Abstract

The old picture environment, introduced by Leslie Lamport into the \LaTeX kernel almost 20 years ago, appears to be neglected in favor of more modern and powerful \LaTeX packages that eliminate all drawbacks of the original environment. Nevertheless it is still being used behind the scenes by a number of other packages. Lamport announced an extension in 1994 that should have removed all the limitations of the original environment; in 2003 it was released the first stable version; in 2009 it was actually expanded to new functionalities. Nowadays the picture environment can perform as most simple drawing programs do, but it has special features that make it invaluable.

Sommario

L’ambiente picture, introdotto da Leslie Lamport nel nucleo di \TeX quasi 20 anni fa, sembra essere trascurato a favore di pacchetti \LaTeX più moderni e più potenti che eliminano tutti gli inconvenienti della versione originale. Tuttavia esso è ancora usato oggi giorno dietro le quinte da numerosi altri pacchetti. Nel 1994 Lamport annunciò un pacchetto di estensione che avrebbe dovuto rimuovere ogni inconveniente; solo nel 2003 apparve una prima versione; nel 2004 questa estensione divenne stabile; e nel 2009 essa è stata arricchita di nuove funzionalità. Oggi l’ambiente picture si comporta come qualsiasi altro semplice programma di disegno, ma presenta alcune particolarità uniche che lo rendono insostituibile.

1 Introduction

The plain \TeX, as described in the \TeXbook (Knuth, 1990), contained a simple way to draw simple graphics with \texttt{tex}; when \LaTeX was first published in 1984, it contained an environment suitable for relatively complex graphics; Lamport’s handbook, \cite{Lamport1994} described its workings and commands. But all this seems to be fallen in complete oblivion.

Most users of \LaTeX related forums keep asking questions such as “How can I produce such and such symbol”; I keep answering “Use the picture environment”. Apparently nobody follows my suggestions, which of course they are free not to; but they would save time if they spent no more than 15 minutes in reading the environment description on Lamport’s handbook; the second edition of this handbook announces the extensions and discusses the eliminated drawbacks; these extensions were eventually realized only in 2003 by Gässlein and Niepraschk, \cite{Gasslein2009}. In 2004 the same authors released a stable version. In 2009, with the contribution of the third author Tkadlec, they released an enhanced versions that added quite a few new commands that substantially extend the picture environment functionality. But, except the latest additions of the year 2009, everything else was already documented by Lamport in his handbook second edition.

Not longer than a few days ago a forum participant asked how he could draw a square wave and a saw tooth wave of suitable size for setting them in line with his text: $\int$ and $\wedge$.

The whole thing required to produce two simple commands by resorting to the recent \texttt{pict2e} extension package new commands:

\begin{verbatim}
\newcommand*{\sqwave}[1]{{0.125ex}\unitlength=#1\relax\beginpicture(20,10)
\polyline(0,0)(6.5,0)(6.5,15)(13,15)(13,0)(19.5,0)\endpicture}
\newcommand*{\sawtooth}[1]{{0.125ex}\unitlength=#1\relax\beginpicture(20,10)
\polyline(0,0)(6.5,0)(6.5,15)(13,0)(19.5,0)\endpicture}
\end{verbatim}

Most of the time suggestions are given by and to the forum participants to use external drawing programs; or to use the sophisticated PStricks or \textit{TikZ} packages. The former solution is generally to be avoided, because if some lettering is needed, it requires more work to use the same fonts as those used in the document. The latter two packages are really capable of doing marvellous and complicated drawings, but require a long documentation time and a steep learning curve.

But there is a unique feature to the picture environment: it can produce drawings of zero width and/or height. This special feature makes them precious for packages such as \texttt{eso-pic}, \cite{Niepraschk2010}, \texttt{crop}, \cite{Franz2003}, \texttt{layout}, \cite{McPherson2000}, \texttt{layouts}, \cite{Wilson2009}, and others. These packages draw things on the page that do not require any external package, and therefore don’t have any dependency; these drawings occupy no space, although they have a specific
position on the page; their contents reach whatever point on the page, as background images or marks that do not interfere with the positioning of other page elements.

In particular eso-pic (and similar packages for setting watermarks) and crop exploit this functionality exactly for setting a background picture or the crop marks in the correct positions without interfering with the other elements of the page.

