Typesetting paragraphs of a specified shape

Donald Arseneau

4694 West 8 Ave,
Vancouver BC,
V6R 2A7 Canada
asnd@reg.triumf.ca

(*Editor's note*: This description of an unusual macro file, `shapepar.sty`, is taken from the documentation; the full style file can be found in CTAN archives.)

`\shapepar` is a macro to typeset paragraphs of a specified shape. The total size is adjusted automatically so that the entire shape is filled with text. This is distinct from the normal `\parshape` command which specifies a shape *and* a size, which may be only partially filled, or over-filled, from top to bottom. In a `\shapepar` there can be no displayed math, and no `\vadjust` material, (including `\vspace`). `\Shapepar` (capital S) is just like `\shapepar` except the paragraph is boxed so it cannot be split over two pages. Shaping paragraphs this way is a slow process, so this style is mainly intended for cards, invitations etc., not for whole books! Although short paragraphs process much faster, only long paragraphs accurately fill complex shapes.

These macros work for both LaTeX and plain TeX. For LaTeX, specify `\documentstyle[...shapepar...]`, or for either, `\input shapepar.sty`.

The command `\shapepar` should be used at the beginning of a paragraph, and it applies to the entire paragraph. There is one parameter: a description of the shape, <*shape_spec*>.

```
  \shapepar {<shape_spec>} Text of para...
```

The syntax rules for <*shape_spec*> are very specific, and must be followed closely. (In these rules, `{ }` mean explicit braces, `[ ]` denote optional parts, `< >` surround a keyword that is defined (perhaps loosely), and `|` means 'or'; do not type `[ ] < >` or `|`, only `{ }`.)

```
<shape_spec}> = {<h_center>} <lines>
<lines> = <line_spec> [\\<lines>]
```

That is, the shape is specified as a single number in braces, followed by the specifications for the lines, with the lines separated by `\\`. The final paragraph will have its <*h_center*> position centered on the page. <*h_center*> is a number (like 10.5) of arbitrary units; whatever units are used for lengths and positions in the <*lines*>, they just need to be consistent.

The lines in the spec are not lines of text; nor are they the lines that you would use to draw the shape itself. They are horizontal scans across the shape at irregular intervals. Curved shapes need many scan lines for accurate rendering while simple shapes need few. Draw a shape on paper, then draw a series of horizontal lines across the shape, including lines that just touch the top and the bottom of the figure. Each line crosses over pieces of the figure in some region. These intersections of line and figure define a <*line_spec*>.

```
<line_spec> = {<v_pos>} <segment>
                     [ other <segment>s ]
```

The <*v_pos*> is the vertical position of the line. Each <*line_spec*> must have a position greater than or equal to that of the previous line, and with all <*v_pos*> > -1000. Position is measured from top to bottom, and always moving down. Each <*segment*> represents a region where text will go in the final paragraph; it is the segment of the horizontal scan line that overlaps the body of the figure. There are five types of segment:

```
<segment> = t{pos}{len} | b{pos} |
                          e{pos} | s | j
```

`b{pos}`	begin text at a point at horizontal position pos
`e{pos}`	end text at a point at horizontal position pos
`t{pos}{len}`	make a block of text at position pos with length len
`s`	split text (begin whitespace)
`j`	join two text blocks (end a gap)

The most common type of segment is t (text). The other types are degenerate in that they are single points rather than finite segments. Types s and j have no explicit position, but they must appear between text segments, and those texts should abut; e.g., `t{3}{2}st{5}{4}` (text from 3 to 5 and text from 5 to 9).

Let's jump right into a simple example, and the meanings will be clearer. A 'diamond' shape can have the four vertices:

```
                         (x=1,y=0)
                             .
+---> x
!            (0,1) .        . (2,1)
!
V y                          .
                           (1,2}
```

This shape can be exactly specified by just three scan lines passing through the vertices. The specification is:

`{1}%`	*h_center*: $x = 1$
`{0}b{1}\\%`	text block begins at point y=0, x=1
`{1}t{0}{2}\\%`	this scan (at y=1) crosses text (len=2) starting at x=0
`{2}e{1}`	text block ends at point y=2, x=1

Other specification lines, like `{1.5}t{0.5}{1}\\%` could be inserted, but would make no difference–the shape is interpolated linearly between scan lines.

