The \texttt{xpicture} package
Robert Fuster

Abstract
The \texttt{xpicture} package extends the graphic abilities of the standard \LaTeX\ environment \texttt{picture} and the packages \texttt{pict2e} and \texttt{curve2e}, adding the ability to work with arbitrary reference systems, Cartesian or polar coordinates. Furthermore, in addition to drawing lines, vectors, polygons and polylines, this package allows you to draw conic sections and arcs, graphs of functions and parametrically defined curves. These curves are composed by using quadratic Bézier approximations automatically determined using the \texttt{calculator} and \texttt{calculus} packages.

1 Introduction
Since Leslie Lamport first included the possibility of composing pictures in \LaTeX, the capabilities in this area have increased in several ways, as you can see by taking a look at The \LaTeX\ Graphics Companion (Goossens, Mittelbach, Rahtz, Roegel, and Voß, 2008) or at related sections on CTAN.¹

Among these many choices, those exploiting the graphical capacity of PostScript (or PDF) have proved to be the most productive. In this area, we find different solutions.² This is what the package \texttt{pict2e} (Gäßlein, Niepraschk, and Tkadlec, 2011) does, solving the limitations in the standard environment \texttt{picture} by abandoning the technique of building pictures using special fonts, instead adopting a driver-oriented technique. The \texttt{curve2e} package (Beccari, 2012) redefines some \texttt{pict2e} commands and introduces more drawing facilities that allow drawing circular arcs and other curves.

The present \texttt{xpicture} package (Fuster, 2012b) retains the coordinate-oriented approach of \texttt{picture} while adding several features, the most important of which are the ability to work with various reference systems and to draw curves (including real functions) using their parametric equations. This package requires several packages: \texttt{curve2e, xcolor} (Kern, 2007) to handle colors, and \texttt{calculator} and \texttt{calculus} (Fuster, 2012a), to define functions and perform calculations.

We present the \texttt{xpicture} package in section 2, briefly describing its main features. In section 3 we discuss how some features of the package have been implemented, particularly the change of coordinate system and the drawing of curves. Finally, some conclusions and ideas about future work will be described in section 4.

2 The \texttt{xpicture} package
The \texttt{xpicture} package introduces several new graphical instructions, and some enriched versions of standard features provided by the \texttt{picture} environment. All these new instructions can be classified as follows:

(a) Declaration and use of different reference systems, using both Cartesian and polar coordinates.

(b) The \texttt{Picture} environment, an alternative to the \texttt{picture} environment, compatible with the new reference systems.

(c) Instructions to show Cartesian or polar frames and grids.

(d) Alternative instructions or extensions of the standard \texttt{picture} commands and those defined by the packages \texttt{pict2e} and \texttt{curve2e}:
- Enriched versions of the commands \texttt{\put} and \texttt{\multiput}, providing adequate control of the precise position in which objects are composed.
- Instructions for drawing straight segments, vectors (in any direction and using any reference system), polygonal lines, and regular or arbitrary polygons.

(e) Regular curves:
- Instructions for drawing conic sections (i.e., circles, ellipses, hyperbolas and parabolas) and arcs of these curves.
- Instructions for graphing functions and parametrically defined curves.

In the following subsections we will describe some of these features. Many of the examples we will show below are incomplete and only show the instructions that these examples try to describe. For example, although the drawings should be included in a \texttt{Picture} environment and they require the \texttt{\unitlength} length, in some examples this is not explicitly shown.

2.1 Reference systems and coordinates
By a reference system we mean an affine reference system, i.e., a point \(O(a,b)\) (the origin) and two linearly independent vectors, \(\mathbf{u} = (u_1, u_2)\) and \(\mathbf{v} = (v_1, v_2)\). If \((x_1, x_2)\) are the coordinates of a point \(P\) with respect to this reference system, then the standard coordinates of \(P\) are \((x_1, x_2)\), where

\[
\begin{pmatrix}
  x_1 \\
  x_2
\end{pmatrix} = \begin{pmatrix}
  a & b \\
  u_1 & v_1 \\
  u_2 & v_2
\end{pmatrix} \begin{pmatrix}
  x_1 \\
  x_2
\end{pmatrix}
\]

¹ http://www.ctan.org/topic/graphics-in-tex,
http://www.ctan.org/topic/graphics-curve,
http://www.ctan.org/topic/graphics-3d, and others.

