## Glisterings

Peter Wilson

Catching fire, taking hold
All that glisters leaves you cold
No-one is near, no-one will hear
Your changeling song take shape
In Shadowtime.

*Shadowtime*, Siouxsie and the Banshees

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

Twixt the optimist and pessimist
The difference is droll:
The optimist sees the doughnut
But the pessimist sees the hole.

*Optimist and Pessimist*,
McLandburgh Wilson

## 1 Cutout windows

While winnowing my shelves and piles of books, journals, magazines, paper, etc., in anticipation of a move from the US to the UK I came across a *TUGboat* article by Alan Hoenig [2] in which he provides TEX code for creating an open window in the middle of a paragraph. An example of a paragraph with a cutout is in Figure 1. This was produced by:

```
\input{cutsty.tex}
\window{2}{0.4\textwidth}{0.25\textwidth}{5}
    This paragraph is set within the ...
...in a minipage in a \TUB\ \texttt{figure*}).
\endwindow
```

I tried out the code as given but found that it needed a tweak here and there to improve the spacing. Here is my version of Alan's code for rectangular cutouts, which can be used in both TEXed and LATEXed documents.[1] Most of my changes to the code are changes of name and argument specification to make it acceptable to both TEX and LATEX.

```
% cutsty.tex  Based on Alan Hoenig,
%  'TeX Does Windows --- The Conclusion',
%  TUGboat 8:2, pp.211-215, 1987
```

First some counts, lengths, and boxes are needed (I have used `cut` as the start of each of these names to try and avoid clashes with other code):

```
\newcount\cutlines \newcount\cuttoplines
\newdimen\cutlftside \newdimen\cutrtside
\newtoks\cuta
\newcount\cutn
\newbox\cutrawtext \newbox\cutholder
\newbox\cutwindow \newbox\cutfinaltext
\newbox\cutaslice \newbox\cutbslice
\newdimen\cuttopheight
\newdimen\cutilgvs % glue or shift
```

[1] Alan also gave code for creating arbitrary shaped holes.

The main user commands are `\window` and the accompanying `\endwindow`. The first of these takes four arguments as:
`\window{`⟨*top-lines*⟩`}{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*cut-lines*⟩`}`
where ⟨*top-lines*⟩ is the number of lines before the window cutout, ⟨*left*⟩ is the width of the text at the left of the window and ⟨*right*⟩ the width of the text at the right, and ⟨*cut-lines*⟩ is the number of lines used for the window (i.e., the height of the window). The macro gets a `\parshape` for the forthcoming text, gets and applies any vertical shift, opens a box for the text and then applies the `\parshape`.

```
\def\window#1#2#3#4{%
  \cuttoplines=#1\relax
  \cutlines=#4\relax
  \cutlftside=#2\relax
  \cutrtside=#3\relax
  \cuta={}%
  % calculate the \parshape spec
  \parshapespec
  % reset the these arguments
  \cuttoplines=#1\relax
  \cutlines=#4\relax
  % calculate and apply any vertical shift
  \cutshift \vskip-\cutilgvs
  % start a box for collecting the text
  \setbox\cutrawtext=\vbox\bgroup
  \parshape=\cutn \the\cuta}
```

The text, in the form of a single paragraph with a constant `\baselineskip` is put between the two `\...window` commands; in the case of LATEX you can, but don't need to, use a `window` environment instead.

The general scheme is to use a specially shaped paragraph which effectively splits the text into three sets of lines; those before the cutout; those that will form the cutout; and the rest. The lines forming the cutout are short while the others are full length. An example is shown in Figure 2. The final output is assembled from the top set of lines, the cutout lines combined in pairs, and the remainder. The final form of a paragraph with a cutout is shown in Figure 3.

```
\def\endwindow{%
  \egroup    % end \box\cutrawtex
  \parshape=0 % reset parshape
  \computeilg % find ILG using current font
  \setbox\cutfinaltext=
    \vsplit\cutrawtext
      to\cuttoplines\baselineskip
```

This paragraph is set within the `window` environment. There are limitations on the `window` arguments and text. There must be at least one line of text above the window and if the number of lines specified for the opening exceeds the available lines then the text after the `window` environment will be moved down by an amount corresponding to the excess. A window will not extend into a second paragraph. The environment is effectively a box and will not break across a page boundary. There should be enough space at the left and right of the window for a few words on each side (don't try to make either of these zero in an attempt to have a window opening to the margin). There is usually not enough width to put a significant window into a column on a two-column page (this has been set in a minipage in a *TUGboat* `figure*`).

**Figure 1**: A generated window

If you have to have a cutout in a narrow column keep the words short. Use one or two or maybe one or more extra letters so that they may fit into the available area without too much odd spacing. If the words are hyphenatable this will help a lot as then a long one may be cut into two short bits.

