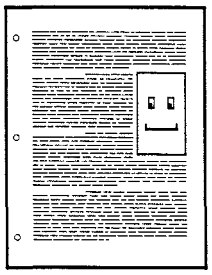
Floating Figures at the Right — and — Some Random Text for Testing

Thomas J. Reid
Texas A&M University

PREFACE: This article is a rewritten version of a note which the author sent to *TEXhax*. The techniques presented in this article represent several significant improvements over those in the earlier note. In particular, the output routine has been simplified and generalized and the `\everypar` token list is now used to control the figure insertion.



Placing a figure to the right of a paragraph of text is relatively easy in TEX. It is done by placing the figure before the start of the paragraph and using `\hangindent` and `\hangafter` to set the amount of the indent and the number of lines needed to "cover" the figure. TEX will automatically adjust the paragraph shape to fit around the figure as it has done here.

However, in practice, the figure is likely to be larger; it is very probable that one paragraph of text may not be enough to "cover" the figure. The figure above shows a sample page of "text" containing such a figure. This smaller figure begins even with the second "paragraph" and extends into the third one. While hanging indents can be used easily to set the shape for paragraph two, their use for paragraph three is more complicated: it is necessary to account for the size of paragraph two and the `\parskip` glue when calculating the number of lines to be indented.

A further complication can arise. Suppose that the desired place to begin the figure had been at the start of paragraph four. There is not sufficient space left on the page to begin the figure; the figure needs to be floated to the next page.

Approach to solution

This problem can be solved by performing a test at the start of each paragraph of text in the area where the figure is to be placed. At the start of each paragraph, one of three possibilities exists:

1. The figure has not yet been started;
2. The figure has been started and the shape of the paragraph about to begin is to be adjusted for the remaining portion of the figure; or
3. The text has proceeded past the end of the figure.

If the figure has not yet started, we check to see if there is room left for it on the current page. If there is, we start the figure and then adjust the paragraph shape. Otherwise, we defer further action on it until the start of the next paragraph. If the figure has been started but the text has not yet covered it (e.g., paragraph three in the sample page figure), we adjust the paragraph shape using the original height of the figure minus the height which has already been covered. Once the text has passed the figure, no action is needed for adjusting paragraph shapes.

Implementation of this procedure requires that the current position on the page be known. This information is available only within the output routine; to get it, we need to change the output routine to save it for us.

The modified output routine

Changes are made to the output routine to allow us to use it to query page height without disturbing its normal operation. These changes consist of adding a reference to an `\outputpretest` token list to the start of the output routine and making the execution of the original output routine conditional upon a flag set by the `\outputpretest` control sequences.

```
\newif\ifoutput
\newtoks\outputpretest
\edef\oldoutput{\the\output}
\output={\the\outputpretest
  \ifoutput \oldoutput \fi}
\outputpretest={\outputtrue}
```

(And all along you thought output routines were nasty beasts.)

Preparing the figure

The figure to be inserted is defined and placed in a box register named `\figbox`. It is important to set the dimensions of this box register to reflect the dimensions of the figure. Another dimension value is set in `\figgutter`; this represents the space to be placed between the text and the figure.

```
\newdimen\unit \unit=5pt
\def\point#1 #2 #3 {\rlap{\kern#1\unit
  \raise#2\unit \hbox{#3}}}
\newbox\figbox
\newdimen\figgutter \figgutter=1pc
```

```
\def\vr<#1,#2,#3>{\vrule height #1
  depth #2 width #3}
\setbox\figbox=\vbox to 50\unit{\hbox{%
  \point  0  0 {\vr<1.5pt,0pt,30\unit>}
  \point  0 50 {\vr<0pt,1.5pt,30\unit>}
  \point  0  0 {\vr<50\unit,0pt,1.5pt>}
  \point 30  0
    {\kern -1.5pt \vr<50\unit,0pt,1.5pt>}
  \point  5 15 {\vr<2.0pt,0pt,20\unit>}
  \point  5 15 {\vr<4\unit,0pt,2.0pt>}
  \point 25 15
    {\kern -2.0pt \vr<4\unit,0pt,2.0pt>}
  \point  7 28 {\vr<1.0pt,0pt,4\unit>}
  \point  7 35 {\vr<0pt,1.0pt,4\unit>}
  \point  7 28 {\vr<7\unit,0pt,1.0pt>}
  \point 11 28
    {\kern -1.0pt \vr<7\unit,0pt,1.0pt>}
  \point  7 28 {\vr<2.5\unit,0pt,2\unit>}
  \point 19 28 {\vr<1.0pt,0pt,4\unit>}
  \point 19 35 {\vr<0pt,1.0pt,4\unit>}
  \point 19 28 {\vr<7\unit,0pt,1.0pt>}
  \point 23 28
    {\kern -1.0pt \vr<7\unit,0pt,1.0pt>}
  \point 19 28 {\vr<2.5\unit,0pt,2\unit>}
\hss}\vss}
\wd\figbox=30\unit
```