² Some of which are quite sophisticated, such as PSTricks (Van Zandt, 2000) and PGF (Tantau, 2010).
The \texttt{xpicture} package includes several new declarations to change the reference system. Our first example shows the geometric transformation produced by the declaration
\begin{verbatim}
\referencesystem(3,3)(1,0)(2,-2)
\end{verbatim}
Here, \((3,3)\) is the new origin of coordinates and the new coordinate vectors are \((1,0)\) and \((2,-2)\).

Using multiple reference systems is one of the strongest features of \texttt{xpicture}. Among other applications, the choice of the reference system allows you to display the graphic effect of several geometric transformations, using different scales in each of the coordinate axes, display inverse functions, etc.

Points can be referred to by their Cartesian or polar coordinates (always with respect to the active reference system). In addition, angles can be measured in both radians and degrees.

\subsection{The new \texttt{Picture} environment}

The \texttt{xpicture} package does not change the behavior of standard \LaTeX\ commands for drawing (more precisely, it does not change the behavior of these commands as packages \texttt{pict2e} and \texttt{curve2e} have redefined them). Instead, it introduces new commands and environments, with a syntax similar to the standard ones. In particular, the new environment \texttt{Picture} (or \texttt{xpicture}) is used as an alternative to the standard \texttt{picture} environment. By using \begin{verbatim}
\begin{Picture}\[(color)\((x0,y0)\)(x1,y1)\]
\end{verbatim}
we fix the drawing area \([x0,y0] \times [x1,y1]\), referred to as the active reference system. More precisely, this environment defines a \texttt{picture} box that circumscribes this drawing area. If the optional argument is used, the background is colored in the given \texttt{color}.

\begin{verbatim}
\referencesystem(0,0)(1,-0.5)(1,0.5)
\fbox{\begin{Picture}[lightgray][%
\cartesiangrid(-3.5,-3.5)(3.5,3.5)
\end{Picture}}
\end{verbatim}

\begin{verbatim}
\begin{Picture}(-2,-2)(6,6)
\mypentagon
\end{Picture}
\begin{Picture}(-6,-3)(6,3)
\mypentagon
\end{Picture}
\end{verbatim}

\subsection{Showing coordinate systems}

Cartesian and polar axes and grids can be easily drawn, with widely customizable lines, cuts and labels, as shown in the following examples.

\begin{verbatim}
\begin{Picture}(-6,-4)(6,4)
\externalaxes
\renewcommand{\ticssize}{3pt}
\renewcommand{\secundarticsize}{1.5pt}
\renewcommand{\axeslabelsize}{\scriptsize}
\renewcommand{\axeslabelmathversion}{bold}
\renewcommand{\xunitdivisions}{3}
\renewcommand{\yunitdivisions}{2}
\cartesianaxes(-5,-3)(5.25,3.25)
\end{Picture}
\end{verbatim}
2.4 Extensions of standard picture commands

Despite the special syntax of `\makebox` inside a picture environment, the precise positioning of objects that are not strictly graphics (for example, labels or formulas) is a bit complicated and often depends on the `\unitlength` value.

The xpicture package defines several new commands, extending `\put` and `\multiput` in several ways. First, coordinates refer to the active reference system. Second, the precise position of that object with respect to the reference point is fixed by a new argument supporting multiple values: a number (interpreted as an angle with respect to the reference point), some of the keys `c`, `t`, `b`, `br`, `rtr`, ... (to similar effect as those keys in other commands), or compass keys `N`, `S`, `SE`, `ENE`, ... In addition, Cartesian and polar coordinates are allowed.