**Figure 2**: Split window lines

```
  \cuttopheight=\cutlines\baselineskip
  \cuttopheight=2\cuttopheight
  \setbox\cutholder=
    \vsplit\cutrawtext
      to\cuttopheight
  % \cutholder contains the narrowed text
  %  for window sides. Slice up \cutholder
  %  into \cutwindow
  \decompose{\cutholder}{\cutwindow}
  \setbox\cutfinaltext=\vbox{%
    \unvbox\cutfinaltext\vskip\cutilgvs
      \unvbox\cutwindow%
      \vskip-\cutilgvs\unvbox\cutrawtext}%
  \box\cutfinaltext}
```

A `\parshape` is used to specify quite general paragraph shapes [3, Ch. 14] or [1, Ch. 18]. Its $2n+1$ parameters specify the indentation and length of the first $n$ lines in the following paragraph which must start immediately (no empty line after the parameters). The first parameter is $n$ followed by $n$ pairs of indentation and line length values. In general:
`\parshape` $n\ i_1\ l_1\ i_2\ l_2\ \ldots\ i_n\ l_n$
If there are more than $n$ lines then the specification for the last line ($i_n\ l_n$) is used for the rest of the lines in the paragraph.

`\parshapespec` calculates the `\parshape` parameters to generate a paragraph with ⟨*top-lines*⟩

If you have to have a cutout in a narrow column keep the words short. Use one or two or maybe one or more extra letters so that they may fit into the available area without too much odd spacing. If the words are hyphenatable this will help a lot as then a long one may be cut into two short bits.

**Figure 3**: Assembled window lines

full lines followed by ⟨*cut-lines*⟩ of length ⟨*left*⟩ alternating with ⟨*cut-lines*⟩ of length ⟨*right*⟩.

```
\def\parshapespec{%
  \cutn=\cutlines \multiply \cutn by 2
    \advance\cutn by \cuttoplines
    \advance\cutn by 1\relax
  \loop
    \cuta=\expandafter{\the\cuta 0pt \hsize}
    \advance\cuttoplines -1\relax
    \ifnum\cuttoplines>0\repeat
  \loop
    \cuta=\expandafter{\the\cuta
          0pt \cutlftside 0pt \cutrtside}%
    \advance\cutlines -1\relax
  \ifnum\cutlines>0\repeat
  \cuta=\expandafter{\the\cuta 0pt \hsize}}
```

An example paragraph at this stage of the process is in Figure 2.

The `\decompose{`⟨*narrow*⟩`}{`⟨*split*⟩`}` command takes a box ⟨*narrow*⟩ and for each pair of lines puts the first at the left and the second at the right of the box `{`⟨*split*⟩`}`. That is, it converts pairs of lines into single lines with text at the left and the right with a space between.

```
\def\decompose#1#2{%
  % loop over the windowed lines
  \loop\advance\cutlines -1
   % get a pair of lines
  \setbox\cutaslice=\vsplit#1 to\baselineskip
  \setbox\cutbslice=\vsplit#1 to\baselineskip
  % split into the two sides
  \prune{\cutaslice}{\cutlftside}
  \prune{\cutbslice}{\cutrtside}%
  % assemble into one line
  \setbox#2=\vbox{\unvbox#2\hbox
  to\hsize{\box\cutaslice\hfil\box\cutbslice}}%
  \ifnum\cutlines>0\repeat}
```

For the example in Figure 2 the `\decompose` macro converts the 6 narrow lines into the 3 cutout lines shown in Figure 3.

`\prune{`⟨*vbox*⟩`}{`⟨*width*⟩`}` is used to prune the glue that TEX puts at the end of a short `\parshape` line. It takes a `\vbox` containing a single `\hbox`, `\unvbox`es it, cancels the `\lastskip` and puts it in a box of ⟨*width*⟩ wide; a `\strut` is needed to keep the spacing consistent.

```
\def\prune#1#2{%
  \unvbox#1
  \setbox#1=\lastbox % \box#1 is now an \hbox
  \setbox#1=\hbox to#2{\strut\unhbox#1\unskip}}
```

`\cutshift` calculates the amount that the windowed paragraph must be raised, which is half a `\baselineskip` for each windowed line. (This is my addition).

```
\def\cutshift{%
  \cutilgvs=\cutlines\baselineskip
  \cutilgvs=0.5\cutilgvs}
```

`\computeilg` computes the interline glue in the windowed paragraph. This is the last macro so finish the file with an `\endinput`.

```
\def\computeilg{%
  \cutilgvs=\baselineskip
  \setbox0=\hbox{(}
    \advance\cutilgvs-\ht0
    \advance\cutilgvs-\dp0}
\endinput
```

Artwork or text may be placed in the cutout. How to do that is a very different problem and one that I am not intending to address here, but zero-sized pictures and headers or footers come to mind [4]. Perhaps solutions will have been published by the time this article appears.

?

Since the preceding was first written, the cutwin package [5] has appeared which lets you create variously shaped cutouts and place things in the resulting window.

## References

[1] Victor Eijkhout. *TEX by Topic, A TEXnician's Reference.* Addison-Wesley, 1991. ISBN 0-201-56882-9. Available at `http://www.eijkhout.net/tbt/`.

[2] Alan Hoenig. TEX does windows — the conclusion. *TUGboat*, 8(2):211–215, 1987.

[3] Donald E. Knuth. *The TEXbook.* Addison-Wesley, 1984. ISBN 0-201-13448-9.

[4] Peter Wilson. Glisterings: Ornaments. *TUGboat*, 32(2):202–205, 2011.

[5] Peter Wilson and Alan Hoenig. Making cutouts in paragraphs, 2010. Available on CTAN in `macros/latex/contrib/cutwin`.

◇ Peter Wilson
20 Sovereign Close
Kenilworth, CV8 1SQ
UK
`herries dot press (at) earthlink dot net`