Every block of text must start with a b specifier and end with an e spec. on some line below. Every segment specified by t must have a length greater than zero. If two blocks of text merge to form one (like at the top of a heart shape) there should be a j spec at the point of junction. If one block bifurcates (like at the top of a hole in a doughnut) there should be an s spec.

Thus, the first line for any valid shape description must consist of only b segment descriptors; the last line can only have e type descriptors. Although the definition of the units is arbitrary, the numbers should range in magnitude from $\tilde{}.1 - 100$ to avoid numeric overflows and underflows.

If there are errors in the format of the specification, `\shapepar` might complain with the error message ‘Shaped Paragraph Error: Error in specification. Check carefully!’ At this point you may as well type `x` or `e`, as there is very little chance that TEX will continue successfully. You might also get one of TEX’s regular error messages, like ‘Illegal unit of measure (pt inserted).’ or ‘Missing number, treated as zero.’ or you might get no error message at all, just ridiculous formatting. Check shape syntax carefully against the rules and the examples before running them through TEX.

What to do if the figure does not start at a point–if it has a flat top? It can start at a single point, but have the next scan line at the same vertical position! A square paragraph is specified by:

`{1}%`	centerline at x=1
`%`	(x=1 is horizontally centered on page)
`{0}b{0}\\%`	begin at (0,0)
`{0}t{0}{2}\\%`	text at y=0, width=2
`{2}t{0}{2}\\%`	text at y=2, width=2
`{2}e{1}%`	end at (1,2)

Both `\diamondpar` and `\squarepar` are defined as paragraphs with these shapes.

Sit
at word
processor for
two hours composing
title for new book. Head
for tea room at 11 0’clock where
six colleagues are sitting round walls
in silence. Reminds me of mental hospital
day room but no strait jackets, except perhaps in-
tellectual. Make coffee with back to them in order to
surreptitiously use someone else’s milk from fridge. Won-
der why no one ever asks me what I have been doing
in the department for the last three years.
Listen to sudden burst of animated con-
versation about new Mac software,
and realise that my social
isolation is due to the
fact that I have
an Am-
strad.

Now let’s get more ambitious. A heart shape must have two simultaneous beginnings, a short stretch of separate text, ending with a join, thereafter there is just one stretch of text leading to the final bottom point. This shape has many scan lines so that the smooth flowing curves are preserved.

```
\def\heartshape{%
{20}{0}b{13.32}b{26.68}%
\\{.14}t{10.12}{4.42}t{25.46}{4.42}%
\\{.7}t{9.14}{7.16}t{23.7}{7.16}%
\\{1.4}t{8.4}{9.02}t{22.58}{9.02}%
\\{2.1}t{7.82}{10.42}t{21.76}{10.42}%
\\{2.8}t{7.36}{11.58}t{21.06}{11.58}%
\\{3.5}t{6.98}{12.56}t{20.46}{12.56}%
\\{4.2}t{6.68}{13.32}jt{20}{13.32}%
\\{4.9}t{6.48}{27.04}%
\\{5.6}t{6.34}{27.32}%
\\{6.3}t{6.28}{27.44}%
\\{7}t{6.26}{27.48}%
\\{7.7}t{6.27}{27.46}%
\\{8.4}t{6.32}{27.36}%
\\{9.1}t{6.4}{27.2}%
\\{9.8}t{6.52}{26.96}%
\\{10.5}t{6.68}{26.64}%
\\{11.9}t{7.12}{25.76}%
\\{13.3}t{7.72}{24.56}%
\\{14.7}t{8.51}{22.98}%
\\{16.1}t{9.5}{21}%
\\{17.5}t{10.69}{18.62}%
\\{18.9}t{12.08}{15.84}%
\\{20.3}t{13.7}{12.6}%
\\{21.7}t{15.62}{8.76}%
\\{22.4}t{16.7}{6.6}%
\\{23.1}t{17.87}{4.26}%
\\{24.6}e{20}%
}
```

