

The pullquote package

Stephan Lehmke

<mailto:Stephan.Lehmke@QuinScape.de>

www.docscape.de

? from ?

Abstract

A *pull quote* (also known as a lift-out quote) is a quotation or excerpt from an article that is typically placed in a larger or distinctive typeface on the same page, serving to entice readers into an article or to highlight a key topic (from [Wikipedia](#)).

In journal publishing, where multi-column typesetting is common, a pull quote is usually placed between two columns, inside a ‘window’ which is cut out of the columns’ text flow. Pictures and other graphical objects are also often placed this way.

This package implements an environment for typesetting a balanced two-column text with a cut-out window of customizable shape in which an arbitrary object is positioned.

Contents

1	Introduction	2
2	Requirements	3
2.1	Required Packages	3
2.2	External Calls to Image Magick	3
2.2.1	Remarks on Ubuntu Linux	5
2.2.2	Remarks on Windows	5
3	Usage	6
3.1	Package Options	6
3.2	Basic Configuration	7
3.3	Basic Usage	7
3.4	Environment Options	11
3.4.1	Geometry Configuration	11
3.4.2	Object Specification	13
3.4.3	Vertical Position of Object Window	14
3.4.4	Window Shape	15

3.5	Shape Functions	18
3.6	Image Shapes	19
3.7	Typesetting Text in Tight Columns	19
3.8	Adding Captions	19
4	Limitations	19
5	Possible Extensions	19
6	Implementation	19
6.1	Initialization and Package Options	19
6.2	User Interface	20
6.2.1	Configuration	20
6.2.2	Environment Options	20
6.2.3	Environment Definition	21
6.3	Internals	25
6.3.1	Auxiliary Registers and Containers	25
6.3.2	Internal Macros	26
6.4	Shape Functions	29
6.4.1	Rectangular shape	30
6.4.2	Circular shape	30
6.5	Image shapes	32
7	Change History	35
8	Index	35

Examples

1	Pull quote with tabular text <code><object></code> .	8
2	Pull quote with image <code><object></code> .	9
3	Pull quote with several options.	10
4	<code>textcoldist</code> example.	13
5	<code>objdist</code> example.	14
6	<code>textcolwd</code> example.	15
7	<code>image</code> example.	16
8	<code>objvalign=top</code> example.	16
9	<code>objvalign=bottom</code> example.	17
10	<code>shape=image</code> example.	18

1 Introduction

This is an *experimental* package for inserting an arbitrary object into a two-column balanced text flow such that a “window” of appropriate size is cut out of the text at the place the object is inserted.

Different window shapes are supported, for instance rectangular and circular shapes. New shapes can easily be added by providing a specific type of macro called *shape function*.

In its current state, the package is more like a proof of concept, demonstrating how this effect can be achieved by T_EX macro programming. For being really useful, there are too many restrictions at the time being. See sections 2, 4, and 5.

Figures 1–3 show some examples of the style of formatting possible with this package.

You are invited to test this package and find useful applications for it, but please be prepared for unexpected failures.

The package was implemented in the course of answering the questions “Implementing a pullquotes algorithm in L^AT_EX” and “Two-column text with circular insert” on the Q&A site T_EX Stack Exchange. Report bugs at the T_EX-SX Launchpad site. There is also a [chatroom](#) dedicated to the T_EX-SX packages.

As soon as the package is a bit more bug-free, basically documented and acceptably user-friendly, it will be prepared for publication on CTAN.

2 Requirements

2.1 Required Packages

The `pullquote` package automatically loads the following further packages:

1. `etoolbox`, `environ`, `keyval`.
2. `microtype` (only if the package option `nomicrotype` is *not* given): The text formatting done by this package can get awfully ‘tight’ when text flows around objects. `microtype` significantly improves typographic quality in these situations. Use the option `nomicrotype` only if you want to load `microtype` yourself and avoid option clashes.
3. `graphicx` (only if the package option `noimageshapes` is *not* given): As the `pullquote` environment with the option `shape=image` executes a call to `\includegraphics` in the `graphicx` variant, this package is needed unless you explicitly deactivate that option.

2.2 External Calls to Image Magick

The `pullquote` environment with the option `shape=image` will generate an external system call to the command `convert` from the image manipulation software [Image Magick](#). The purpose of this call is to determine the *shape* of an inserted image to get an appropriate ‘smooth’ text flow around the image.

This option doesn’t make sense without that call, so to use this option, you need to fulfil the following requirements.

1. The software package [Image Magick](#) should be installed on your computer such that the command `convert` can be called from a command shell. You

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

*Wir müssen wissen.
Wir werden wissen.*
DAVID HILBERT

Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Figure 1: Text with rectangular insert using `pullquote`.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.



Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Figure 3: Text with insert based on image shape using `shape=image`. Image by [Paulo Cereda](#).

They were indeed a queer-looking party that assembled on the bank—the birds with dragged feathers, the animals with their fur clinging close to them, and all dripping wet, cross, and uncomfortable. The first question of course was, how to get dry again: they had a consultation about this, and after a few minutes it seemed quite natural to Alice to find herself talking familiarly with them, as if she had known them all her life. Indeed, she had quite a long argument with the Lory, who at last turned sulky, and would only say, 'I am older than you, and must know better'; and this Alice would not allow without knowing how old it was, and, as the Lory positively refused to tell its age, there was no more to be said.

At last the Mouse, who seemed to be a person of authority among them, called out, 'Sit down, all of you, and listen to me! I'll soon make you dry enough!' They all sat down at once, in a large ring, with the Mouse in the middle. Alice kept her eyes anxiously fixed on it, for she felt sure she would catch a bad cold if she did not get dry very soon.

'Ahem!' said the Mouse with an important air, 'are you all ready? This is the driest thing I know. Silence all round, if you please! 'William the Conqueror,

whose cause was favoured by the pope, was soon submitted to by the English, who wanted leaders, and had been of late much accustomed to usurpation and conquest. Edwin and Morcar, the earls of Mercia and Northumbria—'

'Ugh!' said the Lory, with a shiver. 'I beg your pardon!' said the Mouse, frowning, but very politely: 'Did you speak?'

'Not I!' said the Lory hastily. 'I thought you did,' said the Mouse. '—

I proceed. 'Edwin and Morcar, the earls of Mercia and Northumbria, declared for him: and even Stigand, the patriotic archbishop of Canterbury, found it advisable—'

'Found what?' said the Duck.

'Found it,' the Mouse replied rather crossly: 'of course you know what 'it' means.'

'I know what 'it' means well enough, when I find a thing,' said the Duck: 'it's generally a frog or a worm. The question is, what did the archbishop find?'

The Mouse did not notice this question, but hurriedly went on, '—found it advisable to go with Edgar Atheling to meet William and offer him the crown. William's conduct at first was moderate. But the insolence of his Normans—' How are you getting on now, my dear?' it continued, turning to Alice as it spoke.



Figure 2: Text with circular insert using `shape=circular`. Text and image by LEWIS CARROLL (pdf from [gasl.org](#)) [Public domain], via Wikimedia Commons.

can check whether your installation will work for the purposes of this package by pasting the following call into a command shell **all on one line**:

```
convert pq-duck.pdf -resize 124.99362x123.20798! -bordercolor white
-border 10x10 -morphology Erode Disk:10.3 -resize 26x13!
-black-threshold 95% -monochrome pq-duck.pdf.pqshape.txt
```

(assuming you're in the installation directory of the pullquote package and the file `pq-duck.pdf` is present).

You should get no error message and a non-empty text file `pq-duck.pdf.pqshape.txt` should be produced.

2. The `shell-escape` feature of `(pdf)tex`, enabling \TeX to execute system commands, should be activated. If you don't want to activate it in a global configuration file, you should call `pdflatex` with the `--shell-escape` option.

If you have generated this documentation and the example text in figure 3 flows around the image, then all is well.

If you don't meet these requirements or are not feeling secure when \TeX is calling external tools, then you can turn off the `shape=image` option by giving the package option `noimageshapes`.

Even without the `shape=image` option, you can let text flow around the *bounding box* of an image by using the default rectangular shape of the `pullquote` environment and giving an `\includegraphics` call as `object`.

2.2.1 Remarks on Ubuntu Linux

That's the system I'm testing with. Installing Image Magick should be as easy as typing

```
apt-get install imagemagick
```

or using some dedicated installation tool like `synaptic`.

I didn't have to configure anything special wrt. `shell-escape`, though I'm not entirely sure why this is so...