Controlling the figure placement

In addition to the modified output routine, we define an alternate \outputpretest routine which will be used to decide whether or not to actually output anything to the DVI file.

```
\newbox\pagebox
\newdimen\pageht
\newif\iftryingfig    \tryingfigfalse
\newif\ifdoingfig     \doingfigfalse
\newif\ifpageafterfig \pageafterfigfalse
\def\dofigtest{%
  \ifnum\outputpenalty=-10001
    \setbox\pagebox=\vbox{\unvbox255}%
    \global\pageht=\ht\pagebox
    \outputfalse
    \unvbox\pagebox
  \else
    \outputtrue
    \ifdoingfig
      \global\pageafterfigtrue
    \fi
  \fi}
```

In this alternate pre-test routine, we test for a penalty value of −10,001. This special penalty value will be used in a later macro to signal our intentions to the \outputpretest routine. If this value is found, we get the height of box 255 (\unvboxing it removes the glue from the bottom of the box). This height is saved so that it can be used later and the box contents are returned to the main vertical list while \outputfalse is set to bypass execution of the original output routine.

If any other penalty value caused the output routine to be entered, the \dofigtest macro sets \outputtrue so that the normal output routine will be performed. However, a flag (\pageafterfigtrue) is set if we were "actively doing the figure" when the page break occurred. This condition occurs when the figure is placed flush with the bottom of the page.

Next, the macro which controls the figure placement is defined.

```
\newdimen\startpageht
\newdimen\htdone \htdone=0pt
\edef\oldeverypar{\the\everypar}
\everypar={\tryfig \oldeverypar}
\def\tryfig{%
  \iftryingfig % ------------ Section A
    {\everypar={\relax}\setbox0=\lastbox
      \parindent=\wd0 \parskip=0pt \par
      \penalty-10001 \leavevmode}%
    \dimen0=\vsize
    \advance\dimen0 by -\pageht
    \advance\dimen0 by -2\baselineskip
    \ifdim\dimen0>\ht\figbox
      \dimen0=0.3\baselineskip
      \vrule depth \dimen0 width 0pt
      \vadjust{\kern -\dimen0
        \vtop to \dimen0{%
          \baselineskip=\dimen0
          \vss \vbox to 1ex{%
            \hbox to \hsize{\hss
              \copy\figbox}\vss}\null}}%
      \global\tryingfigfalse
      \global\doingfigtrue
      \global\startpageht=\pageht
      \global\htdone=0pt
      \dohang
    \fi
  \else % ------------------- Section B
    \ifdoingfig
      {\everypar={\relax}\setbox0=\lastbox
        \parskip=\wd0 \parskip=0pt \par
        \penalty-10001 \leavevmode}%
      \global\htdone=\pageht
      \global\advance\htdone by
                         -\startpageht
      \ifpageafterfig
        \global\doingfigfalse
      \else
        \dimen0=\ht\figbox
        \advance\dimen0 by 0.5\baselineskip
        \ifdim\htdone<\dimen0
          \dohang
        \else
          \global\doingfigfalse
        \fi
      \fi
    \else % ---------------- Section C
      \global\outputpretest={\outputtrue}%
    \fi
  \fi}
```

Although this macro is fairly long, it is rather straightforward (with two exceptions). It is divided (as indicated by the comments) into three sections: section A which starts the figure; section B which controls the paragraph shapes after the figure has been started; and section C which is performed after the text has passed the figure. Execution of one of the three sections is determined by the settings of `\iftryingfig` and `\ifdoingfig`.

