pst-layout

Michael Sharpe

msharpe at ucsd.edu

1 Quick overview

This package uses `pstricks` and related packages for a single purpose: to ease the design of quasi-tabular documents of a specified size. For a couple of examples, consider (a) a business card; (b) a form, assumed to be available as an `eps` graphic, that must be filled out by overprinting it with LaTeX text. A much more complex example that was constructed using this package is shown at
`http://cseweb.ucsd.edu/~gill/MissingGeoSite/index.html`.
In essence, you specify the canvas size and the units, you define a number of locations on the canvas to serve as alignment guides, and then enter arrays of parameters that govern the positions and appearances of dots, lines, frames, fragments of text and pictures. After entering these arrays of data, type, for example

```
\psset{unit=1in,picwidth=4in,picheight=6in}
\begin{pslayout}
<optional: other graphics commands>
\end{pslayout}
```

then run `LaTeX+dvips+ps2pdf` to view the result. Because the data arrays are not erased after the canvas is drawn, it is not intended as an environment for creation of more than a single canvas, though with care, that is possible. Units may be specified separately by setting `xunit` and `yunit` instead of `unit`.

A few packages are required.

`pst-node` A recent version is essential.

`pst-layout` This package.

`arrayjobx` This is almost the same as the `arrayjob` package that is part of TeX Live, but with the name `\array` altered in three places (lines 108, 148, 164) so as not to conflict with the usage of `\array` in LaTeX and related parts of `amsmath`, or with `\Array` in `pdftricks`. This command is normally only used internally, so there is little reason not to use `arrayjobx` in place of `arrayjob`.

2 Coordinates and Node Expressions

The origin is the southwest corner of the canvas so the x axis runs along the bottom edge and the y axis along the left edge. Two macros—`\xmax` and `\ymax` are defined within the `pslayout` environment, so that, for example, the top right corner may be referred to as `(\xmax,\ymax)`. Positions of points may be specified in any form acceptable to `\SpecialCoor`, or as a node expression, by which is meant an expression like

```
.25(1cm,3)+.333(2;90)-1.2([nodesep=.5cm]Q)
```

which specifies a linear combination of points (the items enclosed in parentheses) specified in any manner acceptable to `\SpecialCoor`. The parenthesized items themselves cannot themselves be node expressions, as those are not acceptable to `\SpecialCoor`. As a reminder, `\SpecialCoor` understands the following forms:

Cartesian Eg, `(3,4), (3pt,2)`.

Polar Eg, `(2;90), (1cm;45)`.

Mixed Eg, `(2;90 | 3,4), (P | Q)`—x coordinate from first point, y coordinate from second—very useful form in `pslayout`.

Node Eg, `(P)`, where `P` is a previously defined node.

Node relative Eg, `([nodesep=2pt,offset=1cm]P)`—`2pt` to the right and `1cm` up from `P`.

Node segment relative Eg, `([nodesep=1,offset=2pt]{P}Q`. Relative to directed line segment from `Q` toward `P`, 1 unit from `Q` and `2pt` to left (normal to `QP`.)

Postscript `(! <PostScript code>)` Code must leave two items on the stack, interpreted as x and y coordinates. Eg, `(! 90 sin 90 cos)` gives same result as (1,0).

Note that coordinates with an unspecified unit take the appropriate unit, so `(1cm,3)` means the point `1cm` to the right and 3 `yunit` from the origin.

2.1 Predefined points

Some points on the canvas are already named—the corners and midpoints are named `PSPbl` (bottom left), `PSPtl`, `PSPtr`, `PSPbr`, `PSPbl`, `PSPbc`, `PSPtc`, `PSPcl`, `PSPcr`, `PSPcc`, whose meanings should be clear. Three of these have alternative names—`R0=PSPtl`, `C-1=PSPbr`, and `C-2=PSPbc`. These names are special and should not normally be redefined, though you may if you are obstinate. If the choice seems odd, think of `R0` as the row above the first row of the canvas, `C-1` as the last column, and `C-2` as half way across. Names of the form `R<natural number>` and `C<natural number>` are reserved for special use. They should be defined by you as alignment points along the y axis and x axis respectively. Then `pslayout` constructs all possible pairs like `R15C2` (think of it as row 15, column 2) to serve as named positions. Do NOT use either `R` or `C` alone as the name of a point. Note that `R0C-1` will be the same as `PSTtr` and `R0C-2` will be the same as `PSTtc`.

