Overview
The \LaTeX{} Document Preparation System is a collection of \TeX{} macros designed to make it easy to produce high-quality documents. \LaTeX{} was developed by Leslie Lamport — also the author of \LaTeX{}: A Document Preparation System which is published by Addison-Wesley Publishing Company. \LaTeX{} is in the public domain and is included in the standard \TeX{} distribution available from Stanford University. It is also included in most proprietary implementations of \TeX{}.

\LaTeX{} is based on the concept of a document as a set of structures (e.g. descriptions, tables, enumerated lists) called \emph{environments}. Environments start with a \verb|\begin{environment}| statement such as \verb|\begin{tabular}| or \verb|\begin{verbatim}| and end with an analogous \verb|\end{environment}| statement.

This article describes an upward compatible set of extensions to \LaTeX{}'s \emph{picture} environment implemented by Textset to take advantage of the graphics capabilities of PostScript language used by the Apple LaserWriter and other printers. Textset's \TeX{} output driver for the PostScript printers, DVILASERIPS, includes support which makes it easy to integrate user-supplied PostScript statements with the PostScript statements being generated automatically by DVILASERIPS as it converts \TeX{} DVI files into PostScript format. The \LaTeX{} extensions are included free of charge with all DVILASERIPS distributions.

The \LaTeX{} Picture Environment
One of \LaTeX{}'s nicest features is the \emph{picture} environment. With it, one can "draw" pictures. The picture in Figure 1 was created by the \LaTeX{} commands that are to the left of it.

The "(1,1)(-150,27)" specifies how much space the picture should take up and also allows the user to move the picture around on the page. In the example, "(1,1)" tells \LaTeX{} to allocate hardly any space at all for the picture—just an imaginary box 1pt by 1pt, though obviously the actual picture is much larger. The "(-150,27)" positions the picture in the desired place by moving the whole picture to the right 150pt and down 27pt from where \LaTeX{} would have otherwise have put it, at the next place a regular character would go that has dimensions 1pt by 1pt.

Within the \LaTeX{} \emph{picture} environment, pictures normally are "drawn" by positioning special characters, in this case curved and straight line segments, in ways that give the appearance of circles and continuous lines. Circles are actually composed of four discrete curved segments, one for each quarter of the circle. \LaTeX{} carefully aligns them so that
they print as a complete circle. Oblique lines, ones that are neither vertical nor horizontal, are composed of a number of shorter segments which are placed end on end and form what looks like a continuous line. These special characters are obtained from a set of special fonts (circle10, line10, lazy10, etc.) that are distributed with \LaTeX. Using these special fonts, \LaTeX “fools” \TeX into drawing pictures even though \TeX doesn’t really have any graphics capabilities.

\LaTeX’s picture environment is somewhat restrictive since “pictures” must be put together from the limited number of “pieces” available in the special \LaTeX fonts. \LaTeX carefully figures out the position of each circle or line segment. Longer lines, since they are made up of many shorter segments, cause longer execution and printing times.

Even worse, creators of complex pictures often get the error “\TeX capacity exceeded—sorry.” This happens because \TeX runs out of memory since there are too many “characters” on the page to handle. In addition, the user must be sure to use lines having only certain slopes, and be willing to accept circles of only certain diameters since in the standard \LaTeX distribution there are only 36 available slopes for straight lines and 10 different diameters for circles.

Also, \LaTeX can only print hollow circles of diameters that are multiples of 4pt in size. If a user selects an in-between size like 21pt, \LaTeX will pick the closest available size: 20pt. The biggest hollow circle it can do is 40pt—a little over half an inch—in diameter. The biggest solid circle is 15pt in diameter. The steepest non-vertical slope available is $\pm \frac{3}{7}$. \LaTeX checks the slope arguments to make sure neither $\Delta x$ nor $\Delta y$ exceeds $\pm 6$, and that they are both integers. For vectors, because the arrowhead is drawn from a font too and slope availability in its case is even more limited, neither $\Delta x$ nor $\Delta y$ can exceed $\pm 4$.

Figure 2 demonstrates these problems by drawing the same picture with different slopes and diameters. No picture was produced when this was run under standard \LaTeX because first the “$-6$” for the left arm exceeded $6$, caused a \LaTeX error message, and halted execution; then the “$-7$” for the right leg exceeded $6$ and \LaTeX stopped again.

When these two simple problems were “corrected” by substituting “(-1, -4)” for the left arm and “(1, -6)” for the right leg, the picture still did not look as desired—the head was 40pt in diameter instead of 70pt. This messed up the body as well, causing the head to “hover” above the body. In
The left picture was produced by standard \LaTeX, the right picture by extended \LaTeX.

Figure 3, the left picture is how standard \LaTeX did it. On the right is the Extended \LaTeX version of the original example.