At the same time within the picture environment it is possible to use the cubic Bézier curves and to draw polygons, arcs and sectors, oval frames whose corner curvature can be freely specified, white or filled circles of any dimension. The \polyline macro used in the above simple examples makes it very easy to draw polylines with any number of corners and any segment slope, to the point that if the nodes are sufficiently close, it is possible to draw “smooth” curves whose point may be calculated with any number-crunching personal or main frame computer.

2 In detail

Everybody can agree that the original picture environment, created by Lamport with its very first version of \LaTeX, was pretty rudimentary but there was practically nothing else to use in its place. The straight lines could have slopes that were ratios of relatively prime integer numbers not exceeding 6 in absolute value. For vectors the limitation was even stricter; the slopes could not be specified with integer numbers larger than 4 in absolute value. Why there were such strange limitations? because straight lines were made up through the juxtaposition of small segments 10pt (∼3mm) long taken from a special font; this same special font contained also the vector arrow tips that occupied a large part of the available positions; and, remember, at that time the typesetting engine could deal only with 128-glyph fonts, so that the available short segments and arrow tips, plus a selection of closed and filled circles and/or quarter circles required strict limitations on the drawing performances.

Patient programmers made up extension packages such as curves \cite{Maclaine-Cross2008}, that could overcome such limitations by drawing lines of any slope and circles of any diameter by juxtaposing an “infinity” of small dots. For plain \LaTeX there existed another package, PicTeX \cite{Wichura1987}, that with a suitable interface could work also with \LaTeX. That package performed very well on large mainframes with large memory capabilities, but worked very poorly on the desktops of that age, the eighties, when a 20MiB hard disk was a luxury and a 640KiB RAM was almost the maximum available. These packages mostly saturated the RAM and drawing was virtually impossible on personal computers.

Some progress was made when the PostScript format became available; some drawing packages (again curves) exploited the \special command in order to write on the output file raw PostScript drawing commands, so that the actual action of drawing was demanded on the screen or printer driver. Nevertheless powerful extensions in this directions, such as PSTricks appeared much later. Drawing with external programs and importing the resulting artwork was therefore a necessity, but not an easy task.

Things evolved in the right direction when personal computers, having become the universal complement of any person needing to write anything, started having a more user friendly interface, larger RAMs, larger hard disks, and better programs, the \TeX system included. The nineties, besides the important passage from \TeX2.x to \TeX3.0, gave us L\TeX\2e, PostScript fonts, and the drawing instrument METAPOST. This program used more or less the same philosophy that led Knuth to develop METAFONT, in order to draw the \TeX system default fonts; METAPOST produced a simplified output PostScript code that was understood also by the new born typesetting program pdftex. METAPOST was, and still is, fully compatible with the rest of the \TeX system typesetting engines, so that all the \TeX and \LaTeX features could be used for putting any lettering on the METAPOST output files.

Meanwhile \LaTeX went on with its small drawing environment, without exploiting the new possibilities with the PostScript format and the PDF portable document format, until Gäßlein and Niepraschk wrote the picture extension announced by Lamport some ten years before.

Let see what are the enhancements introduced by the extension realized by Gäßlein and Niepraschk;

1. First of all the enhancements rely on the drivers that are being used to display or to print the document. More precisely when the latex program is used, the \special commands to the driver contain only PostScript language commands; this implies a transformation of the resulting DVI file into PostScript one by means of \dvips, and possibly a second transformation to the PDF format by means of ps2pdf. On the opposite, if the document is processed with pdflatex, then the \special commands contain only the PDF language commands. Therefore the extension is fully compatible with the typical output formats provided by the most popular typesetting engines; but this performance is fully automatic and users need not bother about these details.

2. The output file size very often is smaller since the actual drawing computations are performed by the suitable drivers.

3. Here we describe the 2009 extension, therefore
there are also some commands that were not described by Lamport.

4. One of the limitations of the original environment was the slope of lines and vectors; in the first implementation of the extension the slope coefficients had to be integers non higher than 1000, in absolute value, therefore implementing Lamport’s description of 1994; but the 2009 enhancements removes even this limitation, and the slope coefficients can be any fractional decimal number (well, yes, not too large, not higher than 16383.99998 which corresponds to the largest dimension in points that any \TeX{} system typesetting engine can handle). Now line and vector slopes should not have any sensible limitation.