2.2.2 Remarks on Windows

I am indebted to the user `speravir` for providing the following comments on using `pullquote` with Windows.

As I have no possibility to test on Windows myself, I did my best to translate it from German to English, but am otherwise providing this advice as-is. If you are getting good results in other ways, please report to me and I'll try to incorporate further advice.

Image Magick needs to be installed on your system. Unfortunately, there is already a system command `convert.exe` for converting FAT-drives into NTFS-drives. So you need to make sure that after installation, `convert` from the Image Magick suite is found *before* the system version of `convert`.

Installation:

1. Download the binary release from <http://www.imagemagick.org/>, ideally in the form of an *Installer*. The portable version will pose problems (see below).
2. Install. It is important that the program inserts itself into the system path `%PATH%` *in front of* the entry for `C:\Windows\system32` to make sure that a call to `convert` will call the program from the Image Magick suite.
3. This will not work with the portable version of Image Magick, so in that case the tool should be called via a batch file where the system path is set.
4. When using MiKTeX-portable, the *start batch* needs to be augmented. See the following answer on T_EX.SX:

<http://tex.stackexchange.com/questions/50911/using-miktex-portable-texmaker-and-asymptote-from-a-usb-drive/51110#51110>

The topic there was *Asymptote*, but it's the same principle.

When executing T_EX, the command line option `--enable-write18` (or the alias `--shell-escape`) has to be set. People using a T_EX-editor need to configure this in the appropriate place.

3 Usage

3.1 Package Options

Currently, there are only some simple options to pre-configure this package:

nomicrotype Normally, the `pullquote` package loads the package `microtype` which enhances typesetting in ‘tight places’. If you don’t want to have this package loaded by `pullquote` because you don’t want to use it or want to load it yourself, you can disable it by this package option.

The package is not strictly necessary for anything `pullquote` does, so apart from slightly worse typesetting quality, you won’t notice anything when giving that option.

noimageshapes The option `shape=image` of the `pullquote` environment requires loading the package `graphicx` as well as the possibility to make an external system call to the software **Image Magick** (see section 2.2).

You can avoid all this by turning off this part of `pullquote` by giving this package option. Note that using the option `shape=image` will give an error message in this case.

3.2 Basic Configuration

All of the configuration parameters for the `pullquote` environment can be specified by appropriate environment options; see section 3.4. Most of these options have *canonical* defaults; for some of them you can specify default values by setting the following registers.

`\textcoldist` **\textcoldist** The distance between the two text columns. Default for the option `textcoldist`. You can set this register to 6mm by

```
\setlength{\textcoldist}{6mm}
```

Default: **4mm**

`\objdist` **\objdist** The distance around the inserted object. Default for the option `objdist`. You can set this register to 6mm by

```
\setlength{\objdist}{6mm}
```

Default: **4mm**

3.3 Basic Usage

To make a two-column text with a pull quote, you need

- some *text*, which will be denoted by `<text>` below and
- an *object* to be inserted between the columns, denoted by `<object>`.

`pullquote` Then creating the pull quote representation is as easy as calling

```
\begin{pullquote}{object=<object>}
<text>
\end{pullquote}
```

Example 1 assumes you have loaded the package `lipsum` which provides a macro for *blind text*.

In general, the `pullquote` environment is called like this:

```
\begin{pullquote}{<options>}
<text>
\end{pullquote}
```

<pre> \begin{pullquote} {object=% {% \large\itshape \begin{tabular}{@{}l@{}} This is the\pullquote text \end{tabular}% }% }% \selectlanguage{latin}\lipsum[1] \end{pullquote} </pre>		
<p> Lorem ipsum dolor sit amet, con- sectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, con- sectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et ne- tus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna </p>	<p> <i>This is the pullquote text</i> </p>	<p> fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>

Example 1: Pull quote with tabular text $\langle object \rangle$.

The content of the `pullquote` environment is just $\langle text \rangle$, while the *mandatory argument* $\langle options \rangle$ of the environment contains a comma-separated option list for configuring the appearance of the pull quote construction. It’s similar to the *key-value style* options which can be given to `\includegraphics` from the `graphicx` package, only here the argument is not optional because the $\langle object \rangle$ always needs to be specified (usually with the `object` key, only in the special case of *image shapes* you need to use the `image` key; see section 3.6). A full list of option keys and their use is described in section 3.4.


The use of `tabular` above is only one way of arranging text for use as $\langle object \rangle$. In fact, every L^AT_EX construct of fixed *width* and *height* qualifies, for instance `\makebox`, `\parbox` or `minipage`.

Another typical choice for $\langle object \rangle$ is an *image*, like in example 2.

Example 3 shows some more options in use, with an object made by clipping an image with a circular path using `TikZ`, a circular-shaped “window”, an explicit *vertical offset* and a slightly tighter distance between text and object than the usual default. The text by LEWIS CARROLL has been assigned to the macro `\alicetext`.

The `pullquote` environment is executed in the following steps:

1. Typeset $\langle object \rangle$ into a *box*, measuring its *width* and *total height*.

<pre> \begin{pullquote} {object={\includegraphics[width=2cm]{pq-duck}}}% \selectlanguage{latin}\lipsum[1] \end{pullquote} </pre>		
<p> Lorem ipsum dolor sit amet, consecte- tuer adipiscing elit. Ut purus elit, vestib- lum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mau- ris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis ege- stas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vesti- bulum urna fringilla ultrices. Phasellus </p>		<p> eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, ma- lesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget ri- sus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>

Example 2: Pull quote with image `<object>`.

2. Add the value given by the option `objdist` (see section 3.4) to the measured width and height as a distance on all sides.
3. *Normalise* the height (including distance) to a full number of text lines (measured in `\baselineskip`).
4. The width and (normalised) height (including `objdist`) give the total size of the “window” to be cut out of the text columns.
5. Calculate the *vertical position* of the window based on its height and the value of the options `objvalign` or `objvoffset`.
6. Calculate a `\parshape` definition for the surrounding `<text>` based on the size and position of the window.
7. Typeset `<text>` in two balanced columns according to the pre-calculated `\parshape` definition. This may take several attempts as it is impossible to estimate the exact number of lines needed to typeset all text (and that number may even vary depending on the exact ‘relative’ position of the window and its influence on paragraph formatting). The correct balance is determined automatically by an internal loop.
8. Arrange the balanced text columns with the object in the predefined position to output the complete *pull quote* construct.

Summary of remarks and restrictions on `<text>`:

```

\begin{pullquote}
{%
  shape=circular,objdist=2mm,objvoffset=3,%
  object=%
  {%
    \begin{tikzpicture}
      \clip (0,0) circle (1.7cm);
      \node (0,0) {\includegraphics[width=3.2cm]{pq-alice}};
    \end{tikzpicture}%
  }
}
\alicetext
\end{pullquote}

```

They were indeed a queer-looking party that assembled on the bank—the birds with draggled feathers, the animals with their fur clinging close to them, and all dripping wet, cross, and uncomfortable.

The first question of course was, how to get dry again: they had a consultation about this, and after a few minutes it seemed quite natural to Alice to find herself talking familiarly with them, as if she had known them all her life.

Indeed, she had quite a long argument with the Lory, who at last turned sulky, and would only say, ‘I am older than you, and must know better’; and this Alice would not allow without knowing how old it was, and, as the Lory positively refused to tell its age, there was no more to be said.

At last the Mouse, who seemed to be a person of authority among them, called out, ‘Sit down, all of you, and listen to me! I’ll soon make you dry enough!’ They all sat down at once, in a large ring, with the Mouse in the middle. Alice kept her eyes anxiously fixed on it, for she felt sure she would catch a bad cold if she did not get dry very soon.

‘Ahem!’ said the Mouse with an important air, ‘are you all ready? This is the driest thing I know. Silence all round, if you please! “William the Conqueror, whose cause was favoured by the pope, was soon

submitted to by the English, who wanted leaders, and had been of late much accustomed to usurpation and conquest. Edwin and Morcar, the earls of Mercia and Northumbria—”

‘Ugh!’ said the Lory, with a shiver.

‘I beg your pardon!’ said the Mouse, frowning, but very politely: ‘Did you speak?’

‘Not I!’ said the Lory hastily.

‘I thought you did,’ said the Mouse. ‘—I proceed. “Edwin and Morcar, the earls of Mercia and Northumbria, declared for him: and even Stigand, the patriotic archbishop of Canterbury, found it advisable—”

‘Found what?’ said the Duck.

