\TeX\ Does Windows — The Conclusion

Alan Hoenig

In the last issue, readers were invited to think about creating \TeX\ macros to “do windows”—have \TeX\ leave rectangular cutouts horizontally centered within paragraphs and in which you can insert artwork, or whatever. The solution I discuss in this issue differs significantly from the one I had in mind at the time I threw down the gauntlet last time. The window macros have become leaner and more robust. (I am grateful to my colleague Mitch Pfeffer who supplied me with one crucial idea, and to Barbara Beeton for a valuable insight. Barbara also helped keep me honest.) I present the macros below, as well as two extensions, which allow \TeX\ to set rectangular cutouts which aren’t horizontally centered, and which force \TeX\ to set cutouts of arbitrary shape. I do make several limiting assumptions: the cutout fits entirely within a single paragraph, and the \baselineskip\ remains constant within that paragraph. I believe you can modify these macros with little additional work, however. There is one known bug, which I was unable to fix in time to meet the submission date. When the ratio of baselineskip to design font size reaches decreases to a certain critical value, the cutout is not properly formed. You’ll be okay if you keep the baselineskip at least 2 points greater than the design size.

**Using the Macros**

Here’s how to use the macros. First, some anatomy. The several lines of material above the window we’ll call the lintel. The material below it is the sill, and the text to its right and left form its sides. (Of course, some of these regions may be empty depending on your placement of the window.)

We’ll talk about the macro that generates a horizontally-centered window first. Before you invoke the window-making macro, place the code which reserves the registers that the macros use. These definitions form figure 1.

\begin{verbatim}
\newcount\l \newcount\d \newdimen\lftside \newdimen\rtside \newtoks\a
\newbox\rawtext \newbox\holder \newbox\window \newcount\n
\newbox\finaltext \newbox\aslice \newbox\bslice
\newdimen\topheight
\newdimen\ilg \% InterLine Glue
\end{verbatim}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure1}
\caption{The boxes, counts, dimens, and so on you need for the window-making macro.}
\end{figure}

Next, place the actual macro definitions in your document file. There are a slew of such macros—you’ll need them all. They appear in figure 2.

\begin{verbatim}
\def\openwindow{\down#1\in#2\for\linesi%
\hfill #1\ is \ an integer---no. of lines down from par top
\hfill #2\ is a dimension---amount from left where window begins
\hfill #3\ is an integer---no. of lines for which window opening
\hfill persists.
\d=#1 \1=#3 \lftside=#2 \rtside=\lftside \a=()
\createparshapeshpec
\d=#1 \l=#3 \% reset these
\setbox\rawtext=vbox\bgroup
\parshape=n \the\a }
\hfill \%
\def\endwindowtext{%
\egroup \parshape=0 \% reset parshape; end \box\rawtext
\compute\ilg \% find ILG using current font.
\setbox\finaltext=vsplit\rawtext to\d\baselineskip
\topheight=\baselineskip \multiply\topheight by\l
\multiply \topheight by 2
\setbox\holder=vsplit\rawtext to\topheight
\% \holder contains the narrowed text for window sides
\decompose\holder to\window \% slice up \holder
\setbox\finaltext=vbox{unvbox\finaltext vskip\ilg unvbox\window%
\end{verbatim}
Apart from identifying the window text, there are 3 parameters you need to determine: The number of lines down for the the top of the paragraph (that is, the thickness of the lintel), the amount in from the left (the width of the sides), and the number of lines for which the window persists (the thickness of the sides). If $w$ is the width of the side (the value of parameter $#2$ in the openwindow macro), then the width of the window is $\text{hsize} - 2w$. You see in figure 4 the exact way in which I created the window in this paragraph.

\openwindow\down \in 15pc \for 2\lines

Apart from identifying ...

...window in \{sl this\} this paragraph.
\endwindowtext

FIGURE 3. Opening up a window in text.

Horizontally Centered Windows

Here is the basic idea behind these macros. We seek to use the command \parshape to create an odd-shaped paragraph consisting of a top portion identical to the lintel, a bottom portion identical to the sill, and a lengthy and narrow middle portion with the width of the side text. Then, take this typeset text, and slice it like a roast beef. These “slices” will contain lines of text in \vboxes which we rearrange to get
the text we want, cutout and all. In figure 4, you see the intermediate position of some text before and after this rearrangement.

Only a few macros require any special comment. Macro \texttt{\textbackslash createparshapespec} computes the value \( n \) that the \texttt{\textbackslash parshape} command looks for. Next, it builds up a token \texttt{\textbackslash a} which contains the \( 2n \) line lengths and indentations. (I take all the indentations to be \texttt{Opt}, but varying this can add to your palette of special effects.) Some of this token manipulation is a tad tricky. You construct tokens by enclosing the token list in braces, but if you don’t prefix this with an \texttt{\textbackslash expandafter}, \TeX\ will construct \texttt{\textbackslash a} out of the component names, rather than out of their meanings. The last line is necessary because of the way \texttt{\textbackslash parshape} works. If the paragraph is sufficiently long, \TeX\ uses the final line length for the remainder of the text. If we set the final line length to be \texttt{\textbackslash hsize}, then this is what we need to set the sill text.