5. The above is valid also for vectors; even better, now it’s possible to pass an option to the package so that it can draw the arrow tips in “\TeX{} style” or in “PostScript style”. In \TeX{} style the joining sides to the arrow tip are slightly concave, and the arrow base is straight; in PostScript style they form a polygon that resembles a stealth aircraft.

6. Circles and quarter circles were available in a limited set; now they can be drawn in any size both filled and unfilled.

7. Line thicknesses could be specified only as \texttt{\thinline} (default) and \texttt{\thickline} (twice as thick compared to \texttt{\thinline}); only vertical and horizontal lines could receive the thickness specified by means of the command \texttt{\linethickness\langle dimension\rangle}. Now \texttt{\linethickness} can modify the thickness of all sorts of lines, Bézier splines included.

8. “Ovals”, frames with rounded corners, could have the corner quarter circle with an automatic setting of its radius, in any case not larger that about 15pt – about 5mm –, and they could not receive a radius specification that might be chosen by the composer; of course this radius should not exceed the half length of the shorter frame sides (that is, half of the distance of the longer straight lines that form the longer sides of the frame) but the radius can now be specified as an optional argument to the \oval{} command so that the created frame can have a very different look when a smaller radius is chosen compared to the same sized frame with a larger corner radius.

9. Quadratic and cubic Bézier splines are now generated with the driver commands and they result smooth curves, not lines with a rough contour, due to the juxtaposition of many small dots. The possibility of specifying the number of dots is available even now, but it is mostly for backwards compatibility, although, even now, dotted splines might be used for special purposes. In any case they do not suffer any magnification when seen on the screen; they are scalable vector strokes. The previous command \texttt{\bezier} is maintained with its compulsory specification of the number of points to use, but two new commands with an optional specification of the number of points, are introduced, \texttt{\bezier*} for tracing quadratic Bézier splines, and \texttt{\bezier} for tracing cubic Bézier splines; this last command was not described in Lamport, and is a completely new command to the package.

10. Up to this point the traditional commands are discussed and the differences with the past original environment are shown. The last extension of the \texttt{pic2e}, published in the second half of 2009, adds some other commands that draw other lines but in general don’t require the use of \put to place these lines in a special position. Of course they may be shifted with \put, but this might come handy when fine tuning the picture, without being really necessary to use \put.

11. A first exception to the above statement is the new macro \texttt{\arc} that is a generalization of \texttt{\circle} (with or without asterisk; in both cases the command with asterisk produces a filled contour) and requires to put the arc center in a specific position, so that the whole command must be set as an argument to \put; the \texttt{\circle} command is used as:

\begin{verbatim}
\put(⟨x⟩,⟨y⟩){\circle(∗){⟨diameter⟩}}
\end{verbatim}

and similarly the \texttt{\arc} command is used as:

\begin{verbatim}
\put(⟨x⟩,⟨y⟩){\arc(∗){⟨ang1⟩,⟨ang2⟩}{⟨radius⟩}}
\end{verbatim}

The arc has its center at point ⟨⟨x⟩,⟨y⟩⟩, and it will go from the angle ⟨⟨ang1⟩⟩ to the angle ⟨⟨ang2⟩⟩; angles are in sexagesimal degrees and are positive in the anticlockwise direction; if the optional angles are not specified, the full circle is drawn. The arc is actually drawn from the smaller angle to the larger one, so that the order in which ⟨⟨ang1⟩⟩ and ⟨⟨ang2⟩⟩ are given is not important.

12. The following lines do not require the \put command:

\begin{verbatim}
\Line(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)
\polyline(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)...(⟨xN⟩,⟨yN⟩)
\polygon(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)...(⟨xN⟩,⟨yN⟩)
\polygon∗(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)...(⟨xN⟩,⟨yN⟩)
\end{verbatim}
There are dozens of examples in the \texttt{guIt} command is erroneously spelled thor name and the program used for producing it.

art picture has a small legend containing the au-

mentation, ({\textsc{3 Examples}} Claudio Beccari 2011), where every line art picture has a small legend containing the author name and the program used for producing it.

In order to draw the various lines and curves, the internal commands make use of the “turtle graphics” commands used within both the PostScript and PDF languages. Such elementary commands are available to the user also through package \texttt{pict2e}; they are:

\begin{itemize}
  \item \texttt{\moveto((x),(y))}
  \item \texttt{\lineto((x),(y))}
  \item \texttt{\curveto((x2),(y2))((x3),(y3))((x4),(y4))}
\end{itemize}

and a few more that the reader may find in the documentation, (Gäsßlein et al., 2009).

