Framed material for humanists

Claudio Beccari

Abstract Humanists sometimes need to emphasise some small pieces of text or small images so that it is possible to comment them in the proper way; they can use the wrapfig2 package that has been recently upgraded in order to be compatible with the modifications of the *picture* environment that accept coordinates in absolute dimensions, instead as multipliers of `\unitlength` as it has been normal since LaTeX was made available in 1984.

Sommario Gli umanisti talvolta necessitano di evidenziare dei brevi brani di testo o piccole immagini in modo che sia possibile commentarli adeguatamente; essi possono usare il pacchetto wrapfig2, che recentemente è stato reso compatibile con l'ambiente *picture* che accetta coordinate definite in modo assoluto, invece che come coefficienti di `\unitlength`, come è stato normale fin da quando nel 1984 il linguaggio LaTeX è diventato disponibile.

1. Introduction

Recently wrapfig2 has been upgraded in order to avoid the use of package curve2e that does not accept the use of coordinates as phisical quantities, instead of plain rational numbers, used as coefficients of the `\unitlength` unit of measure. That was normal with the previous versions of the *picture* updated with the extension pict2e now updated in 2020 so as to be compatible with the LaTeX Team innovations concerning the official LaTeX tools.

We think that now humanists can exploit the new wrapfig2 functionalities for their documents that, more often than not, need to strongly emphasise some phrases or short paragraphs, or small images in a comfortable way.

Here we show how to use the *wraptext* environment and the `\includeframedtext` command.

2. The *wraptext* environment

The *wraptext* environment inserts a small framed text into the normal text lines that are shaped so that the block containing the framed text is conveniently wrapped, even if this wrapping involves more than a single paragraph.

Κα᾽γὼ σὲ πατάξας διαλύσω τὸ κρανὶον

Text 1. A short phrase from Lucianus

This wrapping sometimes is not successful, because of some idiosyncrasies due to the wrapfig package by Donald Aresenau; wrapfig2 is a fork of Arsenau's one. The difference is that the former package defines only the *wrapfigure* and *wraptable* environments, while the latter package adds the *wraptext* environment with a similar philosophy. Users can read the wrapfig2 documentation by means of

the terminal command `texdoc wrapfig2`, since this new package is part of any TEX system installation.

A small sample of use of this package is in the internal page 73 side; it is also possible to insert the block in the external side, as well to let the text protrude in the margin; as a personal taste we prefer to avoid any protrusion. Wrapped text may be declared to be a floating object. More details are available in the package documentation.

The code for the above example 1 is the following.

```
\begin{wraptext}{i}{0.5\linewidth}
\includeframedtext{%
\foreignlanguage{greek}{Κα'γὼ σὲ πατάξας διαλύσω τὸ κρανἰον}}
\caption{A short phrase from Lucianus}\label{fig:greek}
\end{wraptext}
```

We call the reader's attention on the command `\includeframedtext` that allows also the use of several options expressed with the *key=value* syntax in order to specify up to nine setting options.

3. The `\includeframedtext` command

The `\includeframedtext` command is the one that actually inserts its content into the output PDF file. This command may be used even outside the *wraptext* environment, and, although it is not necessary, it might be used even within the other two environments *wrapfigure* and *wraptable*. But this is the command that may receive the *key=value* settings; in facts its syntax is the following:

`\includeframedtext{`⟨*text to frame*⟩`}[`⟨*key=value settings*⟩`]`

where the ⟨*key=value settings*⟩ are the following (listed in alphabetical order, but they can be used in any order).

`backgroundcolor` sets the background colour among those defined by the default set provided by package xcolor. The default colour is light grey.

`fboxrule` sets the thickness of the frame; a zero value is allowed, otherwise it should not be smaller than `0.4pt`; on the opposite it should not be set too large and `1mm` appears as a thick enough frame around the wrapped text.

`fboxsep` sets the distance of the frame from the wrapped text; by default it is set to `1ex`; also in this case it is better to avoid exaggerations. Notice that the default value depends on the wrapped text font x-height.