‘Found it,’ the Mouse replied rather crossly: ‘of course you know what “it” means.’

‘I know what “it” means well enough, when I find a thing,’ said the Duck: ‘it’s generally a frog or a worm. The question is, what did the archbishop find?’

The Mouse did not notice this question, but hurriedly went on, ‘—found it advisable to go with Edgar Atheling to meet William and offer him the crown. William’s conduct at first was moderate. But the insolence of his Normans—” How are you getting on now, my dear?’ it continued, turning to Alice as it spoke.



Example 3: Pull quote with several options.

- `<text>` should be just continuous text interspersed with `\par`. The result of typesetting `<text>` in the given column width (using the predefined `\parshape` construct) is supposed to be a collection of plain text lines with base line distance `\baselineskip` (with the value in force at the beginning of the `\pullquote` environment). These lines are processed sequentially for calculating the `\parshape` construct and balancing the columns.
- In particular, `<text>` should **not** contain
 - List environments like `itemize`, `quote` or `center`.
 - Displayed math.
 - Section headings.
 - Any commands which change `\baselineskip`.
 - Vertical spacing (or commands inserting it).
 - Nothing which would make a line higher than usual text, which practically rules out `tabular` material.
 - No fancy rules, frames or color tricks which can disturb `\parshape` or vertical splitting of text.

Some of these restrictions may be loosened with future versions of the package; see sections 4 and 5.

3.4 Environment Options

Remember the `pullquote` environment is called like this:

```
\begin{pullquote}{<options>}
  <text>
\end{pullquote}
```

In this section, all the possible option keys which can go into `<options>` are explained, together with their usage and interdependencies.

Table 1 on page 12 gives a summary of all options.

3.4.1 Geometry Configuration

`textcoldist=<len>` sets the distance between text columns produced by `pullquote` to `<len>`. To change the distance to 5mm, use

```
textcoldist=5mm
```

Default: `\textcoldist`

Compare example 1 with example 4.

Table 1: Summary of Environment Options.

Option	Description	Default	P.
<code>textcolldist</code>	$=\langle len \rangle$ Distance between text columns.	<code>\textcolldist</code>	11
<code>objdist</code>	$=\langle len \rangle$ Distance inserted all around $\langle object \rangle$.	<code>\objdist</code>	12
<code>textcolwd</code>	$=\langle len \rangle$ Width of one text column.	$\frac{1}{2} \left(\text{\linewidth} - \langle textcolldist \rangle \right)$	13
<code>object</code>	$=\langle object \rangle$ Object to be inserted between text columns.	<i>mandatory</i>	14
<code>image</code>	$=\langle file\ name \rangle$ Sets $\langle object \rangle$ to be <code>\includegraphics{\langle file\ name \rangle}</code> .	—	14
<code>imageopts</code>	$=\langle opts \rangle$ <code>\includegraphics</code> options in conjunction with <code>image</code> key.	—	14
<code>objvalign</code>	$=\langle top middle center bottom \rangle$ <i>Vertical alignment</i> of the “object window”.	<code>middle</code>	14
<code>objvoffset</code>	$=\langle offset \rangle$ <i>Vertical offset</i> of the “object window”.	—	15
<code>shape</code>	$=\langle rectangular circular image \rangle$ <i>Shape</i> of the “object window”.	<code>rectangular</code>	15
<code>shapefunction</code>	$=\langle cs \rangle$ <i>Shape function</i> .	—	18

`objdist= $\langle len \rangle$` sets the distance inserted all around $\langle object \rangle$ to $\langle len \rangle$. Note that the vertical distance can be slightly larger because it needs to be *normalised* so that the “window” takes a full number of text lines.

To change the distance to 5mm, use

`objdist=5mm`

Default: `\objdist`

Compare example 1 with example 5.

<pre> \begin{pullquote} {textcolldist=1cm,% object=% {% \large\itshape \begin{tabular}{@{}l@{}} This is the\pullquote text \end{tabular}% }% }% \selectlanguage{latin}\lipsum[1] \end{pullquote} </pre>	
<p> Lorem ipsum dolor sit amet, consec- tetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, no- nummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pel- lentesque habitant morbi tri- stique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultri- </p>	<p> ces. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, ia- culis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices biben- dum. Aenean faucibus. Mor- bi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nul- la. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan elei- fend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>

Example 4: `textcolldist` example.

`textcolwd=<len>` sets the width of one text column to `<len>`. If this option is not given, then the default value

$$\frac{1}{2}(\text{\linewidth} - \langle\text{textcolldist}\rangle)$$

is calculated in the moment the `pullquote` environment is executed, so the then-current value of `\linewidth` is respected.

To change the value to 5cm, use

`textcolwd=5cm`

Default: $\frac{1}{2}(\text{\linewidth} - \langle\text{textcolldist}\rangle)$

Compare example 1 with example 6.

3.4.2 Object Specification

An *object* to be inserted between the text columns needs to be specified for every use of `pullquote`. This means one of the following keys has to occur in the mandatory argument of the `pullquote` environment.

<pre> \begin{pullquote} {objdist=1mm,% object=% {% \large\itshape \begin{tabular}{@{}l@{}} This is the\pullquote text \end{tabular}% }% }% \selectlanguage{latin}\lipsum[1] \end{pullquote} </pre>	
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. </p>	<p> Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>
<p><i>This is the pullquote text</i></p>	

Example 5: objdist example.

object=**<object>** specifies the object to be inserted between text columns, in the form of a ‘box-like’ L^AT_EX construct of fixed width and height. See example 1 and section 3.3 for further description and examples.

It is best to generally enclose **<object>** in curly braces {} to avoid conflicts with parsing the key-value list.

In the case that the option **shape=image** is given, the object *needs to be specified with the image key!*

image=**<file name>** specifies the **<object>** to be `\includegraphics{<file name>}`. When the **imageopts**=**<opts>** key is also given, (see below), this becomes **<object>**=`\includegraphics[<opts>]{<file name>}`.

Compare example 2 with example 7.

imageopts=**<opts>** specifies the `\includegraphics` options in conjunction with the **image** key (see above).

3.4.3 Vertical Position of Object Window

objvalign=**(top|middle|center|bottom)** specifies the *vertical alignment* of the “object window” relative to the text columns. The following values for this

```

\begin{pullquote}
{textcolwd=4cm,%
object=%
{%
\large\itshape
\begin{tabular}{@{}l@{}}
This is the\pullquote text
\end{tabular}%
}%
}%
\selectlanguage{latin}\lipsum[1]
\end{pullquote}

```

<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, pla- cerat ac, adipiscing vitae, fel- lis. Curabitur dictum gravida mauris. Nam arcu libero, no- nummy eget, con- sectetuer id, vulpu- tate a, magna. Do- nec vehicula augue eu neque. Pellen- tesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mau- ris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultri- </p>	<p> <i>This is the pullquote text</i> </p>	<p> ces. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Prae- sent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pul- vinar at, mollis ac, nulla. Curabitur auctor semper nul- la. Donec varius orci eget risus. Duis nibh mi, congue eu, ac- cumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>
--	--	---

Example 6: `textcolwd` example.

key are valid:

top Uppermost position.

middle or center Centered position (default).

bottom Lowermost position.


Compare example 1 with examples 8 and 9.

objvoffset=*<offset>* specifies the *vertical offset* of the “object window” from the top of the text columns (as an integer number of lines). See example 3.

3.4.4 Window Shape

shape=(**rectangular**|**circular**|**image**) specifies the *shape* of the “object window”. The following values for this key are valid:

rectangular Box shape (default).

<pre> \begin{pullquote} {image=pq-duck,imageopts={width=2cm}}% \selectlanguage{latin}\lipsum[1] \end{pullquote} </pre>		
<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus </p>		<p> eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>

Example 7: image example.

<pre> \begin{pullquote} {objvalign=top,% object=% {% \large\itshape \begin{tabular}{@{}l@{}} This is the\pullquote text \end{tabular}% }% }% \selectlanguage{latin}\lipsum[1] \end{pullquote} </pre>		
<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna </p>	<p> <i>This is the pullquote text</i> </p>	<p> fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>

Example 8: objvalign=top example.

<pre> \begin{pullquote} {objvalign=bottom,% object=% {% \large\itshape \begin{tabular}{@{}l@{}} This is the\pullquote text \end{tabular}% }% }% \selectlanguage{latin}\lipsum[1] \end{pullquote} </pre>		
<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna </p>	<p> <i>This is the pullquote text</i> </p>	<p> fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. </p>

Example 9: objvalign=bottom example.

circular Circle shape.