These commands may be used in any order, except \texttt{\moveto} that must fix the first positioning of the drawing pen; but in order to finish the path it is optional to use \texttt{\closepath} in order to draw a line from the last point to the initial one, and then it is necessary to use \texttt{\strokepath} in order to actually draw the path or \texttt{\fillpath} in order to fill the path with the default color.

The initial and final points of an open path may be controlled with the commands \texttt{\buttcap} (just cut the path at the end points), or \texttt{\roundcap} (adjust the end points with a filled semicircle), or \texttt{\squarecap} (adjust the end points with a filled half square); in general with line art the \texttt{\roundcap} should be preferable, but sometimes it’s better to use one of the other two kinds of end point finishing, figure 1.

Similarly the joins between adjacent segments of a polyline or a polygon may be adjusted with the three commands \texttt{\miterjoin}, \texttt{\roundjoin}, and \texttt{\beveljoin}, as shown in figure 2.

Since this book, as a collective effort of the Italian Group of \TeX\ Users, is completely downloadable from the \texttt{guIt} site \url{http://www.guit.sssup.it/downloads/GuidaGuIT.pdf} including the source files, where the interested reader can find plenty of ideas and useful “tricks”: well, yes, any software user develops his/her own shortcuts and tricks in order to simplify recurrent operations; he or she will write himself or herself suitable macros which are the ordinary tricks any \TeX\ user eventually makes use of. But this would be done also with any other more sophisticated drawing package, such as \textsc{TikZ} (Tantau, 2010), or \textsc{PSTricks} (van Zandt, 2003).

Here we present few examples, possibly with their source code, in order to see the modern \texttt{picture} environment at work.

\textbf{A heptagon}

We compute the vertices of a heptagon inscribed into a circle with a diameter of 6cm by means of a pocket calculator:

\begin{verbatim}
  v_1 = (1.3017, -2.7029)
  v_2 = (2.9248, -0.6676)
  v_3 = (2.3455,1.8705)
  v_4 = (0,3)
  v_5 = (-2.3455,1.8705)
  v_6 = (-2.9248, -0.6676)
  v_7 = (-1.3017, -2.7029)
\end{verbatim}

Then we set up the picture environment (within a \texttt{figure} environment, so we don’t need to do anything to limit the scope of the \texttt{\unitlength} assignment) with the following code:

\begin{verbatim}
2. The \texttt{guIt} home page is going to change to \url{http://www.guitex.org}. At the time of writing this transfer has not been done yet.
\end{verbatim}

\begin{figure}
  \centering
  \begin{tikzpicture}
    \draw (0,0) -- (1,0) -- (2,0) -- (3,0) -- (4,0) -- (5,0) -- (6,0) -- cycle;
  \end{tikzpicture}
  \caption{Various end point for segments of equal length}
\end{figure}

\begin{figure}
  \centering
  \begin{tikzpicture}
    \draw (0,0) -- (1,0) -- (2,0) -- (3,0) -- (4,0) -- (5,0) -- (6,0) -- cycle;
    \draw (0,0) -- (1,0) -- (1.5,0) -- (0,0);
  \end{tikzpicture}
  \caption{Various joins}
\end{figure}

3\ Examples

There are dozens of examples in the \texttt{guIt} documentation, (Beccari, 2011), where every line art picture has a small legend containing the author name and the program used for producing it.

1. In the documentation, (Gäsßlein et al., 2009), this command is erroneously spelled \texttt{\miterjoin}.  

60
Some splines

We draw some splines inside a square with sides 6cm long; a quadratic spline has its two nodes at the square base vertices, and the control node at the center of the upper side. A cubic spline uses the four square vertices as end and control nodes:

\begin{picture}(6,6)(-3,-3)
\put(-3,-3){\framebox(6,6){}}
\polyline(-3,-3)(0,3)(3,-3)
\polyline(-3,-3)(-3,3)(3,-3)(3,3)
\linethickness{1.5pt}
\qbezier(-3,-3)(0,3)(3,-3)
\cbezier(-3,-3)(-3,3)(3,-3)(3,3)
\end{picture}

Figure 4 displays the result; if you can read this document on the screen, you can magnify the picture and you can check the vector nature of such splines. You may observe the effect of the \texttt{\linethickness} assignment on the splines themselves. Figure 3 contains also the polylines that join the nodes and control points, so that it’s easier to see the effect of these “control segments”.