3 An example file

```
%&latex
\documentclass[dvips,10pt]{article}
\usepackage[dvipsnames]{pstricks}
\usepackage{pst-layout}
\pagestyle{empty}
\setlength\parindent{0pt}
\begin{document}
% Remember that C-1, C-2, R0 are predefined---do not change
\NumPoints=2 % ignore higher indices
\PointName(1)={TR}\PointPos(1)={PSPtr}
\PointName(2)={R1}\PointPos(2)={(R0)-(0,.25)}
\def\MaxR{1}\def\MaxC{0}
\NumFrames=1
\FrameStart(1)={0,0}\FrameEnd(1)={PSPtr}
\NumFragments=1 %
\Frag(1)={Some text}\FragRefPt(1)={B}%Baseline, center
\FragPos(1)={R1C-2}
\psset{unit=1in,picwidth=3in,picheight=2in}
\begin{pslayout}
\end{pslayout}
\end{document}
```

Between `\begin{document}` and `\begin{pslayout}` one enters the data specifications.

NumPoints The entry `\NumPoints=2` directs the program to look for points with index from 1 to 2. Non-existent entries are ignored.

Point... The next two lines define the points with indices 1 and 2. The first line says, in effect, let `TR` be an alternative name for `PSPtr`. The second says to define `R1` to be .25(in) below `R0`, on the y axis.

MaxR, MaxC The following line instructs the program to create names of the for `R<i>C<j>` for `i<=1`, `j<=0`.

NumFragments The entry `\NumFrags=1` says to read `\Frag...` entries with index 1.

NumFrames The entry `\NumFrames=1` says to read `\Frame...` entries with index 1.

Frame(1) The items `\FrameStart(1)` and `\FrameEnd(1)` define opposite corners of a rectangular frame.

psset The `\psset{}` line defines the settings to be used in constructing the canvas.

pslayout The command `\begin{pslayout}` initiates processing the arrays of data and inserts them one by one onto the canvas, Following that, the the graphics commands between `\begin{pslayout}` and `\end{pslayout}`are performed, overprinting all that went before. (There are none in this example.) The red items in the following picture were added but are not shown in the example code.

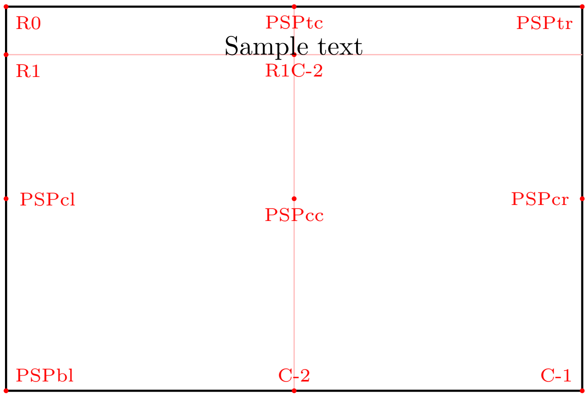

Figure 1: Output (including dots and labels)

The remaining sections describe in detail how to enter graphics, points, frames, lines, dots and text fragments. (That is the order in which they are processed.)

4 Graphics

External graphics must be in `.eps` format. The files should if possible be at natural size so that line widths and text sizes match those of other elements. The Graphics section may have the following items.

```
\NumGraphics=<maximum index>% obligatory
\Graphic(1)={<filename>}% omit the .eps
\GraphicPos(1)={<position>}% may be any form, even node expression
\GraphicOpts(1)={<list of options>}% eg, width=2in
\GraphicRefPt(1)={<reference pt>}% eg, bl, b
```

If `\Graphic(1)` is defined and non-empty, `\GraphicPos(i)` must be specified. If `\GraphicOpts()` and `\GraphicRefPt()` are empty, the effect is to place the graphic using the `graphicx` version of `\includegraphics`, so that its center is at `\GraphicPos()`. If `\GraphicOpts()` is not empty, its contents are passed along as options to `\includegraphics`. If `\GraphicRefPt()` is not empty, its contents are used to specify the reference point of the graphic—the point which is to be translated to `\GraphicPos()`. Typical values are `bl` (bottom left), `b` (bottom center).