If you imagine a *quadratic* window for the rectangular case, then the width (and height) of this window gives the diameter of a circular window, the midpoint of which is in the middle of the square. A trivial conclusion of this is that the area of this circle is smaller than that of the square, so less space is available for *object*.

While for the default case, *object* can have any rectangular shape, here *object* should be *quadratic*, and the real object inside the quadratic box should have the form of a *circle* or *disc*, otherwise it could happen that text overwrites part of the object, or the text otherwise doesn't match the object.

See example 3.

image Window shape derived from image.

In this case, *object* has to be given in the form `image=<file name>`. The shape of the window is calculated from the image file using the image manipulation software *Image Magick*.

The *shape* of the image here means the part of the rectangular bounding box which is not white or transparent. Consequently, for this to have a visible effect, the image should not be a photo, but either clipped with

some path or a drawing with a recognizable shape.

The distance `objdist` is added to the image shape in the form of a ‘smooth’ border. The image object itself is included as it is (i. e. not clipped or anything), but from the way the shape is determined, it is ensured that the text does not overwrite part of the image.

Further documentation is given in section 3.6; see also section 2.2 concerning requirements.


See example 10.

```

\begin{pullquote}
{shape=image,image=pq-duck.pdf,imageopts={width=2cm}}%
\selectlanguage{latin}\lipsum[1]
\end{pullquote}

```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna frin-



gilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Example 10: `shape=image` example.

`shapefunction=<cs>` directly specifies the *shape function*.

Selecting a shape with the `shape` key will internally set a *shape function* representing this shape. With the `shapefunction` key, the shape function can be selected directly.

A *shape function* is an expandable macro with four arguments. It is called internally while the `\parshape` construct for the “window” is calculated. It receives as arguments information about the position of the part of the window being calculated and expands to the width of the cut-out space in the text line at this place.

The value `<cs>` gives the *control sequence* of the shape function macro.

Further documentation is given in section 3.5.

3.5 Shape Functions

Shape functions work at the kernel of the `pullquote` environment. They define the shape to be “cut out” of the text columns to make place for the inserted object. Different shapes can be achieved by using different shape functions.

A *shape function* is an expandable macro with four arguments which will be called as follows.

`\shape function\{<col>\}\{<starty>\}\{<endy>\}\{<line>\}` should expand to a positive *dimension* giving the amount of (horizontal) space which should be left blank in the respective column, counting from the *middle line* between both columns outward. The amount of space between the columns does not need to be considered by the shape function to avoid complicating it; it is subtracted later to get the amount of space to be left blank in the text. Hence, the shape function needs to specify two “half shapes”, one for each column.

The argument `<col>` gives the number of the current column (number 1 or 2). `<starty>` is the *upper* vertical border and `<endy>` is the *lower* vertical border of the region of the shape under consideration, *relative to the upper edge of the insertion window*. Hence, `y=0pt` means the *upper* edge.

Shape functions can use the dimension registers `\windowhextent` and `\windowvextent` giving the (half) total width and total height of the cut-out object window, respectively. For shape functions which absolutely require a quadratic object (like `\circshapefun`), there is also `\windowqhextent` which is the maximum of `\windowhextent` and (half) `\windowvextent` to avoid distortion of the cut-out.

3.6 Image Shapes

3.7 Typesetting Text in Tight Columns

3.8 Adding Captions

4 Limitations

5 Possible Extensions

6 Implementation

6.1 Initialization and Package Options

```

1 \RequirePackage{etoolbox}
2 \RequirePackage{environ}
3 \RequirePackage{keyval}
4
5 \newif\ifmicrotype@pq
6 \microtype@pqtrue
7 \DeclareOption{nomicrotype}{\microtype@pqfalse}
8
9 \newif\ifimgshapes@pq
10 \imgshapes@pqtrue
11 \DeclareOption{noimageshapes}{\imgshapes@pqfalse}
12
13 \ProcessOptions\relax
14
```

```

15 \ifmicrotype@pq
16   \RequirePackage[expansion=alltext]{microtype}
17 \fi
18
19 \ifimgshaped@pq
20   \RequirePackage{graphicx}
21 \fi

```

6.2 User Interface

From an implementation perspective, the user interface consists of three parts: Some basic configuration registers, parsing the key-value style environment options, and the environment definition itself.

<code>\pullquote</code>	The user interface has been changed. There is no macro <code>\pullquote</code> any more. It has been replaced by the <code>pullquote</code> environment. See below.
<code>\pullquotecircular</code>	The user interface has been changed. There is no macro <code>\pullquotecircular</code> any more. It has been replaced by the <code>pullquote</code> environment with option <code>shape=circular</code> . See below.
<code>\pullquoteshape</code>	The user interface has been changed. There is no macro <code>\pullquoteshape</code> any more. It has been replaced by the <code>pullquote</code> environment with option <code>shapefunction=<shapefunction></code> . See below.

6.2.1 Configuration

<code>\textcoldist</code>	Distance between columns (default for environment option).
<pre> 22 \newdimen\textcoldist 23 \textcoldist4mm\relax </pre>	
<code>\objdist</code>	Distance around inserted object (default for environment option).
<pre> 24 \newdimen\objdist 25 \objdist4mm\relax </pre>	

6.2.2 Environment Options

```

26
27 \define@key{pq}{textcoldist}{\textcoldist=#1\relax}
28 \define@key{pq}{objdist}{\objdist=#1\relax}
29 \define@key{pq}{textcolwd}{\textcolwd=#1\relax}
30 \define@key{pq}{objvalign}
31 {%
32   \ifcsname do@valign@#1@pq\endcsname
33     \expandafter\let\expandafter\objvalign@pq\csname do@valign@#1@pq\endcsname
34   \else
35     \PackageError{pullquote}{Invalid valign}{Only the values
36       'top', 'bottom', 'center', or 'middle' are valid for key
37       'valign'. The value you gave
38       was ignored.}

```

```

39     \fi
40 }
41 \define@key{pq}{objvoffset}{\def\objvoffset@pq{#1}}
42 \define@key{pq}{object}{\def\obj@pq{#1}}
43 \define@key{pq}{image}{\def\img@pq{#1}}
44 \define@key{pq}{imageopts}{\def\imgopts@pq{#1}}
45 \define@key{pq}{shape}{\def\shape@pq{#1}\expandafter\let\expandafter\shapefun@pq\csname#1s
46 \define@key{pq}{shapefunction}{\def\shape@pq{fun}\let\shapefun@pq#1}
47
48 \def\constant@image@pq{image}
49 \def\constant@pullquote@pq{pullquote}
50
51 \newcommand\do@valign@middle@pq
52 {\numexpr(\pqlines@pq-\objlines@pq)/\tw@\relax}
53 \newcommand\do@valign@center@pq
54 {\numexpr(\pqlines@pq-\objlines@pq)/\tw@\relax}
55 \newcommand\do@valign@top@pq{z@}
56 \newcommand\do@valign@bottom@pq
57 {\numexpr\pqlines@pq-\objlines@pq\relax}

```

6.2.3 Environment Definition

```

pullquote \begin{pullquote}[\langle options \rangle]
           \langle text \rangle
           \end{pullquote}

```

will typeset $\langle text \rangle$ in two (balanced) columns, embedding an $\langle object \rangle$ (which has to be specified in $\langle options \rangle$) in the middle such that text ‘flows around’ the inserted object.

$\langle text \rangle$ should be just text interspersed with $\backslash par$. No lists, displayed math etc. $\langle object \rangle$ should be a singular object of fixed width like $\backslash includegraphics$ or \tikzpicture , but it could as well be a $\backslash parbox$. Make sure the size of the object and the amount of text match such that the object can be effectively ‘flowed around’.