An electric circuit

Many years ago, at the end of the eighties, when only the \texttt{picture} environment was available, I needed to draw circuit diagrams; I had so many circuit diagrams to insert in my book that I needed to create suitable macros for drawing the circuit components and their connections to the various circuit nodes; of course every component had to be identified with a symbol and optionally should be assigned a value with the proper units.

Nowadays there are modular packages to work with TikZ, circuitikz, [Redaelli](2009), or with PSTricks, pst-circ, [Voss](2011); at that time there was nothing, or at least nothing I was aware of.

In my department there was a very good expert of technical drawing, and for my previous books I had asked him to draw my circuits; these drawings had to be glued to the camera ready copy, because at that time it was very difficult to insert graphical files into a document; not impossible, but difficult. The publisher, in any case, did not want any kind of file; he wanted only camera ready copies. This procedure was pretty lengthy; draw my circuits by hand, pass them to the technician with suitable descriptions about dimensions, lettering, line thicknesses, and the like; after the drawings were done, careful control of the correctness, the proper position of the labels, the right drop of indices, and any possible typo; start again with the second drafts, and so on.

Therefore I decided to write a personal package containing all the circuit macros; that had to work as an interface between the user and the \texttt{picture} environment with its internal macros. It took about two weeks; afterwords I had an almost complete circuit drawing \TeX{} program. At that time, of course, arbitrary sloped lines were done by juxtaposition of a multitude of little dots, as well as quarter, half and full circles of any diameter. One port components were automatically drawn as vertical or horizontal elements; connections would make automatically the necessary bends in order to reach the destination nodes; two ports and four-pole devices were set in the proper orientation in order to avoid crossing their connections; opera-
tional amplifiers, nullators, norators and nullors were correctly designed; block diagrams with their signal flows, their branching nodes, their summing points, and so on, were at hand. The unit length was parametrized to the current font ‘ex’ unit, so that the circuit diagram would scale together with the size of the surrounding text font.

I saved a lot of time using my macros and the technical expert eventually congratulated with me admitting that my drawings were more consistent than his own.

When the important \texttt{pict2e} package became available in 2003, I started to eliminate all references to the old tiny-point-overlay technique, and promptly switched to the new technology.

I eventually added the logical components, so that this private package is almost complete. What is provided by \texttt{circuitikz} is much more complete, and this is the main reason why my package remains private. I keep using it for no other reason than compatibility with the past book files I wrote long ago, from which I often pick up some parts in order to assemble short tutorials for students that ask me some explanations.

Actually the package is too long to publish; therefore I will not show how the user commands are realized with the internal modern \texttt{picture} environment ones. I show just the source code for drawing the circuit diagram of a band elimination filter:

\begin{verbatim}
\begin{circuito}(75,35)
\hconnect(0,0)(19,0)\HPolo(20,0)(65,0)
\polo(20,25)[30,25]\polo(65,25)[75,25]
\hconnect(66,0)(75,0)
\R(75,0)(75,25){R\text{\scriptsize ped}(L)}
\E(0,0)(0,25){E}
\R(0,25)(19,25){R\text{\scriptsize ped}(G)}
\serie*(30,0)(21,25)C{C_1}-L{L_1}
\parallelo(30,25)(55,25)L{L_2}-C{C_2}
\serie(55,0)(64,25)C{C_3}-L{L_3}
\nodi(55,0)(55,25)(30,25)(30,0)
\end{circuito}
\end{verbatim}

and you can see the result in figure 5. As you can see the component and connection macros are really user friendly and the total amount of code for drawing a complicated circuit diagram\footnote{Actually the circuit diagram is not complicated at all; the complication is hidden in the user macros, especially those for inductors, where the various Bézier cubic splines are properly described and connected to one another.} is very limited; if you are reading this file on screen you can magnify the image of figure 5 and verify that the whole drawing is a scalable vector one. You can also recognise that the resistors are drawn by means of the \texttt{polyline} command with the \texttt{miterjoin} specification for the connection of the various segments.