\begin{picture}(-5,-5)(5,5)
\degreespolarlabels
\polargrid{4.25}{24}
\end{picture}

\begin{picture}(-5,-5)(5,5)
\polargrid{4.25}{16}
\end{picture}

\begin{picture}(-2.5,-2.5)(2.5,2.5)
\renewcommand{\xunitdivisions}{5}
\renewcommand{\yunitdivisions}{5}
cartesiangrid(-2.25,-2.25)(2.25,2.25)
\end{picture}

\begin{picture}(-2.5,-2.5)(2.5,2.5)
\renewcommand{\xunitdivisions}{5}
\renewcommand{\yunitdivisions}{5}
cartesiangrid(-2.25,-2.25)(2.25,2.25)
\end{picture}

\begin{picture}(-5,-5)(5,5)
\degreesangles
\multiPut(1,0)(0,30){12}{\circle{0.05}}
\cPut{90}(1,90){\textsc{xii}}
\cPut{0}(1,0){\textsc{iii}}
\cPut{-90}(1,270){\textsc{vi}}
\cPut{180}(1,180){\textsc{ix}}
\end{picture}

In the following example, the label \((x, \sin x)\) was placed at ENE of \((3\pi/4, \sin 3\pi/4)\) (these numbers were computed with the aid of the calculator package).

\begin{picture}(-5,-5)(5,5)
\degreespolarlabels
\polargrid{4.25}{16}
\end{picture}

\begin{picture}(-5,-5)(5,5)
\degreespolarlabels
\polargrid{4.25}{24}
\end{picture}

Robert Fuster
Commands to plot lines, vectors, polylines and polygons (including regular polygons) are provided.

\begin{Picture}(-4.5,-4.5)(4,4.25)
\externalaxes
\cartesianaxes(-4,-4)(4,4)
\thicklines
\xLINE(0,-4)(-4,0)
\xVECTOR(-2,-2)(-2,3)
\arrowsize{10}{4}
\xtrivVECTOR(-2,-2)(-1,-1)
\Put(1,-3){\xline(3,2){3}}
\Put(1,1){\regularPolygon{2}{10}}
\Put(1,1){\regularPolygon{2}{5}}
\Put(1,1)\%
\polarreference\degreesangles
\Polygon(2,0)(2,144)(2,288)(2,432)(2,576)}
\end{Picture}

2.5 Regular curves
Using the \textit{xpicture} package piecewise regular curves can be easily drawn. In a general way, you can draw any two-dimensional curve that can be described with parametric equations.

To simplify common needs, the package includes specific instructions to draw graphs of functions, polar curves and (arcs of) conic sections.

2.5.1 Conic sections and arcs
The \texttt{\Circle{⟨r⟩}} and \texttt{\Ellipse{⟨a⟩}{⟨b⟩}} commands draw the circle of radius \texttt{⟨r⟩} and the ellipse of semiaxes \texttt{⟨a⟩} and \texttt{⟨b⟩}, respectively.

\begin{Picture}(-5.5,-4.5)(5.5,4.5)
\externalaxes
\cartesianaxes(-5,-4)(5,4)
\rotateaxes{\numberQUARTERPI}
\Ellipse{4}{3}
\Polyline(0,3)(0,0)(4,0)}
\end{Picture}

Parabolas and hyperbolas, being unbounded curves, require two additional parameters, \texttt{⟨xmax⟩} and \texttt{⟨ymax⟩}, in order to delimit the portion of the curve to be drawn. Moreover, in the case of hyperbolas, you can draw either just one or both branches.

\begin{Picture}(-5.5,-4.5)(5.5,4.5)
\setlength{\unitlength}{0.5cm}
\renewcommand{\axeslabelsize}{\scriptsize}
\begin{Picture}(-5.5,-4.5)(5.5,4.5)
\externalaxes
\cartesianaxes(-5,-4)(5,4)
\rotateaxes{\numberQUARTERPI}
\Ellipse{4}{3}
\Polyline(0,3)(0,0)(4,0)}
\end{Picture}
\end{Picture}

Commands for drawing arcs of all kinds of conic sections are also provided.

2.5.2 Graphs of functions
In order to draw the graph of a function you first need to define the function by using the tools that the \textit{calculus} package provides. Then you can draw the graph simply by using the command \texttt{\PlotFunction}.

