Some Useful Macros Which Extend the LaTeX `picture` Environment

A.S. Berdnikov, O.A. Grineva and S.B. Turtia
Institute of Analytical Instrumentation
Rizsky pr. 26, 198103 St.Petersburg, Russia
berd@ianin.spb.su,olga@ianin.spb.su,turtia@ianin.spb.su

Abstract

The ability to create pictures using TeX/LaTeX is relatively poor, and many extensions – (`epic/eepic`, `pictex`, `drawtex`, `xypic`, `mfpic`, etc.) – were created to extend this capability to a higher level. The package PMGRAPH.STY (*poor-man-graphics*) which is described herein is not as general as the ones mentioned above. Not being too complicated, these macros appear to be useful in our work, and it seems that they can be also useful for other TeX-users.

The pmgraph.sty Package

This package is based on the use of the pseudo-graphical fonts which are used by generic LaTeX without additional extensions — mainly because the variations of PiCTeX, METAFONT and new graphical font themes are already explored by other authors on a sufficiently higher level. To some extent the purpose of our work was to see how far it was possible to develop new useful graphical primitives for LaTeX *without* the investment of the external graphical tools.

The package PMGRAPH.STY offers the following features:

- vectors with a set of slopes which are as general as the line slopes are implemented in LaTeX;
- vectors with an arrow at the beginning, middle or end of the vector with various orientations of the arrow;
- circles and circular arcs with almost arbitrary diameters using magnified `circle` and `circlew` LaTeX fonts;
- 1/4 circular arcs correctly positioned at the center or at the corner;
- an extended set of frames, which includes various corner styles and optional multiple frame shadows with a variety of styles;
- tools which enable the user to extend the variety of the frame styles and the shadow styles as far as his/her imagination allows it; and
- automatic calculation of the picture size relative to the current width of the text — this includes `picture` environments inside list environments.

Vectors

The number of angles for inclined lines which can be used in LaTeX is limited to a great extent, but the number of angles for *vectors* is even more restricted. The variety of vectors can be increased if, instead of the *strictly* inclined arrows at the end of the inclined lines, arrows with an *approximate* inclination are added. Corresponding changes are incorporated in PMGRAPH where the relation between strict inclinations and approximate inclinations are shown in Table 1. The implementation required the modification of internal LaTeX commands such as `\@svector`, `\@getlarrow`, `\@getrarrow` and the user command `\vector` itself. As a result, the command `\vector` draws vectors for all inclinations valid for LaTeX lines as is shown in Fig. 1. The vectors are not as ideal as normally required by TeX standards, but the results are acceptable for all inclinations, except $(6,1)$.

$(1,1)$	$(1,1)$	$(4,1)$	$(4,1)$	$(5,3)$	$(3,2)$
$(2,1)$	$(2,1)$	$(4,3)$	$(4,3)$	$(5,4)$	$(4,3)$
$(3,1)$	$(3,1)$	$(5,1)$	$(4,1)$	$(6,1)$	$(4,1)$
$(3,2)$	$(3,2)$	$(5,2)$	$(3,1)$	$(6,5)$	$(4,3)$

Table 1: Relation between the line slopes and the approximate vector slopes

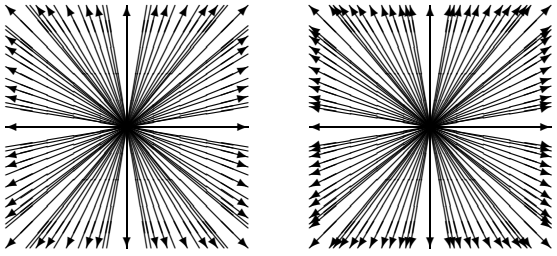

Figure 1: LaTeX and PMGRAPH vectors

LaTeX allows one to put an arrow only at the end of the vector. The `\Vector` command

offers the capability of placing *arbitrary* arrows with different orientations along the vector (see Fig. 2). The predefined arrow styles assign a letter to each position and orientation of the arrow along the `\Vector`. The arrows shown in Fig. 2 are drawn

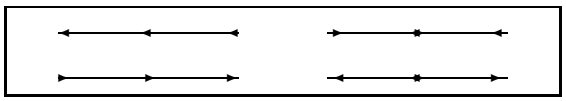

Figure 2: Multi-arrow vectors

by the commands

```
\begin{picture}(300,40)
  \put(20,5){\Vector[bme](1,0){100}}
  \put(20,30){\Vector[BME](1,0){100}}
  \put(170,5){\Vector[xmMZ](1,0){100}}
  \put(170,30){\Vector[XmMz](1,0){100}}
  . . . . . .
```

The letter `e` corresponds to an arrow with a 'normal' orientation at the end of the vector; `E` corresponds to an arrow with a reverse orientation. The letters `b` and `B` correspond to arrows at the beginning of the vector (with normal and reverse orientation); the letter `m` and `M` — to the arrows at the middle, etc. The optional parameter of the command `\Vector` contains a list of letters which describes the set of arrows along the `\Vector`. It is possible to create user-defined styles of arrows using the commands `\VectorStyle` and `\VectorShiftStyle` as described in the PMGRAPH documentation.

