, subtraction, and multiplication, and division by integers.) And all those periods take up a lot of room in TeX's memory. That's the down side; the up side is that there are no restrictions on the slope of lines and the curvature of arcs. By contrast, LaTeX's `picture` environment draws lines and circles by piecing together characters from specially designed fonts. This is fast and uses little storage space, but the choice of slopes and radii is quite limited. Fortunately, this is one instance where you can have your cake and eat it too. PiCTeX knows how to use LaTeX's picture objects, and PiCTeX's macros can be used inside LaTeX. (Don't try this without first reading the relevant section from the PiCTeX manual.)

Figure 5. *Diagram showing how the probabilities for the response categories* I, II, III, *and* IV *in a certain proportional hazards model vary with $\eta$. Probabilities are shown as shaded areas; higher numbered categories have greater shade density.*




PiCTeX shades a region by placing a "shading symbol" at every point of a "shading lattice" that falls within the region. You have control over both the symbol and lattice. In Figure 5 three different shading lattices were used, each with a `\fiverm` period as the shading symbol.

Figure 6. *Suicide rates in western Europe per 100,000 population per year for the years (19xx) indicated.*




## Extending PiCTeX

Consider now how you might go about drawing Figure 6. To get started you could enter

```
\beginpicture
\setcoordinatesystem units <5pt,11pt>
\setplotarea x from 0 to 25,
  y from 0 to 0
\linethickness=.15pt
{\eightpoint \axis top
  ticks numbered from 0 to 25 by 5 / }%
\linethickness=2pt
```

These commands establish the coordinate system, draw the axis, and set PiCTeX's line thickness parameter to 2 points for drawing the black bars. The problem that needs addressing is how to place the bars and their associated labels into the PiCture. You could do this easily with PiCTeX's bar graph command, or even with TeX's `\halign`. However the point of this section is to illustrate a technique that can often be put to good use when there isn't an upper level command that does what you want to do. Let's agree then that you have to solve this exercise using just PiCTeX's primitive commands `\put` and `\putrule`. The most direct solution would be

```
\putrule from  0 -1  to  24.1 -1
\put {Austria \sevenrm 75}
  [Br] <-5pt,-2pt> at  0 -1
\putrule from  0 -2  to  23.8 -2
\put {Denmark \sevenrm 73}
  [Br] <-5pt,-2pt> at  0 -2
```

and a bunch of similar commands. (The 'Br's above stand for 'baseline right' orientation.)

It would be much less tedious to enter just the bare essentials, say in the form

```
\placebars
  Austria      5 24.1
```