5 Points

The Points section of the data may have the following elements.

```
\NumPoints=<maximum index>% obligatory
\PointName(1)={<your choice>}
\PointPos(1)={<position>}% may be any form, even node expression
\PointSeries(1)={<min>;<max>}% first, last indices
\PointInc(1)={<increment between points in series>}
```

If `\PointName(i)` is defined and non-empty, `\PointPos(i)` must be specified. The result is to create a node of specified name at the specified location. If `\PointSeries(i)` is defined, then `\PointInc(i)` must be defined. here is the meaning, by example.

```
\PointName(4)={P}
\PointPos(4)={1,2}
\PointSeries(4)={5;8}
\PointInc(4)={.1in,.2cm}
```

The result is that nodes named `P5...P8` are defined, with `P5` set to (1,2). Then the remaining nodes are set at increments of (.1in,.2cm), so that `P8` is at (1,2)+(.3in,.6cm).

While `\PointPos()` may be specified as a node expression, `\PointInc()` may not—it may however use any form understood by `\SpecialCoor`.

6 Frames

The Frames section of the data may have the following elements.

```
\NumFrames=<maximum index>% obligatory
\FrameStart(1)={<position>}% one corner
\FrameEnd(1)={<position>}% opposite corner
\FrameDelta(1)={<position>}% can specify width,height instead
\FrameOpts(1)={<list of options>}%eg, linewidth, linecolor
\FrameSolid(1)={<non-empty value for solid frame>}
```

If `\FrameEnd(i)` is defined and non-empty, `\FrameDelta(i)` is ignored, otherwise the latter must be specified. The value of `\FrameSolid` determines whether the frame is solid not. If `\FrameSolid(i)` is non-empty (`{1}`, say) the result will be a solid frame with color given by `linecolor`, otherwise there is an outer rectangular frame in `linecolor`, and the interior is filled in `fillcolor` using `fillstyle`. For example.

```
\FrameStart(4)={2,3}
\FrameDelta(4)={1cm,2}
\FrameOpts(4)={fillstyle=solid,fillcolor=yellow}
```

results in a black frame around a solid yellow interior. (By default, `fillstyle =none,fillcolor=white`.)

7 Lines

The Lines section of the data may have the following elements.

```
\NumLines=<maximum index>% obligatory
\LStart(1)={<position>}% initial point
\LEnd(1)={<position>}% other end
\LDelta(1)={<position>}% can specify increment instead
\LOpts(1)={<list of options>}%eg, linewidth, linecolor
\LArrow(1)={<arrow specification>}
```

If `\LEnd(i)` is defined and non-empty, `\LDelta(i)` is ignored, otherwise the latter must be specified. The value of `\LArrow` determines the arrows, if any. The default here is `{-}`, an ordinary line with no arrows. Other common options are `{->}`, `{<->}`. The value of the option `arrowscale` (default value 1) affects the size of the arrowhead.) All of `\LStart()`, `\LEnd()`, `\LDelta()` may be specified as node expressions. For example.

```
\LStart(4)={2,3}
\LDelta(4)={1cm,2}
\LOpts(4)={linewidth=.5pt,linecolor=red}
\LArrow(4)={->}
```

results in a red line from (2,3) to (2,3)+(1cm,2) with an arrowhead at the latter.

8 Dots

The Dots section of the data may have the following elements.

```
\NumDots=<maximum index>% obligatory
\DotPos(1)={<position>}% position
\DotOpts(1)={<list of options>}%eg, linecolor, dotstyle, dotsize
```

The position may be specified by a a node expression. The options available for a dot are many. By default, `dotstyle=*`, which makes a solid circular dot. Other common choices are `dotstyle=o` and `dotstyle=Bo`, both of which produce circular dots, the latter a little bolder. With the default values of `linecolor` and `fillcolor`, you get a black edge with white fill. The size depends on `dotscale` and `dotsize`. See the `pstricks` documentation.

9 Fragments

The Fragments section of the data may have the following elements.

```
\NumFragments=<maximum index>% obligatory
\Frag(1)={<text fragment>}
\FragPos(1)={<position>}% where to position it
\FragRefPt(1)={<one of B, c, t, r, b, tr, tl, br, bl, Bl, Br, etc>}
%\FragRefPt can also be polar offset from \FragPos
\FragRotation(1)={<rotation angle>}% eg 90
\FragBlankBG(1)={<non-empty value blanks background>}
```

The position `\FragPos()` may be specified by a node expression. `\Frag()` specifies the text that is printed. (It may be a `\parbox` or other LaTeX construct.) The fragment is rotated through `\FragRotation()`, if specified. There are two different placement regimes, depending on the form of `\FragRefPt()`. If omitted (in which case it defaults to `c`) or it one of the string values listed above, then that reference point is translated to `\FragPos()`. On the other hand, if `\FragRefPt()` contains the `;` character, it is assumed to specify a polar offset from `\FragPos()`. For example,

`\FragRefPt(3)={8pt;30}` specifies a polar offset of `8pt` at an angle of 30 degrees. (The form `{;30}` uses the value of `labelsep` for the radius, which by default is `5pt`.) These behaviors correspond to those of the `pstricks` `\rput` and `\uput` macros respectively.

