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.

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 \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.\footnote{Alan also gave code for creating arbitrary shaped holes.} 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,\footnote{‘\TeX Does Windows --- The Conclusion’,\% TUGboat 8:2, pp.211-215, 1987}
\% ‘\TeX Does Windows --- The Conclusion’,\%
\newcount\cutlines \newcount\cuttoplines
\newdimen\cutlftside \newdimen\cutrtside
\newtoks\cuta \newcount\cutn
\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}
\def\endwindow{\%
\egroup \%
\parshape=0 \%
\computeilg \% find ILG using current font
\setbox\cutfinaltext=vsplit\cutrawtext
to\cuttoplines\baselineskip

The text, in the form of a single paragraph with a constant \baselineskip is put between the two \window\ \texttt{...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.

\% cutfinaltext \% end box\cutrawtext
\parshape=0 % reset parshape
\computeilg % find ILG using current font
\setbox\cutfinaltext=vsplit\cutrawtext
to\cuttoplines\baselineskip

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

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{Split window lines}
\end{figure}

A \texttt{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:

\texttt{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.

\texttt{parshapespec} calculates the \texttt{parshape} parameters to generate a paragraph with \texttt{(top-lines)}

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

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

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

\begin{verbatim}
\def\decompose\{\{narrow\}\}{\{split\}}
\end{verbatim}

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.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{Assembled window lines}
\end{figure}

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

Peter Wilson
For the example in Figure 2 the \texttt{decompose} macro converts the 6 narrow lines into the 3 cutout lines shown in Figure 3.

\texttt{prune\{vbox\}\{(width)\}} is used to prune the glue that \TeX{} puts at the end of a short \texttt{parshape} line. It takes a \texttt{vbox} containing a single \texttt{hbox}, \texttt{unvboxes} it, cancels the \texttt{lastskip} and puts it in a box of \texttt{(width)} wide; a \texttt{strut} is needed to keep the spacing consistent.

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

\texttt{computeilg} computes the interline glue in the windowed paragraph. This is the last macro so finish the file with an \texttt{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 \textsuperscript{4}. Perhaps solutions will have been published by the time this article appears.

Since the preceding was first written, the \texttt{cutwin} package \textsuperscript{5} has appeared which lets you create variously shaped cutouts and place things in the resulting window.

\textbf{References}


\begin{flushright}
\textsuperscript{⋄} Peter Wilson \\
20 Sovereign Close \\
Kenilworth, CV8 1SQ \\
UK \\
herrys dot press (at) \\
earthlink dot net
\end{flushright}