The \LaTeX Picture Environment Extensions
The PostScript printer language is very powerful. It has a very general and flexible set of capabilities enabling the creation of practically any textual or graphic image. It can draw letters, lines, and circles of virtually any size and slope. It can shade the inside of any closed figure it draws. And it allows lines of text to be drawn at any angle. The extended version of \LaTeX uses these PostScript capabilities to provide a number of features not normally possible with \LaTeX. All extensions are modelled closely after the way \LaTeX normally does things.

Extensions to Existing \LaTeX Commands
Line Thickness Standard \LaTeX provides two line thicknesses which are selected by typing either \texttt{\thicklines} or \texttt{\thicklines} inside the picture environment. These two declarations simply tell \TeX which font to use, either \texttt{\line} or \texttt{\line}, respectively. \LaTeX version 2.05 also provided a \texttt{\linethickness{}} command with which the user could vary the thickness of non-oblique (vertical or horizontal) lines.

The latest release of \LaTeX does not document the use of the \texttt{\linethickness{}} command, probably because of the confusion which arose because it could only do non-oblique lines. With the \LaTeX extensions, though, it has been brought back. By typing something such as \texttt{\linethickness{5pt}} you can set the line thickness of a PostScript-drawn line or circle to 5pt.

\texttt{\thlinelines} is equivalent to the command \texttt{\linethickness{0.4pt}}. \texttt{\thicklines} is equivalent to typing \texttt{\linethickness{2pt}}. The Extended \LaTeX version of \texttt{\thicklines} draws a much thicker line than regular \LaTeX \texttt{\thicklines} does. Lines of intermediate thickness may be drawn with the \texttt{\linethickness{}} command.

Circles Circles of practically any diameter may be drawn. For hollow circles, use \texttt{\circle{diam}}. For solid circles, use \texttt{\circle*[diam]}. 
**Lines and Vectors**  Length and accuracy are virtually unlimited. Line lengths and \texttt{put} coordinates need not be restricted to integer values as is the case normally with \LaTeX. However, whole numbers are still required for $\Delta x$ and $\Delta y$. This might seem restrictive, but if you want a line 34.5 points long with a slope of, say, 0.66, try \texttt{line (100,66) (34.5)}. A negative length causes the line to project opposite to the specified direction.

The \texttt{vector} command uses the same syntax as the \texttt{line} command and is just as versatile. The arrowhead points away from the location of the \texttt{put} command. If you choose to use a thick line, you’ll find the arrowhead may not look very good since it was designed to be used with thinner lines. In this case, use the \texttt{PSarrowhead} command (described later) to custom-make your own vector with a bigger arrowhead.

**New \LaTeX** Commands**

**PSoval** If you’ve used \LaTeX’s \texttt{picture} environment, you have seen that the \texttt{oval} command doesn’t really produce an oval at all; it’s actually a rectangle with rounded corners. The corners are made up of the same fonts that make circles. Because \texttt{oval} produces this unique-looking figure, the original \LaTeX macro was not redefined. Instead, a new command, \texttt{PSoval}, was created. The syntax of \texttt{PSoval} is as follows:

\begin{verbatim}
PSoval\{width\}\{height\}
\end{verbatim}

The \texttt{put} command specifies the location of the center of the ellipse. Again, width and height are virtually unlimited.

You can get a solid oval by typing \texttt{PSoval*} with the same syntax as \texttt{PSoval}.

Two drawbacks — you can’t use \texttt{PSoval} to put text inside the oval. Use an extra \texttt{put} command to do that if you need to. Also, this release of \texttt{PSoval} has no provision for printing a portion of an oval — something \LaTeX can do with \texttt{oval}.

**PSarrowhead** Extended \LaTeX allows the user to control the shape of arrowheads. The user can make almost any size or shape of arrowhead desired. Add a line and get a custom-made vector as well. The syntax of the command is:

\begin{verbatim}
PSarrowhead\{\Delta x, \Delta y\}\{length\}\{width\}\{depth\}
\end{verbatim}

where \textit{length}, \textit{width} and \textit{depth} are defined in the summary. Keep $\Delta x$ and $\Delta y$ as whole numbers as in the \texttt{line} and \texttt{vector} commands. You can use almost any values you want for the length, width and depth. Use \texttt{PSarrowhead*} with parameters as above to get a solid arrowhead.

The Extended \LaTeX \texttt{vector} command uses a solid arrowhead that has a length of 8pt, a width of 4pt, and a depth of 2pt.

**PSfill** Extended \LaTeX, because it can use PostScript capabilities, allows the user to temporarily “rotate” the coordinate system of the page for individual lines. This means that text no longer has to be horizontal. In fact, it can be at any slope desired. The syntax of \texttt{PSfill} is

\begin{verbatim}
PSfill\{\Delta x, \Delta y\}\{object or text\}
\end{verbatim}

The user can put text or a picture object like a \texttt{framebox} inside the third (brace enclosed) argument to \texttt{PSfill}. This addition is very useful for labelling lines and rotating \LaTeX picture objects such as \texttt{oval’s} and \texttt{framebox’es}.