Within section A, we first invoke the output routine to get the current page height. This is a little bit tricky since we have just started the paragraph and are in horizontal mode (`\tryfig` is called from the `\everypar` token list immediately *after* entering horizontal mode and inserting the `\parindent` glue). Thus, we need to define a temporary "dummy" `\everypar` token list (to prevent endless recursion), then break out of the paragraph, signal the output routine, and restart the paragraph.

It is then a simple matter to compute the space left on the page and test to see if that is greater than the height of `\figbox`. (The extra two times `\baselineskip` is added to avoid a problem situation. For more details, see under "Problems with the insertion macros.") If there is not enough room, we will exit the macro without changing `\iftryingfig`. This will cause section A to be checked again at the next paragraph break. If there is room on the page for the figure, we output the figure and change the `\iftryingfig` and `\ifdoingfig` flags. The height of the page when the figure is saved since it will be needed later. Then, `\htdone` is preset to zero. This indicates how much of the figure has been covered by text and it is used in calculating the number of lines to be shortened. Finally, we call the `\dohang` macro to calculate and set the hanging indent values.

Inserting the figure represents another tricky situation. The commands here use the same techniques as those given for Exercise 14.28 in *The TeXbook*: We insert a strut in the current line to give it a known depth; then, with a `\vadjust`, we insert a box with zero height and a depth equal to that of the strut. Inside this `\vtop` box, we define a `\vbox` to force the figure to have a height equal to the x-height of the current font. Finally, the innermost `\hbox` causes the figure to be right aligned.

Section B also begins by calling the output routine to get the page height. We then compute the amount of the figure which has been covered. If a page break occurred since we started the figure (`\pageafterfigtrue`) or if the figure has been covered, we set `\doingfigfalse` and terminate the macro. Otherwise, call `\dohang` again to compute and set the hanging indents for the new paragraph.

Section C is quite simple: we redefine `\outputpretest` so that `\dofigtest` won't be called anymore.

Note that we redefine the `\everypar` token list when the `\tryfig` macro is defined. This will cause `\tryfig` to be invoked at the start of every paragraph following the definition. However, the initial settings of `\tryingfigfalse` and `\doingfigfalse` result in Section C being executed each time. Sections A and B won't be used until we activate them.

To activate the insertion process, we define a macro to perform the needed setup.

```
\def\rightinsert{%
  \outputpretest={\dofigtest}
  \tryingfigtrue \doingfigfalse
  \pageafterfigfalse}
```

Finally, the `\dohang` macro is defined which actually sets the paragraph shape.

```
\newcount\hangcount
\def\dohang{%
  \dimen0=\ht\figbox
  \advance\dimen0 by -\htdone
  \advance\dimen0 by 1.49\baselineskip
  \hangcount=\dimen0
  \divide\hangcount by \baselineskip
  \dimen0=\wd\figbox
  \advance\dimen0 by \figgutter
  \global\hangafter=-\hangcount
  \global\hangindent=-\dimen0}
```

There are two interesting points to note in the `\dohang` macro. First, the calculation of the height of the paragraph which is to be cut out involves adding 1.49 times `\baselineskip`. This increase is done to provide a minimum of one half of `\baselineskip` of gutter space between the bottom of the figure and the x-height of the line of text below the figure. Since the top of the figure is even with the x-height of the first line, we need to increase its height by one half of `\baselineskip`. The additional `.99\baselineskip` is added since numbers are truncated on division; we want the `\divide` to give us a "ceiling" result, not a "floor."

The second point of interest in `\dohang` is the actual calculation of `\hangcount`. Here, we set a count variable to a dimension and then divide the count by another dimension. TeX allows dimensions to be coerced into numbers; the number becomes the value of the dimension in units of scaled points. By dividing a coerced dimension by another dimension, the units are cancelled out leaving us with a count.

Applying the insertion macros

Now that all the pieces have been defined, we can see how to combine them within a document to set the figure. The following TeX commands show the portion of an input file where the figure is to be used.

```
  ⟨define macros previously shown in article⟩
  ⟨text for paragraph preceding figure⟩
\rightinsert
\par
  ⟨text for first paragraph after figure⟩
\par
  ⟨text for next paragraph⟩
\par
  ⟨text for still another paragraph⟩
    ...
```

When using these macros, care should be taken when using grouping. The macros have been designed to allow a group to be entered after the figure has been started. However, the `\rightinsert` call itself should *not* be placed within a group unless you are certain that the entire figure will be covered before the group ends.