### A Cartesian diagram

While teaching the synthesis of electrical circuits I often needed Cartesian diagrams of their performances; in figure 6 the squared magnitude of a fifth order elliptical filter characteristic function is plotted. The name “elliptical” derives from the use of first and second kind of elliptical integrals and functions. The diagram is just qualitative, although it would not have been a problem to compute the actual points by means of a suitable program, for a qualitative diagram the extreme points and the peaks and zeros should be sufficient. The whole diagram had to be also shown with a beamer, therefore a \texttt{beamer} presentation was made containing the same code:

```latex
\unitlength=0.9mm
\begin{picture}(80,60)(-40,-5)
\VECTOR(-40,0)(40,0)
\Zbox(40,-2)[tr]{\omega}
\VECTOR(0,-1)(0,55)
\Zbox(-1,55)[tr]{|F|^2}
\multiput(-35,5)(4,0){18}{\line(1,0){2}}
\Zbox(2,7)[bl]{1}
\multiput(-35,45)(4,0){18}{\line(1,0){2}}
\Zbox(1,46)[bl]{H^2}
\multiput(-2,15)(4,0){4}{\line(1,0){2}}
\LINE(-10,0)(-10,-1)
\Zbox(0,-2)[t]{0}
\Zbox(-10,-2)[t]{-1}
\LINE(10,0)(10,-1)
\Zbox(0,-2)[t]{1}
\linethickness{1.5pt}%
\cbezier(-12.5,55)(-12,40)(-12,40)\(-12,40)%\(-12,40)%\(-10,5)%\(-10,5)%\(-10,1,0)(-9.75,0)%\(-9.5,0)%\(-9.5,0)(-9,0)(-9,5)%\(-8,5,5)%\(-8,5.5)%\(-7.75,5)%\(-7.75,0)%\(-7,0)%\(-7,0)(-5.5,0)(-5.5,5)%\(-4,5)%\(-4,5)%\(-2,5)%\(-2,0)%\(-2,0)%\(-4,5)%\(-4,5)%\(-13,5,5)%\(-13,75,45)%\(-13,75,45)%\(-14,5,45)%\(-14,5,45)%\(-15,75,45)%\(-15,75,45)%\(-17,55)%\(-17,55)%
\end{picture}
```

**Figure 5:** A band elimination filter
L'approssimazione ellittica funzioni ellittiche danno risultati del tipo seguente:

Le specifiche di progetto approssimate con le

I parametri scattering
L'approssimazione di
L'approssimazione della trav smettere

As I said in the Introduction the picture environment is a very simple one, with few and specific drawing commands; the documentation is so simple that less than 10 pages suffice. At the same time these simple commands may be used to create more complex macros and eventually produce professional drawings. Certainly this environment cannot compete with more elaborate ones, such as those provided by the packages TikZ and PSTricks; but the latter ones require a steep learning curve, while the former one can be mastered in a few minutes. Very often the results obtained with the picture environment, completed by the recent enhancements provided by the pict2e package, are fully acceptable: it's possible to create complicated diagrams as well as simple symbols; it is possible to use this environment to place background or foreground images or symbols "wherever" on the page, even outside the margins; in [Beccari (2011) it is shown also how to make strange and unusual tables, Cartesian diagrams, and any sort of mix between line art and included pictures. In any case if any lettering is placed in the drawing, it surely uses the same fonts as those used for the text, thus eliminating the usual risk that may occur when using external drawing software.

I believe that beginners would find this enhanced environment the right first step for programmed drawings; with minimum effort they can reach really good results.

References

Beccari, C. (2007). «I font per le slide I\TeX resuscitati», As\TeXnica, (4).

Beccari, C. (a cura di) (2011). Introduzione all’arte della composizione tipografica con I\TeX. G.Fr.


Gäßlein, H., Niepraschk, R. e Tkadlec, J.
(2009). «The pict2e package». \TeX\ documentation attached to the pict2e.sty package.


Niepraschk, R. (2010). «The eso-pic package». \TeX\ documentation attached to the eso-pic.sty package.


Tantau, T. (2010). «The TikZ and pgf packages». \TeX\ documentation attached to the tikz.sty package.

Van Zandt, T. (2003). «Pstricks – PostScript macros for generic \TeX\». \TeX\ documentation attached to the pstriks.sty package.

Voss, H. (2011). «pst-circ a PSTricks package for drawing electric circuits». \TeX\ documentation attached to the pst-circ.sty package.


Wilson, P. e Robertson, W. (2009). «The layouts package». \TeX\ documentation attached to the layouts.sty package.

> Claudio Beccari
Villarbasse (TO)
claudio dot beccari at gmail dot com