Note that these macros use \texttt{\textbackslash struts} to help maintain proper vertical line spacing. If you use type at other than the usual 10 point design size, \texttt{\textbackslash make sure to redefine the \textbackslash strut to reflect this change}.

\begin{quote}
London. Michaelmas Term lately over, and the Lord Chancellor sitting in Lincoln’s Inn Hall. Implacable November weather. As much mud in the streets, as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill. Smoke lowering down from chimney-pots, making a soft black drizzle, with flakes of soot in it as big as full-grown snow-flakes—gone into mourning, one might imagine, for the death of the sun. Dogs, undistinguishable in mire. Horses, scarcely better; splashed to their very blinkers. Foot passengers, jostling one another’s umbrellas, in a general infection of ill-temper, and losing their foot-hold at street-corners, where tens of thousands of other foot passengers have been slipping and sliding since the day broke (if this day ever broke), adding new deposits to the crust upon crust of mud, sticking at those points tenaciously to the pavement, and accumulating at compound interest.
\end{quote}

\begin{quote}
London. Michaelmas Term lately over, and the Lord Chancellor sitting in Lincoln’s Inn Hall. Implacable November weather. As much mud in the streets, as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill. Smoke lowering down from chimney-pots, making a soft black drizzle, with flakes of soot in it as big as full-grown snow-flakes—gone into mourning, one might imagine, for the death of the sun. Dogs, undistinguishable in mire. Horses, scarcely better; splashed to their very blinkers. Foot passengers, jostling one another’s umbrellas, in a general infection of ill-temper, and losing their foot-hold at street-corners, where tens of thousands of other foot passengers have been slipping and sliding since the day broke (if this day ever broke), adding new deposits to the crust upon crust of mud, sticking at those points tenaciously to the pavement, and accumulating at compound interest.
\end{quote}

\begin{figure}
\centering
\includegraphics{figure4}
\caption{The window macro generates a funny-looking paragraph (top) using the \texttt{parshape} command, which it then rearranges to form the window (bottom).}
\end{figure}

The actual “deli slicing” is done by the \texttt{\textbackslash decompose} macro, and the slicing mechanism is done via the \texttt{\textbackslash vsplit} primitive. Decomposition also calls for the \texttt{\textbackslash prunings} of each slice—removing the glue \TeX\ places on the right of a short, \texttt{\textbackslash parshape} line.

\section*{On Beyond Centering: Off-Center Windows}

All the hard work in cutting out windows is already present in these macros. It’s almost trivial to generalize to the case where a rectangular window is to be off-center. The only real change is to the definition of \texttt{\textbackslash openwindow} to allow for another parameter—the width of the right side of the window. Here, in figure 5, is the way I’ve redefined the beginning of \texttt{\textbackslash openwindow}.

\begin{verbatim}
def\openwindow\down#1 \in #2 \fromright #3 \for #4\lines{%
  \d=#1 \l=#4 \lftside=#2 \a={} \rtside=#3
\end{verbatim}

\begin{figure}
\centering
\includegraphics{figure5}
\caption{The opening for the openwindow macro to account for windows that can be off-centered horizontally. Parameter 3 is now to be the distance from the right margin.}
\end{figure}
On Beyond Rectangles: Arbitrary Shapes

The most interesting part of writing these macros was their extension to windows of arbitrary shape. Users specify the "shapespec" for their cutout by creating a list of the right and left line lengths, each of which is separated by a double backslash \; this list is then passed to the macro as a single parameter. Changes to the macros to handle this shape spec are localized to a few places. As before, the beginning of the \openwindow statement needs modifying to accept a different parameter. At the risk of appearing repetitious, all the listings for all the macros in The \openwindow macro (of Appendix D in the \TeXbook) to chop off the individual line lengths from the shape spec. The \shapespec itself is of the form \[ \ll_{1} \rr_{1} \ll_{2} \rr_{2} \ldots \ll_{n} \rr_{n} \] where $l_{i}$ should be the length of the left side, and each $r_{i}$ is the length of the right side for the $i^{th}$ line of the cutout. Note well the double backslash which both begins and ends the \shapespec. The \shapespec for this paragraph is \[ 168x 168x 154x 145x 145x 145x 138x 138x 134x 134x 144x 144x 154x \] containing the value 1.22pt.

\% \% Special registers \% \newcount\l \newcount\d \newdimen\lftside \newdimen\rtside \newtoks\a \newtoks\b \newbox\rawtext \newbox\holder \newbox\window \newcount\n \newbox\finaltext \newbox\aslice \newbox\bslice \newdimen\topheight \newdimen\ilg % InterLine Glue \newtoks\c \% \% \% \% Here are the special WINDOW MACROS. \% \long\def\openwindow\down#1\for#2\linesand#3\as\spec{\% \% #1 is an integer---no. of lines down from par top \% #2 is an integer---no. of lines for which window opening \% #3 is the \shapespec token list \d=#1 \l=#2 \def \b{#3) \def \tail{\hsize \edef \CO pt3 \createparshapespec \d=#1 \l=#2 \setbox\rawtext=vbox\bgroup \parshape=\n \the\c \b \tail } \% \% \% \% \def\endwindowtext{\% \egroup \parshape=0 \reset \parshape; end \box\rawtext \computeilg % find ILG using current font.
FIGURE 6. These are the macros you will need to generate cutouts of arbitrary text within your document.

A few final comments. Use these macros sparingly; it's difficult and mighty uncomfortable to read across a large gap in text. If you do use these macros, try not to let the width of the sides get too narrow or you'll have lots of 'overfull' messages. You may therefore want to up the \tolerance of your document. Also, the use of these macros may greatly increase the \TeX compilation time. This may be especially noticeable on microcomputers like an IBM PC. Wait a good few minutes before you decide the system has hung and it's time to re-boot.