Demonstrating the insertion macros

To demonstrate these macros under a variety of circumstances, we need to vary the amount of text prior to starting the figure and the amount which is placed after its start. This brings us to the second major topic of this article: generating variable amounts of random text.

While this may seem like a frivolous application of TeX, it does serve some useful functions. When showing how page layouts will look, typographers often use Latin text to fill out pages. For those of us who don't have any Latin text handy or don't want to type it in, it is possible to have TeX make up some text with the aid of a pseudo-random number generator.

Another benefit of randomly-generated text is that by shifting the position within the random sequence, we can vary the amount of text that will be created. This technique will be used to show how the figure placement macros act under different conditions.

Random numbers in TeX

To start, we need a random number generator. The following one uses the linear congruential method. It has a period of 50,000 numbers with chi-squared values for 1000 number sequences falling within the 35% to 70% range. A call to `\rnd` results in `\rndval` being set to the next number in the pseudo-random sequence. The number will be between 0 and 99, inclusive.

```
\newcount\rndnum
\newcount\rndval
\newcount\rndtemp
\rndnum=0
\def\rnd{%
  \global\multiply\rndnum by 371
  \global\advance\rndnum by 1
  \ifnum\rndnum>99999
    \rndtemp=\rndnum
    \divide\rndtemp by 100000
    \multiply\rndtemp by 100000
    \global\advance\rndnum by -\rndtemp
  \fi
  \global\rndval=\rndnum
  \global\divide\rndval by 1000 \relax}
```

Now, the random number generator can be used to generate random paragraphs consisting of a random number of sentences; random sentences made up of a variable number of words; and words made up from a number of randomly selected letters. Following the normal practice in English, we capitalize the first word of each sentence.

```
\newcount\ns \newcount\nw
\newcount\nc \newcount\np
\newcount\ASCII
\def\randompar{\rnd \ns=\rndval
  \divide\ns by 10\advance\ns by 3
  \loop \ifnum\ns>0 {\randomsent}.%
    \advance\ns by -1 \repeat}
\def\randomsent{\rnd \nw=\rndval
  \divide\nw by 7 \advance\nw by 5
  \ASCII="41
  \loop \ifnum\nw>0 { \randomword}%
    \advance\nw by -1 \repeat}
\def\randomword{\rnd \nc=\rndval
  \divide\nc by 15 \advance\nc by 2
  \loop \ifnum\nc>0 {\randomchar}%
    \advance\nc by -1 \repeat}
\def\randomchar{\rnd
  \multiply\rndval by 29
  \divide\rndval by 100
  \ifnum\rndval=26 \rndval=0 \fi
  \ifnum\rndval>26 \rndval=4 \fi
  \advance\rndval by \ASCII
  \char\rndval \global\ASCII="61}
```

Applying the random numbers

Using the random text generator to demonstrate the figure placement macros can be accomplished with the following TeX code.

```
  (define macros for figure placement)
  (define macros for random text generation)
\message{Give me a number from 0 to 99:}
\read-1 to\mynum \ns=\mynum
\ifnum\ns>99 \ns=99 \fi
\ifnum\ns=1 Skipping 1 number.
\else Skipping \number\ns\ numbers.\fi
\loop \ifnum\ns>0 \rnd
  \advance\ns by -1 \repeat
\rnd \ifnum\rndval>49
  \parskip=6pt plus 4pt minus 2pt
  \parindent=0pt
\else
  \parskip=0pt plus 2pt
  \parindent=20pt
\fi
\def\dopar{\par}
\rnd \np=\rndval
\divide\np by 20 \advance\np by 5
\loop \ifnum\np>0 \dopar {\randompar}%
  \advance\np by -1 \repeat
\rightinsert
\par
Insert the figure here or soon after.
\rnd \np=\rndval
\divide\np by 10 \advance\np by 7
\loop \ifnum\np>0 \dopar {\randompar}%
  \advance\np by -1 \repeat
\bye
```

Those who actually try the macros presented in this article are advised to try the following sequence numbers: 7, 8, 28 and 42–43.