If `\Frag(i)` is specified, then `\FragPos(i)` is obligatory. The other items are optional, but the default value of `\FragRefPt(i)` is not usually appropriate in the `layout` context. Most commonly, you will wish to align text fragments by baseline, centered `{B}` or by baseline, left `{Bl}`. Columns of numbers may require `{Br}`.

10 Type

Dots, Frames and Lines have another feature not yet mentioned—an optional `Type`. There are arrays `\DotType()`, `\FrameType()` and `\LType()` which behave the same way in each case, so we discuss only `\DotType()`.

Suppose you have several different types of Dots you wish to use in your design, with different parameters. Instead of entering these parameters each time in `\DotOpts()`, you define a number of types incorporating those parameters. Let's suppose we want to use two different Dots that don't use just the default settings. We define two types, say `A` and `B`. In the document preamble, define them with, say,

```
\newpsobject{psdotA}{psdot}{dotstyle=Bo,dotsize=5pt}
\newpsobject{psdotB}{psdot}{dotstyle=o,dotsize=2pt}
```

Then, if `\DotType(i)={A}`, the Dot will use the first set of parameters. You may of course make changes and changes to those new defaults in `\DotOpts(i)`.

There is one further complication in using `\LType`. In order to prepare for a new Type `A`, you need to add two line to the preamble of your document, like

```
\newpsobject{pslineA}{psline}{linecolor=red,arrows=->}
\newpsobject{psrlineA}{psrline}{linecolor=red,arrows=->}
```

because `layout` may use the macro `\psrline` from `pst-node` instead of `\psline`.

11 Other issues

From the user's point of view, the `pslayout` environment works as follows.

- If there is a macro by the name `\AtLayoutStart`, insert its contents so that it applies to the entire layout. Eg,

  ```
  \def\AtLayoutStart{\small \psset{Linewidth=.6pt,arrowscale=1.5}}
  ```

- If `showfullcanvas` was specified by, eg,

  ```
  \psset{unit=1in,picwidth=3.5in,picheight=2in,showfullcanvas}
  ```

  then an invisible off-white frame is drawn around the canvas so that if white space is trimmed from the resulting graphic, no part of the canvas will be trimmed.

- All nodes defined by `\PointName()` are constructed.

- All external graphic files listed in `\Graphic()` are placed.

- If there is a macro by the name `\FrameSettings`, and if `\NumFrames>0`, its code is inserted here. Most commonly, this will be a `\psset{}` to control the appearance of all frames. It could be any `pstricks` command, if you needed to have an object appear before frames were drawn. The code is grouped so that it applies only within the Frames section.

- All frames defined by `\Frame...()` are constructed.

- If there is a macro by the name `\LineSettings`, and if `\NumLines>0`, its code is inserted here. Most commonly, this will be a `\psset{}` to control the appearance of all lines. It could be any `pstricks` command, if you needed to have an object appear before lines were drawn. The code is grouped so that it applies only within the Lines section.

- All lines defined by `\L...()` are drawn.

- If there is a macro by the name `\DotSettings`, and if `\NumDots>0`, its code is inserted here. Most commonly, this will be a `\psset{}` to control the appearance of all dots. It could be any `pstricks` command, if you needed to have an object appear before dots were drawn. The code is grouped so that it applies only within the Dots section.

- All dots defined by `\Dot...()` are drawn.

- If there is a macro by the name `\FragSettings`, and if `\NumFragments>0`, its code is inserted here. Most commonly, this will be a `\psset{}` to control the appearance of all fragments. It could be any `pstricks` command, if you need to have an object appear after dots are rendered but before fragments are drawn. The code is grouped so that it applies only within the Fragments section.