User documentation is found in section 3.3.

```

58 \NewEnviron{pullquote}[1]
59 {%
60   \ifx\@currenenv\constant@pullquote@pq
61     \textcolwd\z@
62     \let\objvalign@pq\do@valign@middle@pq%
63     \let\objvoffset@pq\empty
64     \let\obj@pq\empty
65     \let\img@pq\empty
66     \let\imgopts@pq\empty
67     \let\shape@pq\empty
68     \let\shapefun@pq\rectangularshapefun
69     \setkeys{pq}{#1}%
70     \ifdim\textcolwd=\z@
71       \textcolwd\dimexpr.5\linewidth-.5\textcoldist\relax
72     \fi

```

```

73     \ifx\img@pq\empty
74     \else
75     \def\obj@pq
76     {\expandafter\includegraphics\expandafter[\imgopts@pq]{\img@pq}}%
77     \fi
78     \ifx\shape@pq\constant@image@pq
79     \ifimgshapes@pq
80     \ifx\img@pq\empty
81     \PackageError{pullquote}{No image given}{You need to
82     specify an image with the key "image". Your command
83     was ignored. No output will be generated.}
84     \else
85     \@pullquoteimgshape@pq
86     \fi
87     \else
88     \PackageError{pullquote}{Image shapes disabled}{Your command
89     was ignored. No output will be generated. Try again
90     without package option "noimageshapes".}
91     \fi
92     \else
93     \ifx\obj@pq\empty
94     \PackageError{pullquote}{No object given}{You need to give one of
95     the keys "object" or "image" to specify the object to
96     insert between columns. Your command
97     was ignored. No output will be generated.}
98     \else
99     \@pullquote@pq
100    \fi
101    \fi
102    \else
103    \PackageError{pullquote}{pullquote is an environment now}{The
104    macro \string\pullquote\space does not exist any
105    more. Please use
106    \string\begin{pullquote}...\string\end{pullquote}. Your command
107    was ignored. No output will be generated.}
108    \fi
109    }

```

\@pullquote@pq

```

110    \newcommand\@pullquote@pq
111    {%

```

We allow widows and orphans as they would lead to glitches in the paragraph shape.

```

112    \clubpenalty=\z@
113    \widowpenalty=\z@

```

Make sure both columns start at the same vertical position.

```

114    \splittopskip\dimexpr\baselineskip-\dp\strutbox\relax

```

Don't complain about underfull boxes at \vsplit.

```

115      \vbadness\maxdimen
116      \vfuzz\maxdimen
Put the object into a box which can be measured.
117      \setbox\objbox@pq
118      =\hbox{%
119      \obj@pq%
120      }%
The text is typeset once to get a rough estimate of the required number of lines.
121      \typesettext@pq{\BODY}{}%
Calculate the number of lines for text and object.
122      \pqlines@pq
123      =\numexpr
124      \dimexpr\ht\textbox@pq+\dp\textbox@pq\relax
125      /\baselineskip
126      /\tw@
127      \relax
128      \objlines@pq
129      =\numexpr
130      \dimexpr\ht\objbox@pq+\dp\objbox@pq+2\objdist\relax
131      /\baselineskip
132      \relax
(Half) total width of the object including margin.
133      \windowhextent=\dimexpr.5\wd\objbox@pq+\objdist\relax
Total height of the object including margin.
134      \windowvextent=\objlines@pq\baselineskip
(Half) total width of the object including margin (assuming quadratic object).
135      \windowqextent=.5\windowvextent
136      \ifdim\windowhextent>\windowqextent
137      \windowqextent\windowhextent
138      \fi
Text width on the side of object.
139      \narrowhsize@pq
140      =\dimexpr\textcolwd-\windowhextent+.5\textcoldist\relax
Column line count is only a rough estimate, not considering the text extension by
\parshape. So we \loop until correct column line count is reached.
141      \loop
142      \typeout{trying \the\pqlines@pq\space lines.}%
Calculate the number of lines above object.
143      \objtopoffset@pq
144      =%
145      \ifx\empty\objvoffset@pq
146      \objvalign@pq
147      \else
148      \objvoffset@pq\relax
149      \fi

```

Number of lines in parshape.

```
150      \global\parshapelines@pq=\numexpr2*\pqlines@pq+\@ne\relax
```

Calculate “global” parshape from object size and position, applying the shape function.

```
151      \xdef\parshape@pq
152      {%
153      \number\parshapelines@pq\space
154      \iterate@mkps@pq{1}{1}%
155      0pt\space\the\textcolwd\space
156      }%
```

Re-typeset text with parshape setting.

```
157      \typesettext@pq{\BODY}
158      {%
159      \let\o@par@pq\par
160      \let\par\par@pq
161      \parshape\parshape@pq
162      }%
```

Split off two columns.

```
163      \setbox\columnabox@pq=\vsplit\textbox@pq to \pqlines@pq\baselineskip
164      \setbox\columnbbox@pq=\vsplit\textbox@pq to \pqlines@pq\baselineskip
```

Iterate until estimation for column line count is correct, which means splitting off the two columns does not leave anything in \textbox@pq.

```
165      \unless\ifvoid\textbox@pq
```

We need to advance line count by half the “leftover” lines in \textbox@pq. To make sure we’re not over-extending the line count (by any strange effect of line breaking with the changed parshape) which could lead to unnecessary white space at the bottom of the right column, we deduce one from the result.

```
166      \@tempcnta
167      \numexpr
168      \dimexpr\ht\textbox@pq+\dp\textbox@pq\relax
169      /\baselineskip
170      /\tw@
171      -\@ne
172      \relax
```

But advance line count by at least one.

```
173      \ifnum\@tempcnta<\@ne\@tempcnta\@ne\fi
174      \advance\pqlines@pq\@tempcnta
175      \repeat
```

When the loop is over, output text columns and object.

```
176      \hbox
177      {%
178      \rlap
179      {%
180      \hskip
181      \dimexpr\narrowhsize@pq+\objdist\relax
```



```

182          \raise
183          \dimexpr
184            \numexpr\pqlines@pq-\objlines@pq-\objtopoffset@pq\relax
185            \baselineskip
186            +.5\dimexpr\windowvextent-\ht\objbox@pq\relax
187            +.5\dp\objbox@pq
188          \relax
189          \box\objbox@pq
190        }%
191        \rlap{\box\columnabox@pq}\hskip\textcolwd\hskip\textcoldist\box\columnbbox@pq%
192      }%
193    }

```

6.3 Internals

6.3.1 Auxiliary Registers and Containers

<code>\textbox@pq</code>	Box for full text content.
194	<code>\newbox\textbox@pq</code>
<code>\columnabox@pq</code>	Box for first column.
195	<code>\newbox\columnabox@pq</code>
<code>\columnbbox@pq</code>	Box for second column.
196	<code>\newbox\columnbbox@pq</code>
<code>\objbox@pq</code>	Box for object.
197	<code>\newbox\objbox@pq</code>
<code>\pqlines@pq</code>	Line count for one column.
198	<code>\newcount\pqlines@pq</code>
<code>\parshapelines@pq</code>	Line count for “global” parshape.
199	<code>\newcount\parshapelines@pq</code>
<code>\objlines@pq</code>	Line count for object.
200	<code>\newcount\objlines@pq</code>
<code>\objtopoffset@pq</code>	Vertical position of object.
201	<code>\newcount\objtopoffset@pq</code>
<code>\textcolwd</code>	Width of text column.
202	<code>\newdimen\textcolwd</code>
<code>\windowhextent</code>	Half the total width of “window”.
203	<code>\newdimen\windowhextent</code>
<code>\windowqextent</code>	Half the total width of “window”, assuming a quadratic object.
204	<code>\newdimen\windowqextent</code>

`\windowvextent` Total height of “window”.

```
205      \newdimen\windowvextent
```

`\narrowhsize@pq` Line width besides object.

```
206      \newdimen\narrowhsize@pq
```

`\parshape@pq` Container for “global” parshape definition.

```
207      \newcommand*\parshape@pq{}
```

6.3.2 Internal Macros

`\typesettext@pq` `\typesettext@pq{<Text>}{<Prefix>}` will typeset `<Text>` as one single column of width `\textcolwd` into `\textbox@pq`, to be split into two separate columns later.

`<Prefix>` can be used to prepend additional settings, for instance the parshape definition.

```
208      \newcommand\typesettext@pq[2]%
209      {%
210          \setbox\textbox@pq
211          =\vbox{%
212              \hsize\textcolwd
```

As lines might get very narrow around the insert “window”, we need more tolerance to avoid overfull lines.

```
213          \tolerance9999\relax
```

Make sure every line has exactly “height” `\baselineskip`.

```
214          \lineskiplimit-\maxdimen
215          \parskip\z@
216          #2%
217          \strut#1%
218      }%
219  }%
```

`\iterate@mkps@pq` `\iterate@mkps@pq{<Line>}{<Column>}` expandably generates the parshape definition by “walking” through the text line by line and column by column and calling the shape function where necessary.

It’s mainly a big case distinction to find out whether the current line is inside the cut-out “window” and other things, with a recursive call at the end (if required).