Improvements to random text generator

Words composed of randomly selected letters cause several problems. First, it is a very time-consuming operation to select each character at random. Further, the words do not tend to hyphenate very well. This can result in overfull hboxes (try sequences 8, 12 or 73) or word spacing that is too loose. If one is showing a layout that has narrow columns, it is desirable for paragraphs to be set without any bad breaks.

An improvement would be to select words at random from a pre-built list of words. This requires considerably fewer random numbers. Words should be chosen of varying lengths so that the text appears to be realistic. To avoid distracting the reader into trying to make sense out of the text, the word list should be made up using a language that the reader is not likely to know. Latin perhaps?

Problems with the insertion macros

One weakness that these macros have is that they don't account for any stretching of the `\parskip` glue. This can result in an extra line being shortened below the figure if the `\parskip` glue between the paragraphs covering the figure is allowed to stretch too much (try sequence 4).

This extra line is normally not a big problem. However, if the figure is being placed close to the bottom of the page, it is possible for the extra shortened line to be placed at the top of the next page. The \tryfig macro avoids this problem by subtracting an extra two times \baselineskip from the remaining space on the page before testing to see if there is room.

An example of the problem can be seen by removing the extra \advance\dimen0 from Section A or \tryfig and running the macros with random sequences 42 and 43.

Improvements to the insertion macros

When a figure won't fit on a page, it is deferred until a paragraph break on the next page. However, it does not always start with the first paragraph on the next page (try sequences 9 and 35). Sequence 55 shows a variation of this same problem. The output routine lags behind the building of the vertical list. Thus, it is possible that when this "first" paragraph break occurred, TEX was still processing the earlier page. The improvement to the figure placement would be to get the figure to be output as soon as possible after the page break.

A further refinement would be to get the figure to start at the very top of the next page. This might involve placing the figure in the middle of a paragraph.

Perhaps the definitive improvement to these figure insertion techniques would be to define a new \insert (say, \newinsert\rightins) and revise the output routine to handle a \rightins as it does other inserts. This is where output routines do become nasty beasts.

LaTEX

Contents of LaTEX Style Collection as of 6th September 1987

Ken Yap
University of Rochester

The LaTEX style collection now contains the files listed below. They are available for anonymous ftp from `Rochester.Arpa` in directory `public/latex-style`. You should retrieve the file `00index` first to obtain a brief description of current directory contents. The file `00directory` contains a reverse time sorted list of files; this may be helpful in keeping your collection in sync with LaTEX-style.

File	Description
`00directory`	
`00index`	
`00readme`	
`a4.sty`	Set page size to A4
`a4wide.sty`	Adjusts width too to suit A4
`aaai-instructions.tex`	Instructions to authors
`aaai-named.bst`	BiBTEX style to accompany `aaai.sty`
`aaai.sty`	Style file for AAAI conference 1987
`acm.bst`	ACM BibTEX style
`agugrl.sty`	AGU Geophysical Research
*`agugrl.sample`	Letters style, sample
`agujgr.sty`	AGU Journal of Geophysical
*`agujgr.sample`	Research style, sample
`amssymbols.sty`	Load AMS symbol fonts
*`article.txt`	Standard files in text format,
*`art10.txt`	with places to make
*`art11.txt`	language specific
*`art12.txt`	changes indicated
`biihead.sty`	Underlined heading
`cyrillic.sty`	Load cyrillic font
`dayofweek.tex`	Macros to compute day of week and phase of moon Examples of how to use TEX arithmetic capabilities
`deproc.sty`	DECUS Proceedings style
`deprocldc.tex`	Paper that describes the above
`docsty.c`	Program to convert .doc to
`docsty.readme`	.sty by stripping comments
`doublespace.sty`	Double spacing in text
*`draft.sty`	Draft option for documents for "debugging"
`drafthead.sty`	Prints DRAFT in heading
`dvidoc.shar1`	Sh archive of DVIDOC, DVI
`dvidoc.shar2`	to character device filter for Unix BSD systems
`dvidoc.sty`	Style file to substitute all fonts with doc font
`epic.shar1`	Sh archive of extended
`epic.shar2`	picture environment
*`espo.sty`	Style file for Esperanto
`format.sty`	Print FP numbers in fixed format
`fullpage.doc`	Get more out of a page
`fullpage.sty`	
`geophysics.sty`	Geophysics journal style
*`german.sty`	Redefines keywords for German documents