Circles

The range of the diameters for circles and disks (black circular blobs) available in LaTeX is very restricted. It can be enlarged by using magnified versions of the pseudo-graphical LaTeX fonts if the user does not have something better at his/her disposal such as `curves.sty`, PiCTeX or MFPiC. The disadvantage of the method presented here is that the width of the lines is magnified too, which is inconsistent with the rigorous TeX accuracy requirements, but for *poor-man-graphics* these circles can be satisfactory.

The scaling of circular fonts is performed by the commands

`\scaledcircle{`*factor*`}`
`\magcircle{`*magstep*`}`

which correspond to the TeX commands

`\font ... scaled` *factor*
`\font ... scaled \magstep` *magstep*

The valid *magstep* values are `0`, `h`, `1`, `2`, `3`, `4`, `5`. The values *factor=1000* and *magstep=0* correspond to the one-to-one magnification. The circle magnification, like other TeX commands, returns to its previous value outside the group inside which it was changed.

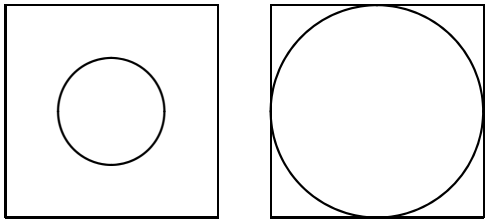

Figure 3: Magnified circles

In order to properly calculate the character code of the circle segment needed to build a circle, taking the magnification into account, it was necessary to redefine some more internal LaTeX commands such as `\@getcirc` and `\@circ`. To reflect in magnified fonts the changes of the line thickness, the commands `\thinlines` and `\thicklines` are also modified.

The example in Fig. 3 is produced by

```
\setlength{\unitlength}{1pt}
\begin{picture}(200,100)(-100,-50)
  \put(-50,0){\thicklines\circle{80}}
  \put(-50,0){\squareframe{40}}
  \magcircle{4}
  \put(50,0){\thinlines\circle{80}}
  \put(50,0){\squareframe{40}}
\end{picture}
```

where `\squareframe` is the user-defined command which draws the square with the specified side and the center at (0,0). It shows how the usage of the magnified circles enables one to overcome the upper limit of 40pt of the LaTeX circle diameter. It is necessary to note that following the magnification with `\magcircle{4}`, the thickness of the `\thinline` circles corresponds approximately to the thickness of the ordinary `\thickline` circles (`\magstep4` $\approx$ 2000).

Additional macros can draw 90° quarters of circles explicitly without tricky refinement of the parameters of the command `\oval`:

`\trcircle{`*diam*`}` $\longrightarrow$ `\oval[tr]...`
`\brcircle{`*diam*`}` $\longrightarrow$ `\oval[br]...`
`\tlcircle{`*diam*`}` $\longrightarrow$ `\oval[tl]...`
`\blcircle{`*diam*`}` $\longrightarrow$ `\oval[bl]...`

The center of the circular arc is positioned strictly at the point which represents the argument of the corresponding `\put`. The commands `\TRcircle`,

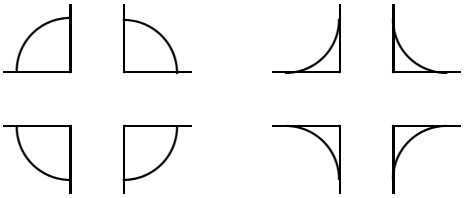

Figure 4: 90° circular segments

`\BRcircle`, `\TLcircle`, `\BLcircle` draw the 90° quarter circles with the reference point positioned at the corner instead of the center. Similarly, the commands `\tlsector`, `\TLsector`, `\blsector`, `\BLsector`, etc., draw circular segments together with horizontal and vertical radii. The proper positioning of the circular segments requires special precautions since it is necessary to take into account the line thickness and the specific alignment of the circular elements inside the character boxes.

The example in Fig 4 shows the usage of these commands:

```
\begin{picture}(200,60)(-100,-30)
  \put(-60,10){\thicklines\tlcircle{50}}
  \put(-60,10){\circle*{1}}
  \put(-60,10){\line(-1,0){25}}
  \put(-60,10){\line(0,1){25}}
  \put(40,10){\thicklines\BRcircle{50}}
  \put(40,10){\circle*{1}}
  \put(40,10){\line(-1,0){25}}
  \put(40,10){\line(0,1){25}}
  ... ... ...
```

The actual diameter of the circular segment is adjusted just as is done with the circles. The commands `\scaledcircle` and `\magcircle` also affect the thickness and the diameter of these circular segments.