**PSpath** Extended \LaTeX allows a much easier way of drawing straight lines — by naming the coordinates of the endpoints. In fact, a series of connected lines can be drawn with just one command. The syntax of \texttt{PSpath} is

\begin{verbatim}
PSpath\{(x_0, y_0)\{(x_1, y_1)\{(x_2, y_2)\ldots\{(x_n, y_n)\}
\end{verbatim}

You don’t even need the \texttt{put} command, since the starting point is defined to be $(x_0, y_0)$ and the $n$ points are then connected in sequence. The number of points that can be connected using just one \texttt{PSpath} varies from system to system. If you try to use a lot of points and get “capacity exceeded” errors, use more \texttt{PSpath} commands.

The map example of Figure 6 demonstrates a series of points connected as one \texttt{PSpath}.

**Conclusion**

The \LaTeX extensions described in this article represent one way that \TeX and PostScript can be used in a combination that is more powerful than either one alone. While the implementation described here is available only with Textset’s DVILASERIPS program, other individuals or organizations might make use of a similar strategy to design their own individualized \LaTeX extensions.

**Note added in press**

Leslie Lamport read a preprint of this article and had several much appreciated comments. He correctly guessed that we were not, at the time the article was written, aware of the new option that draws quadratic Bezier splines. He suggested for portability’s sake that the extensions be enabled with a document-style option; the extensions in fact already must be enabled by a command and will work within any document style. We incorrectly stated that the \texttt{linethickness} command was
Control Macros
\PSextensionsOn Enables LaTeX extensions.
\PSextensionsOff Disables LaTeX extensions and reverts to regular LaTeX processing.

Redefined Macros
\linethickness{dimen} Sets the thickness of all lines, including sloped lines, circles, and other geometric shapes. \textit{dimen} is something such as \texttt{0.01in} or \texttt{3pt}.
\thinlines Sets line thickness to \texttt{0.4pt}.
\thicklines Sets line thickness to \texttt{2pt}.
\circle{\textit{diam}} Puts a hollow circle at the current location with diameter \textit{diam}, which can be to almost any size and accuracy desired.
\circle*{\textit{diam}} Just like \circle, only the circle is solid.
\line(\Delta x,\Delta y){length} Puts a line starting at the current point with a slope \textit{\Delta y}/\textit{\Delta x} and a length defined as described in Lamport's book, where \textit{length} is how far over the line is to go in terms of \textit{x}-units, unless the line is vertical, where \textit{length} then means the length of the line up or down. Use whole numbers for \textit{\Delta x} and \textit{\Delta y}. \textit{length} can be almost any number to virtually any accuracy.
\vector(\Delta x,\Delta y){length} Just like \line above, except in addition, a solid arrowhead is added at the other end of the line, pointing away from the \put point.

New Macros
\PSarrowhead(\Delta x,\Delta y){\textit{length}}\{\textit{width}\}\{\textit{depth}\} Puts a hollow arrowhead with its tip at the position of the \put command, pointing at a slope \textit{\Delta y}/\textit{\Delta x} and where the arguments are as shown:

\begin{center}
\psfig{file=arrowhead.png}
\end{center}

When \textit{depth} is positive, the figure produced is a concave arrowhead as shown. But when \textit{depth} is negative, the figure is convex, and other shapes can then be created, like diamonds and rotated squares.
\PSarrowhead*{same as above} Puts a solid arrowhead at the position specified by the \put command.
\PSoval{\textit{width}}\{\textit{height}\} Puts a hollow ellipse with its center at the position specified by \put.
\PSoval*{\textit{width}}\{\textit{height}\} Puts a solid ellipse with its center at the position specified by \put.
\PStilt(\Delta x,\Delta y)\{\textit{object}\} Puts \textit{object} at the position specified by \put, but at a slope \textit{\Delta y}/\textit{\Delta x}. \textit{Object} can be text, or any \textit{picture} environment object such as \oval, \PSarrowhead, or \framebox with their appropriate arguments included.
\PSpath(x_0,y_0)\{(x_1,y_1)\ldots(x_n,y_n)\} Connects the points \{(x_1,y_1)\} with straight lines starting at the point \((x_0,y_0)\) and ending with \((x_n,y_n)\). Can have an arbitrary number of points, but the exact number varies from system to system.

Figure 4.
Summary of Syntax of LaTeX Extensions.

no longer documented; see page 199 of the \texttt{LaTeX} manual. He pointed out that our "oval" shape is actually an ellipse, while his, though not an oval, is a convenient shape in which to insert text. Textset plans to implement several of his suggestions into the program.
This extended \LaTeX\ code and the DVILASER/PS system produced the map in Figure 6 (facing page).
Figure 6.