`<Line>` is in the range `1...\pqlines@pq` and `<Column>` is either 1 or 2.

```
220      \newcommand\iterate@mkps@pq[2]%
221      {%
```

Are we “beyond” the calculated number of `\parshape` lines? In this case the recursion ends.

```
222      \ifnumgreater{#1*#2}{2*\pqlines@pq}
223      {}
224      {%
```

If the line counter exceeds the calculated number of lines in the first column, we restart at line 1 of the second column.

```
225      \ifnumgreater{#1}{\pqlines@pq}
226      {\iterate@mkps@pq{1}{2}}
227      {%
```

To be in the cut-out “window”, the line counter has to be below the upper border of the “image frame”.

```
228      \ifnumgreater{#1}{\objtopoffset@pq}
229      {%
```

If the line counter is even below the lower border of the “image frame”, we’re outside the cut-out part and the shape is just “full line”.

```
230      \ifnumgreater{#1}{\objtopoffset@pq+\objlines@pq}
231      {0pt\space\the\textcolwd\space}
232      {%
```

If we’re inside the cut-out “window”, the parshape expression has to be calculated based on the result of the shape function.

```
233      \ifnumequal{#2}{\@ne}
```

In the first column, the line starts at the left margin; the width is given by the shape function.

```
234      {%
235      0pt\space
236      \the
237      \dimexpr
```

To get the width of the text line, we have to subtract the object width (result of the shape function) from the column width. As the width delivered by the shape function starts right in the middle of the object, we need to subtract half the column distance (which means to add it to the width of the text line). The shape function is called with the column number (here, 1) and the “upper” and “lower” vertical border of the line being calculated (relative to the cut-out “window”) as arguments.

```
238      \textcolwd+
239      .5\textcoldist-
240      \shapefun@pq\@ne
241      {%
242      \the\dimexpr
243      \numexpr#1-\objtopoffset@pq\relax\baselineskip-
244      \baselineskip
245      \relax
246      }
247      {\numexpr#1-\objtopoffset@pq\relax\baselineskip}%
248      {\number\numexpr#1-\objtopoffset@pq\relax}%
249      \relax
250      \space
251      }
```

In the second column, the line starts at the distance given by the shape function and ends at the right margin. As we need the value of the shape function two times here (once for the indent and once for the width), we calculate it only once and give the value as an argument to the helper macro `\right@psexpr@pq` which expands to the output.

```

252      {%
253      \expandafter\right@psexpr@pq\expandafter
254      {%
255      \the\dimexpr
256      \shapefun@pq\tw@
257      {%
258      \the\dimexpr
259      \numexpr#1-\objtopoffset@pq\relax\baselineskip-
260      \baselineskip
261      \relax
262      }
263      {\numexpr#1-\objtopoffset@pq\relax\baselineskip}%
264      {\number\numexpr#1-\objtopoffset@pq\relax}%
265      -.5\textcolldist
266      \relax
267      }%
268      }%
269      }%
270      }%

```

If the line counter is above the upper border of the “image frame”, we’re outside the cut-out part and the shape is just “full line”.

```

271      {0pt\space\the\textcolwd\space}%

```

Recursive call. The line counter is incremented.

```

272      \expandafter\iterate@mkps@pq\expandafter
273      {\number\numexpr#1+\@ne\relax}{#2}%
274      }%
275      }%
276      }

```

`\right@psexpr@pq` `\right@psexpr@pq{<Width>}` is a helper macro which takes the `<Width>` of the object at a certain point and expands to a “right-side” parshape expression.

```

277      \newcommand\right@psexpr@pq[1]
278      {%
279      #1\space
280      \the\dimexpr\textcolwd-#1\relax\space
281      }

```

`\par@pq` This is the internal definition of `\par` which is used for typesetting text in the presence of a “global” parshape definition.

```

282      \def\par@pq
283      {%

```

First, end paragraph with original `\par`.

```
284      \o@par@pq
```

`\prevgraf` gives the number of lines of the just-finished paragraph. If it is smaller than `\parshapelines@pq`, then there will be lines left in the global parshape definition after removing the lines of the previous paragraph.

```
285      \ifnum\prevgraf<\parshapelines@pq
```

In that case, we remove the corresponding number of lines from the global parshape definition and reassign the parshape.

```
286      \global\advance\parshapelines@pq-\prevgraf
```

```
287      \xdef\parshape@pq{\expandafter\gobbleparshapeprefix@pq\parshape@pq}%
```

```
288      \parshape\parshape@pq
```

```
289      \else
```

Otherwise, we just turn off parshaping.

```
290      \global\parshapelines@pq\z@
```

```
291      \fi
```

```
292  }
```

`\gobbleparshapeprefix@pq` Removes lines one-by-one from a parshape definition until there are again exactly `\parshapelines@pq` of them.

```
293      \def\gobbleparshapeprefix@pq#1 #2 #3 %
```

```
294      {%
```

The first item in a parshape definition is the number of lines. If it is greater than `\parshapelines@pq`, call `\gobbleparshapeprefix@pq` recursively (effectively removing #2 and #3) with number reduced by 1.

```
295      \ifnumgreater{#1}{\parshapelines@pq}
```

```
296      {\expandafter\gobbleparshapeprefix@pq\number\numexpr#1-\@ne\expandafter\relax\space}
```

Otherwise, put back parshape line and end recursion.

```
297      {#1 #2 #3 }%
```

```
298  }
```

6.4 Shape Functions

Shape functions are needed to define the shape to be “cut out” of the text columns to make place for the inserted object. Different shapes can be achieved by using different shape functions. In the following, some shape functions for common shapes are predefined.

A *shape function* is an expandable macro with three arguments which will be called as follows.

`<shape function>{<col>}{<starty>}{<endy>}` should expand to a positive *dimension* giving the amount of (horizontal) space which should be left blank in the respective column, counting from the *middle line* between both columns outward. The amount of space between the columns does not need to be considered by the shape function to avoid complicating it; it is subtracted later to get the amount of

space to be left blank in the text. Hence, the shape function needs to specify two “half shapes”, one for each column.

The argument `<col>` gives the number of the current column (number 1 or 2).

`<starty>` is the *upper* vertical border and `<endy>` is the *lower* vertical border of the region of the shape under consideration, *relative to the upper edge of the insertion window*. Hence, `y=0pt` means the *upper* edge.

Shape functions can use the dimension registers `\windowhextent` and `\windowvextent` giving the (half) total width and total height of the cut-out object window, respectively. For shape functions which absolutely require a quadratic object (like `\circshapefun`), there is also `\windowqhextent` which is the maximum of `\windowhextent` and (half) `\windowvextent` to avoid distortion of the cut-out.

6.4.1 Rectangular shape

`\bbshapefun` `\bbshapefun{<col>}{<starty>}{<endy>}{<line>}` ignores its arguments and simply returns the value of `\windowhextent`. This way, the full (rectangular) bounding box of the object including the object distance is cut out.

```
299 \newcommand\rectangularshapefun[4]
300 {%
301     \the\windowhextent
302 }%
```

6.4.2 Circular shape

`\circshapefun` `\circshapefun{<col>}{<starty>}{<endy>}{<line>}` ignores `<col>` and returns a circle approximation based on `\windowqhextent` (as circle radius) and the vertical position given by `<starty>` and `<endy>`.