Frames

The number of frames which is available in LaTeX is increased by PMGRAPH — besides the solid and dashed rectangular frames it is possible to draw double and triple frames in a variety of styles (Fig. 5). The commands `\frameBox`, `\ovalBox`, `\octalBox`, `\astroBox`, `\parquetBox` have the same structure as the command `\framebox`, but they draw the corresponding fancy frames:

```
\put(0,0){\ovalBox(100,50){oval}}
\put(70,0){\astroBox(100,50){astro}}
. . . . . .
```

An ordinary solid frame is drawn by `\frameBox`, the double and triple frames are drawn by `\frameBoX` and `\frameBOX`, respectively. Similar commands exist for double and triple fancy frames. The user can prepare personal macro commands to draw

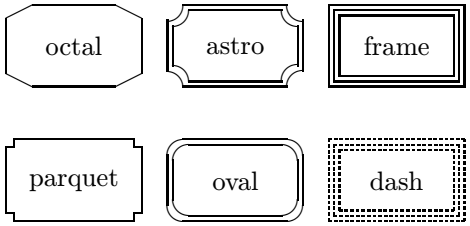

Figure 5: Examples of frame styles

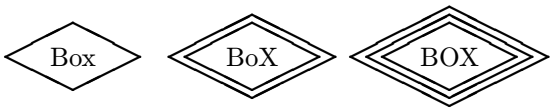

Figure 6: Romb-style frames

frame corners and extend the variety of fancy frames up to the limit of his/her imagination.

A more exotic variant of a frame can be created with the commands `\rombBox`, `\rombBoX` or `\rombBOX` as shown in Fig. 6. The style (i.e., inclination of the romb sides) and the distance between multiple rombs are set by the command `\rombboxstyle` with the default settings

```
\rombboxstyle(2,1,2pt)
```

The alignment of the romb around the box specified for these commands can be changed using an additional optional parameter (see the PMGRAPH manual for more details).

Each command to create a rectangular box has an optional parameter which specifies the "shadows" around the box. Each shadow style has a special letter, and a list of letters as the optional parameter results in a combination of the corresponding shadows. The standard shadow types are shown in Fig. 7. It is possible to draw several shadows of different types around an arbitrary corner of the frame as shown in Fig. 8:

```
\unitlength=10pt
\begin{picture}(20,15)
```

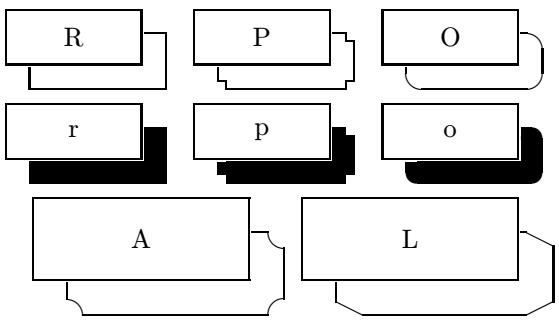

Figure 7: Examples of shadows

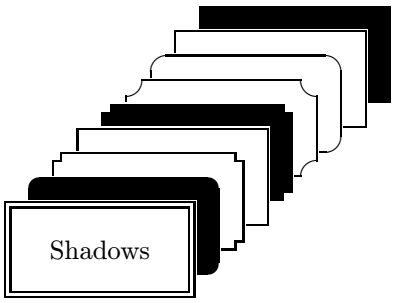

Figure 8: Multiple shadows

```
  \shadowcorner{B}
  \put(0,0){\frameBoX[oPR...](10,5){...}}
\end{picture}
```

The parameters of the shadows – thickness, corner size, additional shift, etc., – can be changed by user commands described in the manual.

Automatically scaled pictures

The idea behind the macros which are responsible for these functions is to calculate the `\unitlength` value of in terms of the *relative fraction* of the page width instead of explicitly specifying its value in points, centimeters, inches, etc.

The `\pictureunit[`*percent*`]{`*x-size*`}` command selects the value of `\unitlength` so that the picture, which is *x-size* units in width, occupies *percent*% of the width of the text. The `Picture` environment combines the automatic calculation of `\unitlength` with `\begin{picture}`–`\end{picture}`. The default, *percent*=100, corresponds to 90% of the text width. The default *percent* value can be adjusted by the command

`\renewcommand\defaultpercent{`*percent*`}`.

Examples:

```
\pictureunit[75]{120}
\begin{picture}(120,80)
...
\end{picture}

\begin{Picture}[75](120,80)
...
\end{Picture}
```

These macros are inspired by the `fullpict.sty` developed by Bruce Shawyer. Careful examination of the file `fullpict.sty` demonstrates some further bugs/warnings which require correction:

- each automatic scaling of `\unitlength` allocates a new counter;
- automatic scaling uses `\textwidth` as the reference width which results in improper functioning inside the list and `minipage` environments;
- the environments `fullpicture`, `halfpicture` and `scalepicture` are centered internally with `\begin{center}`—`\end{center}` which prevents the proper positioning of the picture in most cases.

The PMGRAPH.STY macros calculate the dimension `\unitlength` using the value `\hsize`, and as a result it works corectly also for `twocolumn` mode, inside the *list* environments `itemize`, `enumerate`, etc. (for example, all the figures in this paper are drawn using the environment `Picture`). The automatic centering and repeated allocation of the registers are corrected as well.

Acknowledgements

The authors are grateful to Dr. Kees van der Laan for the opportunity to present the preliminary results of our research at EUROTEX'95 (Aarnhem, The Netherlands). This research was partially supported by a grant from the Dutch Organization for Scientific Research (NWO grant No 07 – 30 – 07).