Absolute positioning with textpos

Norman Gray

Abstract

I describe the textpos package, which allows you to place blocks of text at arbitrary positions on the page. I give an overview of its functionality, and discuss a few points of TEXnical interest.

1 Introduction

TEX has many wonderful features, but the one most often, and most forcefully, advanced to potential converts is that they need no longer fret about layout. 'Concentrate on the text; the layout is not your concern', we say, 'Produce your deathless text in a golden stream and let LATEX handle the minutiae of fonts and linebreaking and spacing'. This is true, but in those few cases where the layout actually is part of the point — this block of text must go *here*, the logo *just there* — authors find themselves embroiled in an unseemly struggle with an application which suddenly seems officious rather than helpful, resorting to increasingly shrill demands ('`\newline` damnit!'), and more or less desperate hackery. This is where textpos can help.

In 1998 I was involved in just such a struggle with LATEX, laying out text on an A0 sheet to produce a conference poster. The package I produced I released as textpos, and it has become a common way to achieve this sort of positioning control. The package is available from CTAN, at `macros/latex/contrib/textpos`, and also from the textpos home page, `http://www.astro.gla.ac.uk/users/norman/distrib/latex/`.

In this article I'll give a quick overview of the functionality of textpos. Then I'll make a few observations on the way that textpos is implemented, and on its relationship with the everyshi package.

2 Using textpos

The textpos package is not a complicated one, since in outline it consists of only a single environment, plus a starred variant and a few configuration parameters. The manual distributed with the package [1] gives the details; I need only outline the principles here.

The package defines an environment `textblock` which contains the text (or other material) which is to be placed in a block, and takes parameters which specify the width of the block and its position relative to a reference point. The syntax is as follows:

```
\begin{textblock}{⟨width⟩}[⟨hx⟩,⟨hy⟩](⟨x⟩,⟨y⟩)
...
\end{textblock}
```

The ⟨*width*⟩ gives the width the block is to have, and the ⟨x⟩ and ⟨y⟩ parameters give the position of the block's 'handle' relative to the 'reference point'. The handle is by default the top-left corner of the block, but may be moved using the optional argument (in square brackets as usual); the reference point I will return to in a moment. Notice that the positioning arguments for the `textblock` environment — the coordinates (⟨x⟩,⟨y⟩) — are in parentheses rather than curly braces, in slight imitation of the `picture` environment.

The salient features of this syntax and its effects are illustrated in figure 1. In this illustration, the rules around the boxes are there because I included the `[showboxes]` option when I loaded the textpos package at the beginning of this article — this is

```
In my beginning
\begin{textblock}{2.5}(0.5,2)
\raggedright
Work is of two kinds: first, altering
the position of matter at or near the
earth's surface relatively to other such
matter; second, telling other people
to do so.
\end{textblock}
\begin{textblock}{2}[0.5,0.5](4,2)
\raggedleft
The first kind is unpleasant and ill paid;
the second is pleasant and highly paid
\emph{[Russell]}.
\end{textblock}
is my end.
```

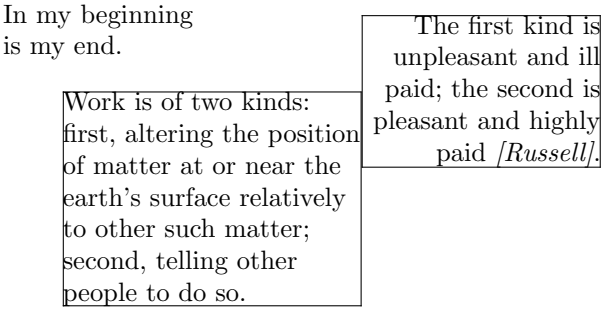

Figure 1: An example of using the `textblock` environment.

useful when laying out the boxes, but it is off by default.

In a `textblock` environment, you specify the widths of the textblocks, but not their heights, which are as large as they have to be to enclose their content.

Here, I have shown the content as simple text, but the contents can be anything which can go into a `\vbox`.

The positioning parameters specify the position of a notional handle, which by default is located at the top left corner of the block. However, you can move this handle to any part of the block by using the arguments `[`⟨*hx*⟩`,`⟨*hy*⟩`]`, as I did in the second block in figure 1. The coordinates of the handle are given as multiples of the horizontal and (final) vertical size of the block, so that `[0,0]` is the top left (the default), `[1,0]` is the top right, and `[0.5,0.5]` is the centre. The fractions don't have to be restricted to the range $0 \leq f \leq 1$.