The following example shows the graph of the function \(f(t) = t \cos t, -10 \leq t \leq 10\). We define it as the product of two already known functions (the identity and cosine functions), and call it \texttt{\Function}{\Function}:

\begin{Picture}(-16,6.4)(16,6.4)
\setlength{\unitlength}{0.5cm}
\begin{Picture}(-16,6.4)(16,6.4)
\xLINE(16,6.4)(-16,-6.4)
\xline(16,6.4)(-16,-6.4)}
\end{Picture}
\end{Picture}
Then, the instruction
\PlotFunction[30]{\Ffunction}{-10}{10}
does all the work to draw the curve.\(^3\)

\begin{Picture}(-11,-11)(11,11)
  \makenotics\makenolabels
  \cartesianaxes(-10,-10)(10,10)
  \printxticlabel{\numberPI}{\pi}
  \printxticlabel{\numberTWOPI}{2\pi}
  \printxticlabel{\numberTHREEPI}{3\pi}
  \printxticlabel{-\numberPI}{-\pi}
  \printxticlabel{-\numberTWOPI}{-2\pi}
  \printxticlabel{-\numberTHREEPI}{-3\pi}
  \printyticslabels{-10}{2}{10}
  \PRODUCTfunction{\IDENTITYfunction}{\COSfunction}{\Ffunction}
  \PlotFunction[30]{\Ffunction}{-10}{10}
\end{Picture}

\section{2.5.3 Parametrically defined curves}
You can declare the curve \(x = f(t), \ y = g(t)\) with the \PARAMETRICfunction declaration. Assuming that functions \(f\) and \(g\) are stored in the commands \(\text{XFunction}\) and \(\text{YFunction}\), we can declare the new parametric function \(\text{MyParametricFunction}\) like this:

\PARAMETRICfunction{\text{XFunction}}{\text{YFunction}}{\text{MyParametricFunction}}

To print it, use the \PlotParametricFunction command. An example:

\begin{equation*}
  \text{<MyParametricFunction>0}{\text{<MyPolarFunction>}}
  \Put[t](0,1){\text{The Archimedean spiral } \rho = 0.5\phi}
\end{equation*}

\section{2.5.4 Polar curves}
Polar curves \(\rho = f(\phi)\) are a particular case of parametrically defined curves. But, instead of defining the two functions \(x = \rho \cos \phi\) and \(y = \rho \sin \phi\), you can directly declare the function \(y = f(t)\) as a polar function. Assuming that the function \text{MyFunction} is defined, you can declare the polar curve \(\rho = f(\phi)\) with

\begin{equation*}
  \POLARfunction{\text{MyFunction}}{\text{MyPolarFunction}}
\end{equation*}

Then you can draw this curve as a parametric curve.

The Lissajous curve \(x = \sin 3t, \ y = \sin 4t\)

\begin{equation*}
  \text{SCALEVARIABLEfunction}{3}{\text{SINfunction}}{\text{XFunc}}
  \text{SCALEVARIABLEfunction}{4}{\text{SINfunction}}{\text{YFunc}}
  \text{PARAMETRICfunction}{\text{XFunc}}{\text{YFunc}}{\text{MyParametricFunction}}
  \MULTIPLY{10}{\text{<numberPI>}}{\text{<numberTENPI>}}
  \setlength{\text{unitlength}}{0.4\text{linewidth}}
  \text{linethickness}{1pt}
  \begin{Picture}(-1,-1)(1,1.2)
    \text{PlotParametricFunction}[24]{\text{MyParametricFunction}}{0}{\text{<numberTWOPI>}}
    \Put[t](0,1){\text{The Lissajous curve } x = \sin 3t, \ y = \sin 4t} \end{Picture}
\end{equation*}

\(^3\) The optional argument here means that the curve is approximated by 30 small quadratic pieces.

Robert Fuster
2.5.5 Drawing curves from a table of values

All the instructions to draw curves described here are based on the \texttt{\textbackslash qCurve} command, which draws quadratic Bézier curves. Specifically, 

\texttt{\textbackslash qCurve}(x_0,y_0)(u_0,v_0)(x_1,y_1)(u_1,v_1)

draws a smooth curve between the points \((x_0,y_0)\) and \((x_1,y_1)\), with tangent vectors \((u_0,v_0)\) and \((u_1,v_1)\), respectively.