Sit at word processor for
two hours compos- ing title for new
book. Head for tea room at 11 0'clock where six
colleagues are sitting round walls in silence. Re-
minds me of mental hospital day room but no strait
jackets, except perhaps intellectual. Make coffee
with back to them in order to surreptitiously use
someone else's milk from fridge. Wonder why no
one ever asks me what I have been doing in the
department for the last three years. Listen to
sudden burst of animated conversation
about new Mac software, and real-
ise that my social isolation is
due to the fact that I
have an Am-
strad.

Look at `\heartshape` and find the two b specifiers at the beginning; find the j a few lines below. Notice that above the j there are two segments per line, but only one below it; the text to the left and right of the join meet at the join point: 20. I drew this heart freehand, and measured lengths from the sketch, so you should be able to do better!

Text can have holes. For example, a doughnut-shape would have a b on the first line, followed by some lines with a single t, then a line with $t\ s\ t$ at the start of the hole. The hole is represented by lines with two t specs–the gap between them is the hole. A line with $t\ j\ t$ ends the hole. There are more lines with single t, and then an e line to end with. Our final example is a nut. Not a doughnut, but a hex-nut (for a machine screw) — a regular hexagon with a circular hole in the center. The hexagon is flat on top and bottom so the specification begins and ends like the square shape. The circle is rendered as a 24-gon, beginning with a split (s) of the surrounding text and ending with a join (j). If the spacing of the scan lines looks odd, it is because the hexagon alone would need few scans, but the circle needs many; the points on the circle are at 15 degree intervals.

```
\def\nutshape{%
{0}%
{0}b{0}\\%
{0}t{-12.5}{25}\\%
{11.65}t{-19.23}{19.23}st{0}{19.23}\\%
{11.99}t{-19.42}{16.835}t{2.59}{16.835}\\%
{12.99}t{-20}{15}t{5}{15}\\%
{14.58}t{-20.92}{13.85}t{7.07}{13.85}\\%
{16.65}t{-22.11}{13.45}t{8.66}{13.45}\\%
{19.06}t{-23.51}{13.85}t{9.66}{13.85}\\%
{21.65}t{-25}{15}t{10}{15}\\%
{24.24}t{-23.51}{13.85}t{9.66}{13.85}\\%
{26.65}t{-22.11}{13.45}t{8.66}{13.45}\\%
{28.72}t{-20.92}{13.85}t{7.07}{13.85}\\%
{30.31}t{-20}{15}t{5}{15}\\%
{31.31}t{-19.42}{16.835}t{2.59}{16.835}\\%
{31.65}t{-19.23}{19.23}jt{0}{19.23}\\%
{43.3}t{-12.5}{25}\\%
{43.3}e{0}%
}
```

Sit at word processor for two
hours composing title for new
book. Head for tea room at 11
0'clock where six colleagues are sit-
ting round walls in silence. Reminds me
of mental hos- pital day room
but no strait jackets, except
perhaps intel- lectual. Make
coffee with back to them in order
to surreptitiously use someone
else's milk from fridge. Wonder
why no one ever asks me what I
have been doing in the department
for the last three years. Listen to sudden
burst of animated conversation about
new Mac software, and realise that
my social isolation is due to the
fact that I have an Amstrad.

`\shapepar` cheats a bit when the horizontal gap between two bits of text is small (like down in the notch of `\heartpar`). When the gap is less than an inter-word space it is eliminated, and the texts are joined; when it is somewhat larger it is expanded to give it more visibility.

Since the processing is slow, there are some messages to say how things are going. These can be eliminated to save space (Put a % at the start of every `\message` line.) Or you can get even more verbose messages by *removing* the % that precedes many other `\message` commands.

There are also a number of parameters which can be changed to affect the size-optimization procedure. Search for the word *optimize* in the source.