Each of the environments takes up zero space, so that the 'reference point' — the point relative to which the boxes are positioned — is the same for each of the environments, as long as there is no material between them (or more precisely, no material with a vertical extent greater than 0 pt). This is why the text 'is my end' appears close to the text in the top left; observe, however, that they are not immediately adjacent, and the presence of the `textblock` environments has inserted a single paragraph break. If a `textblock` appears when TEX is typesetting the text of a paragraph (that is, it is in horizontal mode), then the environment ends the paragraph, as if you had typed `\par` at that point, or inserted a blank line. While I'm talking about spacing, note that there is nothing inhibiting you from (or defending you against!) overlapping the text of the boxes, so that there is necessarily an element of visual layout involved in using the environment.

2.1 Lengths

The widths and positions which are arguments to the `textblock` environment are given in units of `\TPHorizModule` for the horizontal lengths, and `\TPVertModule` for the vertical ones. These are TEX ⟨*dimen*⟩s, so you can set their values in the usual way, with `\setlength`. You usually will want to do this, since the default values, of $1/16$ of the `\paperwidth` and `\paperheight` respectively, are not likely to be particularly useful. For example, the figures above were preceded by

```
\setlength{\TPHorizModule}{\columnwidth}
\divide\TPHorizModule by 5
\setlength{\TPVertModule}{\baselineskip}
```

to adapt the positioning to the typographical environment. Alternatively, you could use the calc package and write this more straightforwardly as

```
\setlength{\TPHorizModule}{\columnwidth/5}
```

textpos is compatible with calc, thanks to code from Rolf Niepraschk. You can also use these dimensions directly, in `\hspace {2 \TPHorizModule}` for example, if that helps to give your document more consistency.

Using these modules makes your arrangement of blocks easily rescalable, and it helps fit the blocks neatly into a larger structure; however, they can also help your layout look better. Although you can give fractional positions (as I illustrated in figure 1), your layout will tend to look more coherent if you pick a suitable module and try to restrict yourself to integer multiples of it. This can make the difference between a layout which looks busy and cluttered, and one which is elegantly restrained.

If your layout requirements are more specific than this, then you may want to use the starred variant of the `textblock` environment. This is just like the unstarred version, except that the block width and positioning parameters are given as absolute lengths, in any TEX units such as `pt` or `em`, rather than as multiples of the horizontal and vertical modules (the optional arguments remain relative to the size of the final block).

2.2 Absolute positioning

As I explained above, the position arguments are by default relative to a reference point which is identified as the 'current position' on the page, which is unchanged by the presence of the textblock. For some applications, such as laying out a conference poster, it is most useful if the reference point can be guaranteed to be in a particular location, and guaranteed not to move, and it is for this reason that textpos also has an 'absolute' mode.

You put textpos in this mode with

```
\usepackage[absolute]{textpos}
```

after which all textblocks are positioned relative to a single origin on the page, irrespective of any material that separates them. This origin is by default located at the top left corner of the paper (that is, 25.4 mm (ahem!) leftwards and upwards of TEX's usual nominal reference point), but you can adjust it with the `\textblockorigin`$\{\langle x\rangle\}\{\langle y\rangle\}$ command, which takes two arguments, giving the horizontal and vertical position of the origin, relative to the top left corner of the paper. The dimensions $\langle x\rangle$ and $\langle y\rangle$ must have units; they are not multiples of any module.

The command

`\TPGrid[`$\langle bh\rangle$`,`$\langle bv\rangle$`]`$\{\langle nh\rangle\}\{\langle nv\rangle\}$

is an alternative way of setting the `\TPHorizModule` and `\TPVertModule` lengths, particularly useful in absolute mode. This firstly sets the modules so that

$$\langle nh\rangle \times \texttt{\textbackslash TPHorizModule} + 2\langle bh\rangle = \texttt{\textbackslash paperwidth}$$

$$\langle nv\rangle \times \texttt{\textbackslash TPVertModule} + 2\langle bv\rangle = \texttt{\textbackslash paperheight}$$

and secondly calls `\textblockorigin`$\{\langle bh\rangle\}\{\langle bv\rangle\}$, so that the modules form a $\langle nh\rangle \times \langle nv\rangle$ grid on the paper, with a border $\langle bh\rangle$ wide and $\langle bv\rangle$ deep around it. If the optional border argument is absent, it defaults to `[0pt,0pt]`. The `\textblockorigin` command is available only in 'absolute' mode, but the `\TPGrid` command is available in relative mode also.

For other hints on formatting posters, see [2].

There are a few other textpos details, to do with colouring in boxes, and overlaying them or not; for those, see the textpos documentation.

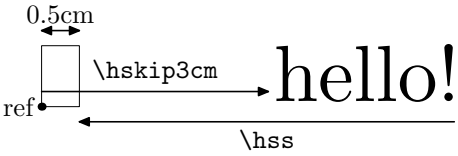

Figure 2: Positioning text in and out of boxes.