More generally, \texttt{\textbackslash PlotQuadraticCurve} draws a curve through several points.

With the \texttt{\textbackslash PlotQuadraticCurve} command you can approximate any smooth curve passing through a list of points when you know the tangent vectors. As a particular case of special interest (at least in a calculus course), the \texttt{\textbackslash PlotxyDyData} draws the graph of a function of a real variable from a table of values of the function and its derivative.

\texttt{\textbackslash PlotxyDyData}(0,0,2)(1,1,0)(2,2,3)
\texttt{(3,4,0)(5,1,-2)}

3 Implementation notes

The \texttt{xpicture} package loads the \texttt{curve2e} (and, then, \texttt{pict2e}), \texttt{xcolor}, \texttt{calculator} and \texttt{calculus} packages. The packages \texttt{curve2e} and \texttt{xcolor} are used as our interface with PostScript or PDF; so, \texttt{xpicture} is compatible with \texttt{dvips}, \texttt{DVIPDFMx}, \texttt{pdflatex}, \texttt{LuaLaTeX}, \texttt{XeLaTeX} or any other \LaTeX{} successor supporting these packages.

The abilities of \texttt{calculator} are used to change the reference system (applying affine transformations in such a way that coordinates of points and vectors are internally converted to its standard coordinates), to convert from polar to rectangular coordinates, to

\texttt{\textbackslash PlotQuadraticCurve}(1,0)(1,0)(0,1)(0,1)
\texttt{(-1,0)(-1,0)(0,-1)(0,-1)}
\texttt{(1,0)(1,0)}
\texttt{\textbackslash rotateaxes}{45}
\texttt{\textbackslash PlotQuadraticCurve}(1,0)(1,0)(0,1)(0,1)
\texttt{(-1,0)(-1,0)(0,-1)(0,-1)}
\texttt{(1,0)(1,0)}

\texttt{\textbackslash PlotxyDyData}(0,0,2)(1,1,0)(2,2,3)
\texttt{(3,4,0)(5,1,-2)}

4 In earlier versions (only privately distributed at Universitat Politècnica de València), there was an option (not recommended) to omit loading \texttt{curve2e} and \texttt{pict2e}, so it was possible to compile the document as a pure \texttt{dvi}. But the quality of documents obtained was very poor and build time increased considerably. Therefore, and given that probably there is no reason to justify producing a \texttt{dvi} document instead of PostScript or PDF, that option has been abandoned.

The \texttt{xpicture} package
compute the precise position required by the \texttt{\Put} and \texttt{\multiPut}-like commands, and more.

Straight lines and vectors, once their standard coordinates have been calculated, are processed by \texttt{curve2e} commands.

Curves are locally approximated by quadratic Bézier splines, using the \texttt{\bezier} command. This means that every curve is made up of several small quadratic approximations. A quadratic Bézier is determined by three control points: the endpoints of the curve, $P_0$ and $P_1$, and the point $P_r$ where tangent lines at the endpoints intersect.

The \texttt{\bezier}(x_0, y_0)(x_r, y_r)(x_1, y_1) command draws the Bézier curve whose control points are $P_0(x_0, y_0)$, $P_r(x_r, y_r)$ and $P_1(x_1, y_1)$.

However, the \texttt{calculus} package can determine the points $P_0(x_0, y_0)$ and $P_1(x_1, y_1)$ that lie on the curve and the tangent vectors, $\vec{u}$ and $\vec{v}$, at these points.

Then, to determine $P_r$, \texttt{xpicture} uses \texttt{calculator} to find the intersection of the two tangent lines, i.e., it solves the linear system

\begin{align*}
    u_1 y - u_2 x &= u_1 y_0 - u_2 x_0 \\
    v_1 y - v_2 x &= v_1 y_1 - v_2 x_1
\end{align*}

This is, essentially, what the command \texttt{\qCurve} does. All other commands to draw curves use \texttt{\qCurve}. For example, the command

\texttt{\PlotParametricFunction[n]{\xFV}{\a}{\b}}

divides $[a, b]$ in $n$ pieces, $[t_{i-1}, t_i]$ ($1 \leq i \leq n$), computes $F(t_{i-1}), F'(t_{i-1}), F(t_i)$ and $F'(t_i)$, and calls \texttt{\qCurve}(F(t_{i-1}), F'(t_{i-1}))(F(t_i), F'(t_i)). If the tangent lines are parallel and not coincident, then it subdivides the interval into two half pieces.