- All fragments defined by `\Frag...()` are drawn.
- All `pstricks` commands between `\begin{pslayout}` and `\end{pslayout}` are carried out. This the where you place items that must be on the top layer. In particular, the command `\showRC` will show the defined `R` and `C` nodes and their combinations using green lines, as a temporary guide in your design.
- The pair

  ```
  \begin{pslayout*}
  \end{pslayout*}
  ```

  behaves the same way as the `pslayout` environment, but clips all material to the boundary of the canvas.

12 Drawing at a fixed location on the paper

In LaTeX, this is handled most easily using the geometry package. See the last example below. You may use to print on custom-sized paper, placing the canvas as you wish by specifying the appropriate margin parameters and nohead to geometry so that the rectangle that remains is exactly the size of your canvas. Remember then to set `\parindent=0pt`.

13 Examples

The following would make an eps graphic the size of a standard business card.

```
%&latex
%process with latex+dvips+ps2eps(+epstopdf)
\documentclass[dvips,10pt]{article}
\usepackage[dvipsnames]{pstricks}
\usepackage{pst-layout}
\usepackage{mathptmx}
\font\lfA=ptmr at 9.5pt %
\font\lfB=ptmr at 9pt %
\font\lfC=ptmr at 8.5pt %
\font\lfD=ptmr at 7.5pt %
% With 10pt as \normalsize, \small is 9pt
%\scriptsize is 7pt. The above provide more choices
```

```
\pagestyle{empty}
\setlength\parindent{0pt}
\begin{document}
% Remember that C-1, C-2, R0 are predefined---do not change
\def\AtLayoutStart{\scriptsize }
\NumPoints=5 % ignore higher indices
\PointName(1)={TR}\PointPos(1)={PSPtr}
% The next lines define nodes R1..R7 to specify rows
\PointName(2)={R}\PointPos(2)={0,1.03}%
\PointInc(2)={0,-.138in}\PointSeries(2)={1;7}
\PointName(3)={C0}\PointPos(3)={.19,0}%start column for name etc
\PointName(4)={C1}\PointPos(4)={3.31,0}%end column for name etc
\PointName(5)={C2}\PointPos(5)={2.22,0}%start column for email etc
\def\MaxR{8}\def\MaxC{2}
\NumGraphics=1
\Graphic(1)={mygraphic}%requires mygraphic.eps
\GraphicPos(1)={.453,1.49}
\GraphicRefPt(1)={c}
\GraphicOpts(1)={width=43pt,height=44pt}
\NumFragments=9 %
\Frag(1)={\normalsize YOUR OWN NAME}\FragRefPt(1)={B}%Baseline, center
\FragPos(1)={R1C-2}
\Frag(2)={\lfB Your Position}\FragPos(2)={R2C-2}\FragRefPt(2)={B}
\Frag(3)={\lfB Your Department}\FragPos(3)={R3C-2}\FragRefPt(3)={B}
\Frag(4)={YOUR INSTITUTION}\FragPos(4)={R6C0}\FragRefPt(4)={Bl}
\Frag(5)={\lfD Your Address}\FragPos(5)={R7C0}\FragRefPt(5)={Bl}
\Frag(6)={\lfD Phone:} \FragPos(6)={R6C2}\FragRefPt(6)={Bl}
\Frag(7)={\lfD email:}\FragPos(7)={R7C2}\FragRefPt(7)={Bl}
\Frag(8)={\lfD Your phone}\FragPos(8)={R6C1}\FragRefPt(8)={Br}
\Frag(9)={\lfD Your email}\FragPos(9)={R7C1}\FragRefPt(9)={Br}
\psset{unit=1in,picwidth=3.5in,picheight=2in,showfullcanvas}
\begin{pslayout}
\end{pslayout}
\end{document}
```

Once this graphic is created and saved in `.eps` format, say as `mycard.eps`, you may print out ten cards per page using the following.

```
%&latex
%latex+dvips+epstopdf
\documentclass[dvips,10pt]{article}
\usepackage{graphicx}
\usepackage{pstricks-add}
\usepackage[vmargin=.5in,hmargin=.75in,nohead]{geometry}
\usepackage{mathptmx}
```

```
\pagestyle{empty}
\setlength\parindent{0pt}
\begin{document}
\psset{xunit=3.5in,yunit=2in}
\begin{pspicture}(2,5)
\multido{\iA=0+1}{5}{%
\multido{\iB=0+1}{2}{\rput[bl](\iB,\iA){\includegraphics{mycard}}}}
\end{pspicture}
\end{document}
```