3 Implementation

The textpos package is not a complicated one, and is at heart a wrapper round two simple but powerful techniques, namely positioning with glue and boxes, and hooking into the output routine using everyshi. I will describe these here, in case they are of wider TEXnical interest.

3.1 The basics of positioning

At one level, a TEX page is no more than a sequence of boxes, glue, and a few more exotic things that need not concern us here. Each box has a height, a depth, a width, and at this level nothing else. The material inside these boxes, consisting of characters, rules, and other boxes, is what we ultimately wish to see on the page, but there is no requirement for this material to lie strictly within the boundaries of the box it is associated with; furthermore, glue can be negative in extent as well as positive, and these two observations together give us a technique for positioning things.

Consider the TEX box

```
\hbox to 0.5cm{\hskip 3cm hello!\hss}
```

which is illustrated in figure 2. That creates an `\hbox` which is exactly 0.5 cm wide, and puts into it a skip and some text which are together substantially larger than the box. This would create an overfull `\hbox` were it not for the 'infinitely shrinkable glue', the `\hss`, at the end. This inserts whatever glue is required to make the whole construction have zero badness. The end result is that we have placed the text 'hello!' at a point 3 cm to the right of the reference point of the `\hbox` which, as far as TEX is concerned, is only 0.5 cm wide. We can do exactly the same thing with `\vbox`es and `\vss` glue and, putting these together, get the result in figure 3.

That — plus a little bit of syntactic sugar[1] and some spacing magic — is textpos.

3.2 Absolute positioning and `\shipout`

Well, *almost* all there is to it. While the technique in the last section is enough for the default 'relative'

[1] The fact that `textpos.sty` has ended up over 250 lines long, shows that sugar can be very fattening indeed.

```
I am here
\vbox to 0pt{%
 \vskip 1cm
 \hbox to 0pt{\hskip 2cm In my beginning.\hss}%
 \vss}Or there, or elsewhere.
```

I am here Or there, or elsewhere.

In my beginning.

Figure 3: Boxes in boxes: how textpos works.

mode of textpos, it does not address the problem of pinning the reference point for absolute mode to a fixed position on the page. The most elegant way to do this was suggested by Olaf Maibaum, using Martin Schröder's everyshi package (also at CTAN, of course).

The 'shipout' is the very final stage of TEX's handling of a page. When it has found an optimal page break, TEX puts the page contents into the special box register \box255 and calls the \output routine. That routine's job is to do the final assembly of a page, handling footnotes, marginal notes, and the rest, and when it is finished, it wraps it all up into a final box which it passes to the TEX primitive \shipout. The everyshi package gives you a last chance to tinker with the output, by letting you register a sequence of commands which will be invoked at precisely this point, with the contents of \shipout's argument available in \box255 (though this is almost certainly not the same \box255 which was originally prepared for the \output routine). It is the content of this box *after* you've made any adjustments that is passed to the primitive \shipout. This is a tremendously powerful mechanism.

In absolute mode, textpos converts each text block into a zero-height \vbox as usual (in fact, into the temporary \box0), but instead of contributing them to the current page, it accumulates them in a holding box:

```
\global\setbox\TP@holdbox=\vbox{%
  \box0
  \unvbox\TP@holdbox}
```

However, in this mode textpos has also registered some commands with the everyshi hook:

```
\EveryShipout{%
  \global\setbox255=\vbox{%
    \unvbox\TP@holdbox
    \unvbox255}}
```

Thus, whenever the output routine is called, either because a page has filled up or because the input file has come to an end, it constructs its final box and calls \shipout. At this last moment this fragment of code prepends the textpos hold box to the everyshi \box255 and lets this enlarged box be the one shipped out.

3.3 So ...

textpos has pulled off the rather un-TEX-like trick of supporting the *non*-automatic layout of text, and it has done so without outrageous trickery, by simply exploiting the core functionality of TEX's page-layout algorithm — constrained gluing together of boxes of given sizes — in an unexpected way. As a result, understanding textpos requires, or prompts, an understanding of that algorithm, with its parameters \baselineskip, \prevdepth and friends, and this, together with the use of other common techniques such as TEX arithmetic and token lists, means that it manages simultaneously to solve a non-trivial problem usefully, and to be instructive.

And that's a good position to be in.

References

[1] Norman Gray. *textpos User Manual, version 1.4*, 2003. Distributed with the textpos package.

[2] Norman Gray. Using LATEX to produce conference posters, 2003. http://www.astro.gla.ac.uk/users/norman/docs/posters/.

◇ Norman Gray
Department of Physics and Astronomy
University of Glasgow
Glasgow, UK
norman@astro.gla.ac.uk
http://www.astro.gla.ac.uk/users/norman/