\section{Conclusions and future work}

The aim of this package is not to compete with advanced general purpose drawing packages, but, first, to complete the abilities of the \texttt{picture} environment, by adding the capacity to use arbitrary reference systems and a greater control of the position of objects, and, second, to provide a solid tool for drawing scientific graphics.

Other well-known packages with the capability to draw graphs of curves from their equations (such as \texttt{pst-plot}, based on PSTricks (Van Zandt and Voß, 2013), \texttt{pgfplots}, based on PGF/Ti\textsc{k}Z (Feuersäger, 2012), and \texttt{LaPDF} (Reimers, 2011)) provide similar results. The differences between them lie in the way they make calculations to produce the graphs and in the syntax they use.

In this respect, the most interesting properties of \texttt{xpicture} are the following: the package is an extension of the standard \texttt{picture} environment and the syntax that is used is typical of \TeX; all calculations are performed directly by \TeX, by using the packages \texttt{calculator} and \texttt{calculus}; by using package \texttt{calculus} you can define virtually any elementary function or parametric curve; you can determine values of functions, lengths and other numerical parameters to fine-tune your picture, also by using \texttt{calculator} and \texttt{calculus}.

Finally, the most important feature of \texttt{xpicture} is, probably, its capability to change the reference system. As far as I know, no other package includes a general mechanism to manage arbitrary reference systems. The ability to change reference systems is a powerful tool for applying geometric transformations of objects without devoting a lot of effort to calculate coordinates. On the other hand, using this package you can draw virtually any curve that you can define by means of analytical equations. These are the most outstanding features of this package. Without reaching the high sophistication of packages based on PSTricks or PGF/TikZ, this package allows you to draw high quality graphics (especially scientific graphics) by using the standard \TeX syntax, hoping that users will feel comfortable without having to spend too much time learning to use it.

\subsection{Enhancements}

This package can be improved in several ways, some of them without expending too much effort. In the next version we will include extensions (compatible with the active reference system) of almost all commands of the standard \texttt{picture} environment and its extensions \texttt{pict2e} and \texttt{curve2e}. Namely, extensions of commands such as \texttt{oval}, \texttt{Curve} or \texttt{lineto}, using the active reference system, will be defined.
Three dimensional straight lines and curves can be easily drawn by using a linear transformation between three- and two-dimensional spaces. Developing this idea, we intend to build a three-dimensional version of the \texttt{xpicture} package.

The other area that we want to develop is the representation of graphs (in the sense of graph theory). The idea is to define objects of types \texttt{node}, \texttt{edge}, \texttt{arc}, and \texttt{graph} and mechanisms to graphically represent them, and to manipulate them in order to show their different attributes. To draw a graph using \texttt{xpicture} is not very complicated, but we would like to provide a tool to easily choose the best display of the graph. This is a medium term project.

As longer term projects, it may be interesting to implement nonlinear coordinate transformations, which would allow, for example, to use logarithmic scales; the ability to draw smooth curves passing through several points (not including vectors of direction) would be another interesting utility (this could be done, for example, if we could build interpolating polynomials).

Acknowledgements

Claudio Beccari was kind enough to take a look at this package and suggested several improvements. Thanks to him, and to Gabriel Valiente Feruglio, who both took the time to make useful comments about \texttt{xpicture}. And many thanks to Karl Berry and Barbara Beeton for their patient kindness in reviewing my manuscript.

References


Reimers, Detlef. \texttt{LAPDF}. Drawing in \TeX{} with PDF commands”. Available from CTAN, /macros/latex/contrib/lapdf, 2011.


\copyright{} Robert Fuster

Universitat Politècnica de València
Departament de Matemàtica Aplicada
Cami de Vera, 14
València E46022
Spain
rfuster (at) mat dot upv dot es
\url{http://www.upv.es/~rfuster/}