To get an efficient expandable macro, we split the calculation into three parts:

1. `\circshapefun{<col>}{<starty>}{<endy>}{<line>}` calculates the horizontal position on the circle diameter separately for `<starty>` and `<endy>` (using `\@circshapefun`) and hands the results to `\dimmax@pq`.
2. `\@circshapefun{<y>}` is the circle approximation itself, basically calculating an approximation to

$$\sqrt{\text{\windowqhextent}^2 - <y>^2}$$

3. `\dimmax@pq{<x1>}{<x2>}` expands to the maximum of `<x1>` and `<x2>`.

```
303 \newcommand\circularshapefun[4]
304 {%
305     \expandafter\dimmax@pq\expandafter
306     {%
307         \the\dimexpr
308         \expandafter\@circshapefun\expandafter
309         {%
```

The arguments <starty> and <endy> are counted from the *top* edge of the window (value 0) to the *bottom* edge (value `\windowvextent`). For the circle calculation we normalize this to the mid point (i. e. value 0 occurs at half `\windowvextent`) and non-negative values (i. e. calculating two quarter circles).

```

310         \the\dimexpr
311         \ifdim #2>.5\windowvextent
312         #2-.5\windowvextent
313         \else
314         .5\windowvextent-#2%
315         \fi
316     \relax
317 }%
318 \expandafter
319 \relax
320 \expandafter
321 }%
322 \expandafter
323 {%
324     \the\dimexpr
325     \expandafter\@circshapefun\expandafter
326     {%
327         \the\dimexpr
328         \ifdim #3>.5\windowvextent
329         #3-.5\windowvextent
330         \else
331         .5\windowvextent-#3%
332         \fi
333     \relax
334     }%
335 \relax
336 }%
337 }
```

`\@circshapefun` `\@circshapefun{<y>}` represents the circle approximation itself. It calculates the following approximation formula. Thank you to [tohecz](#) for providing it.

$$r - 0.5 \cdot y^2/r - 0.125 \cdot y^4/r^3 - 0.0625 \cdot y^6/r^5 - 0.0390625 \cdot y^8/r^7$$

Here, r represents the circle radius given by `\windowqhextent` and y stands for the macro argument <y>.

```

338 \newcommand\@circshapefun[1]
339 {%
340     \the\dimexpr
341     \windowqhextent-
342     .5\dimexpr#1\relax*\dimexpr#1\relax/\windowqhextent-
343     \dimexpr
344     \dimexpr0.125\dimexpr#1\relax*\dimexpr#1\relax/\windowqhextent\relax*
345     \dimexpr#1\relax/\windowqhextent
346     \relax*
```

```

347 \dimexpr#1\relax/\windowqhextent-
348 \dimexpr
349 \dimexpr
350 \dimexpr
351 \dimexpr0.0625\dimexpr#1\relax*\dimexpr#1\relax/\windowqhextent\relax*
352 \dimexpr#1\relax/\windowqhextent
353 \relax*
354 \dimexpr#1\relax/\windowqhextent
355 \relax*
356 \dimexpr#1\relax/\windowqhextent
357 \relax*
358 \dimexpr#1\relax/\windowqhextent-
359 \dimexpr
360 \dimexpr
361 \dimexpr
362 \dimexpr
363 \dimexpr
364 \dimexpr0.0390625\dimexpr#1\relax*\dimexpr#1\relax/\windowqhextent\relax*
365 \dimexpr#1\relax/\windowqhextent
366 \relax*
367 \dimexpr#1\relax/\windowqhextent
368 \relax*
369 \dimexpr#1\relax/\windowqhextent
370 \relax*
371 \dimexpr#1\relax/\windowqhextent
372 \relax*
373 \dimexpr#1\relax/\windowqhextent
374 \relax*
375 \dimexpr#1\relax/\windowqhextent
376 \relax
377 }%

```

`\dimmax@pq` `\dimmax@pq{<d1>}{<d2>}` is a generic macro taking two dimensions as arguments and expanding to their maximum.

```

378 \newcommand\dimmax@pq[2]
379 {%
380 \ifdim#1>#2
381 #1%
382 \else
383 #2%
384 \fi
385 }%

```

6.5 Image shapes

The following is an *experimental* application for including images with automatic determination of the *image shape*.

`\objrows@pq` “Grid row” count for object.


```

386      \newcount\objrows@pq

\objrowwd@pq  “Grid row” width for object.
387      \newdimen\objrowwd@pq

\@inputfile@pq  File handle for text representation of image shape.
388      \newread\@inputfile@pq

\pullquoteimgshape  \pullquoteimgshape[<includegraphics opts>]{<image name>}{<text>} will
typeset <Text> in two (balanced) columns, embedding the image <image name>
in the middle such that text ‘flows around’ the image. The “outer shape” of the
image is automatically determined and used as the shape of the cut-out area of
text.

389      \def\white@pq{white}
390      \def\black@pq{black}
391      \newcommand\@pullquoteimgshape@pq
392      {%
393          \setbox\objbox@pq=\hbox{\obj@pq}%
394          \objlines@pq
395          =\numexpr
396              \dimexpr\ht\objbox@pq+\dp\objbox@pq+2\objdist\relax
397              /\baselineskip
398          \relax
399          \windowhextent=\dimexpr.5\wd\objbox@pq+\objdist\relax
400          \windowvextent=\objlines@pq\baselineskip
401          \@tempdima
402          \dimexpr
403              \p* \wd\objbox@pq/\dimexpr.1\objdist\relax
404          \relax
405          \@tempdimb
406          \dimexpr
407              \p* \ht\objbox@pq/\dimexpr(\windowvextent-\ht\objbox@pq)/20\relax
408          \relax
409          \objrows@pq\numexpr2*\objlines@pq\relax
410          \objrowwd@pq\dimexpr2\windowhextent/\objrows@pq\relax
411          \edef\@tmp
412          {%
413              convert \img@pq\space -resize \strip@pt\@tempdima x\strip@pt\@tempdimb! -bordercolor whi
414              \number\objrows@pq x\number\objlines@pq! -black-threshold
415              95\@percentchar\space-monochrome \img@pq.pqshape.txt
416          }%
417          \immediate\write18{\@tmp}%
418          \typeout{\@tmp}%
419          \begingroup
420              \global\expandafter\let\csname pqshapemin:0@pq\endcsname\relax
421              \global\expandafter\let\csname pqshapemax:0@pq\endcsname\relax
422              \openin\@inputfile@pq \img@pq.pqshape.txt
423              \loop
424                  \unless\ifeof\@inputfile@pq

```

```

425         \read\@inputfile@pq to \@inputline@pq
426         \expandafter\expandafter\expandafter\analyse@gridline@pq
427         \expandafter\@inputline@pq\terminategridline@pq\@nil
428     \repeat
429     \closein\@inputfile@pq
430 \endgroup
431 \@pullquote@pq
432 }

```

\analyse@gridline@pq The shape analysis works by converting the image to a special text file with Image Magick. The helper macro \analyse@gridline@pq reads and dissects a line of this text file and builds the mapping for the shape function.

```

433 \edef\terminategridline@pq{,\space\space\space}
434 \long\def\analyse@gridline@pq#1,#2:#3) #4 #5 #6\@nil%
435 {%
436     \ifx\empty#4\empty%
437     \else%
438         \def\@tmp{#5}%
439         \ifx\@tmp\white@pq
440         \else
441             \ifnum#1>\objlines@pq
442                 \expandafter\gdef\csname pqshapemax:#2@pq\endcsname{#1}%
443             \else
444                 \expandafter\ifx\csname pqshapemin:#2@pq\endcsname\relax
445                 \expandafter\gdef\csname pqshapemin:#2@pq\endcsname{#1}%
446             \fi
447         \fi
448     \fi
449     \global\expandafter\let
450     \csname pqshapemin:\number\numexpr#2+\@ne @pq\endcsname\relax
451     \global\expandafter\let
452     \csname pqshapemax:\number\numexpr#2+\@ne @pq\endcsname\relax
453 \fi
454 }%

```

\imgshapefun \imgshapefun{<col>}{<starty>}{<endy>}{<line>} returns the mapping previously extracted from the image shape, determined by <col> and <line>.

```

455 \newcommand\imgshapefun[4]
456 {%
457     \expandafter\@imgshapefun\expandafter{\the\dimexpr\expandafter\@imgshapefun\expandafter{\
458 }

```

\@imgshapefun

```

459 \newcommand\@imgshapefun[2]
460 {%
461     \ifnum#2=\@ne
462         \expandafter\ifx\csname pqshapemin:#1@pq\endcsname\relax
463             \opt
464         \else

```

```

465         \the\dimexpr
466         \windowextent-
467         \csname pqshapemin:#1@pq\endcsname\objrowwd@pq
468         \relax
469     \fi
470 \else
471 \expandafter\ifx\csname pqshapemax:#1@pq\endcsname\relax
472     Opt
473 \else
474     \the\dimexpr
475     \numexpr\csname pqshapemax:#1@pq\endcsname+\@ne\relax\objrowwd@pq-
476     \windowextent
477     \relax
478 \fi
479 \fi
480 }

\@@imgshapefun
481 \newcommand\@@imgshapefun[1]
482 {%
483     \ifdim#1<.5\textcoldist.5\textcoldist\else#1\fi
484 }

```

7 Change History

1.0		replaced by environment	
	General: Converted to DTX file .. 1	pullquote. 20	
1.1		\pullquotecircular: Command	
	General: Added image shape recog-	\pullquotecircular replaced	
	nition 32	by environment pullquote with	
1.2		option shape=circular. 20	
	General: Now loading also graphicx ,	\pullquoteshape: Command	
	suggested by Andrew Stacey. . 19	\pullquoteshape replaced by	
2.0		environment pullquote with	
	General: New user interface (envi-	option shapefunction=. 20	
	ronment + KV options) 20	pullquote: Environment pullquote	
	\pullquote: Command \pullquote	introduced. 21	

8 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	\@circshapefun	\@currenvir 60
\@@imgshapefun 457, <u>481</u> 308, 325, <u>338</u>	\imgshapefun . 457, <u>459</u>

<code>\@inputfile@pq</code> . 388, 422, 424, 425, 429	E	L
<code>\@inputline@pq</code> 425, 427	<code>\end</code> 106	<code>\lineskiplimit</code> 214
<code>\@percentchar</code> 415	<code>environ</code> (package) . . . 3	<code>\linewidth</code> 71
<code>\@pullquote@pq</code>	environment options:	<code>lipsum</code> (package) 7
. 99, 110, 431	<code>image</code> 2, 12, 14, 16, 17	<code>\loop</code> 141, 423
<code>\@pullquoteimgshape@pq</code>	<code>imageopts</code> 12, 14	M
. 85, 391	<code>objdist</code> 12	<code>\maxdimen</code> . 115, 116, 214
A	<code>object</code> 12, 14	<code>microtype</code> (package) 3, 6
<code>\analyse@gridline@pq</code>	<code>objvalign</code>	<code>\microtype@pqfalse</code> . . 7
. 426, 433	. . 2, 12, 14, 16, 17	<code>\microtype@pqtrue</code> . . . 6
B	<code>objvoffset</code> . . . 12, 15	
<code>\baselineskip</code>	<code>shape</code> 2,	N
. 114, 125, 131,	12, 14, 15, 18, 20	<code>\narrowhsize@pq</code> . . .
134, 163, 164,	<code>shapefunction</code> 139, 181, 206
169, 185, 243, 12, 18, 20	<code>\newbox</code> 194, 195, 196, 197
244, 247, 259,	<code>textcolldist</code> . . 11, 12	<code>\newcount</code> 198,
260, 263, 397, 400	<code>textcolwd</code> 12, 13	199, 200, 201, 386
<code>\bbshapefun</code> 299	environments:	<code>\newdimen</code>
<code>\begin</code> 106	<code>pullquote</code> 7, 58	22, 24, 202, 203,
<code>\black@pq</code> 390	<code>etoolbox</code> (package) . . . 3	204, 205, 206, 387
<code>\BODY</code> 121, 157	G	<code>\NewEnviron</code> 58
C	<code>\gobbleparshapeprefix@pq</code>	<code>\newif</code> 5, 9
<code>\circshapefun</code> 303 287, 293	<code>\newread</code> 388
<code>\circularshapefun</code> . . 303	<code>graphicx</code> (package) . .	<code>noimageshapes</code>
<code>\clubpenalty</code> 112 3, 7, 8	(package option)
<code>\columnabx@pq</code>	I 3, 5, 7
. . . . 163, 191, 195	<code>\ifcsname</code> 32	<code>nomicrotype</code>
<code>\columnbbox@pq</code>	<code>\ifimgshades@pq</code> 9, 19, 79	(package option)
. . . . 164, 191, 196	<code>\ifmicrotype@pq</code> . . 5, 15 3, 6
<code>\constant@image@pq</code> .	<code>\ifnumequal</code> 233	O
. 48, 78	<code>\ifnumgreater</code> . . 222,	<code>\o@par@pq</code> 159, 284
<code>\constant@pullquote@pq</code>	225, 228, 230, 295	<code>\obj@pq</code> 42,
. 49, 60	<code>image</code> (environment option)	64, 75, 93, 119, 393
D	. . 2, 12, 14, 16, 17	<code>\objbox@pq</code> . 117, 130,
<code>\DeclareOption</code> . . . 7, 11	<code>imageopts</code> (environment option)	133, 186, 187,
<code>\define@key</code> 12, 14	189, 197, 393,
. 27, 28, 29, 30,	<code>\imageshapefun</code> 455	396, 399, 403, 407
41, 42, 43, 44, 45, 46	<code>\img@pq</code> . . 43, 65, 73,	<code>\objdist</code> 7,
<code>\dimmax@pq</code> 305, 378	76, 80, 413, 415, 422	24, 28, 130, 133,
<code>\do@valign@bottom@pq</code> 56	<code>\imgopts@pq</code> . . 44, 66, 76	181, 396, 399, 403
<code>\do@valign@center@pq</code> 53	<code>\imgshapefun</code> 455	<code>objdist</code> (environment option)
<code>\do@valign@middle@pq</code>	<code>\imgshades@pqfalse</code> . . 11 12
. 51, 62	<code>\imgshades@pqtrue</code> . . 10	<code>object</code> (environment option)
<code>\do@valign@top@pq</code> . . 55	<code>\includegraphics</code> . . 76 12, 14
	<code>\iterate@mkps@pq</code> 154, 220	<code>\objlines@pq</code> . 52, 54,
	K	57, 128, 134, 184,
	<code>keyval</code> (package) 3	200, 230, 394,
		400, 409, 414, 441

<code>\objrows@pq</code>	<code>\prevgraf</code> 285, 286	<code>textcoldist</code> (environment option) 11, 12
. 386, 409, 410, 414	<code>\ProcessOptions</code> 13	
<code>\objrowwd@pq</code>	<code>\pullquote</code> 22, 104	
. 387, 410, 467, 475	<code>pullquote</code> (environ- ment) 7, 58	<code>\textcolwd</code> 29, 61, 70, 71, 140, 155, 191, 202, 212, 231, 238, 271, 280
<code>\objtopoffset@pq</code> 143, 184, 201, 228, 230, 243, 247, 248, 259, 263, 264	<code>pullquote</code> (package) 3, 6, 7	
<code>objvalign</code> (environment option) 2, 12, 14, 16, 17	<code>\pullquotecircular</code> . 22	<code>textcolwd</code> (environment option) 12, 13
<code>\objvalign@pq</code> 33, 62, 146	<code>\pullquoteimgshape</code> . 389	<code>\tolerance</code> 213
<code>objvoffset</code> (environment option) 12, 15	<code>\pullquoteshape</code> 22	<code>\typeout</code> 142, 418
<code>\objvoffset@pq</code>		<code>\typesettext@pq</code> 121, 157, 208
. 41, 63, 145, 148	R	U
	<code>\raise</code> 182	<code>\unless</code> 165, 424
	<code>\rectangularshapefun</code> 68, 299	V
	<code>\RequirePackage</code>	<code>\vbadness</code> 115
 1, 2, 3, 16, 20	<code>\vfuzz</code> 116
	<code>\right@psexpr@pq</code> 253, 277	<code>\vsplit</code> 163, 164
	<code>\rlap</code> 178, 191	
	S	W
	<code>\setkeys</code> 69	<code>\white@pq</code> 389, 439
	<code>shape</code> (environment option) 2, 12, 14, 15, 18, 20	<code>\widowpenalty</code> 113
	<code>\shape@pq</code> 45, 46, 67, 78	<code>\windowhextent</code> 133, 136, 137, 140, 203, 301, 399, 410, 466, 476
	<code>\shapefun@pq</code> 45, 46, 68, 240, 256	<code>\windowqhexent</code> 135, 136, 137, 204, 341, 342, 344, 345, 347, 351, 352, 354, 356, 358, 364, 365, 367, 369, 371, 373, 375
	<code>shapefunction</code> (environment option) 12, 18, 20	<code>\windowvextent</code> 134, 135, 186, 205, 311, 312, 314, 328, 329, 331, 400, 407
	<code>\splittopskip</code> 114	
	<code>\strut</code> 217	
	<code>\strutbox</code> 114	
	T	
	<code>\terminategridline@pq</code> 427, 433	
	<code>\textbox@pq</code> 124, 163, 164, 165, 168, 194, 210	
	<code>\textcoldist</code> 7, 22, 27, 71, 140, 191, 239, 265, 483	
<code>\par</code> 159, 160		
<code>\par@pq</code> 160, 282		
<code>\parshape</code> 161, 288		
<code>\parshape@pq</code> 151, 161, 207, 287, 288		
<code>\parshapelines@pq</code> 150, 153, 199, 285, 286, 290, 295		
<code>\parskip</code> 215		
<code>\pqlines@pq</code> 52, 54, 57, 122, 142, 150, 163, 164, 174, 184, 198, 222, 225		