`fontstyle` sets any available *declaration*[1] that changes the characteristics of a font: size, series, shape; it is possible to use also the `\usefont` command with all its four arguments, even

1. Remember the difference between *command* and *declaration*; the former either prints a default string, or operates only on its arguments and its effect stops immediately; the latter is a macro that may have arguments, but operates on its possible arguments, and its effects last for a whole group or for the whole document. If `\itshape` is specified, it is a declaration, while `\textit{`⟨*text*⟩`}` is a command that operates only on its argument ⟨*text*⟩.

the font encoding. The wrapfig2 package has available also the `\setfontsize` command that can select any size with any font that has available at least a stepwise continuous size set; for example, the Latin Modern fonts have a stepwise continuous size set, while Computer Modern have available only a discrete size set.

`framecolor` sets the color of the frame; the colours available are those available with package xcolor to which no options have been specified; see its documentation and in case load xcolor with the desired options before this package wrapfig2. The default colour is a very dark grey.

`insertionwidth` sets the insertion width; as it was previously specified, if this width is too small or too large it will be automatically reassigned a value within the allowed range.

`radius` sets the optional radius of the frame rounded "corners"; if it is not specified, such radius is equal to the default value of `\fboxsep`.

`scalefactor` sets the value that establishes a reasonable range for the insertion width; users can specify any value in the range $xy_0 = y_{\min} \leq y \leq y_{\max} = y_0/x$, where y_0 is the default value, and x is the scaling factor that by default equals 0.8; this means that if y_0 equals half the current measure, the inserted wrapped text produces an indentation of the wrapping lines approximately between 60% and 40% of the current measure; the wrapped text should never have too short a measure and the wrapping indented lines never have too short a measure. If users specify a different value to this key, they might get problems with inter word spacing and with hyphenation.

`textcolor` sets the text colour among those available with the default set provided by package xcolor. The default colour is black.

All settable quantities have default values, specified in the above list. The example is in figure 1.

All this means that a single command can do several different things on a given text, depending on the various combinations set up by the user. It may be recalled that the object to be wrapped may be anything, not only text; if instead of a ⟨*text*⟩ a suitable image is entered, the command provides to set everything necessary to frame everything; by specifying a zero frame thickness, the object is not framed and if the background color is set to white, the object to frame is not emphasised at all.

The code for typesetting figure 1 is the following.

```
\begin{figure}
% first line
\makebox[\textwidth]{\includeframedtext{Text}%
  [insertionwidth=0.45\linewidth]
\hfill
\includeframedtext{Text}%
  [insertionwidth=0.45\linewidth,fboxrule=1mm]}\\[2ex]
% Second line
\makebox[\textwidth]{\includeframedtext{Text}%
  [insertionwidth=0.45\linewidth,textcolor=red,fboxrule=2pt]
\hfill
\includeframedtext{Text}%
```

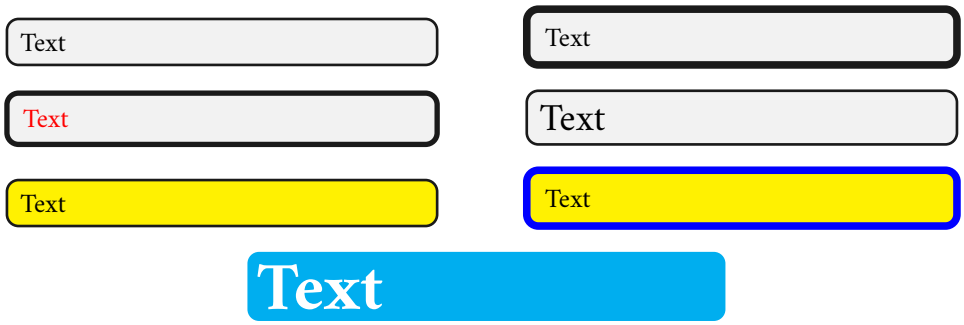

Figure 1. Some framed text boxes with different dimensional parameters, different font size, and different colours.

```
  [insertionwidth=0.45\linewidth,fontstyle=\Large]}\\[2ex]
% Third line
\makebox[\textwidth]{\includeframedtext{Text}%
  [insertionwidth=0.45\linewidth,backgroundcolor=yellow]
\hfill
\includeframedtext{Text}%
  [insertionwidth=0.45\linewidth, framecolor=blue,
  backgroundcolor=yellow, fboxrule=1mm]}\\[2ex]
% Fourth line
\makebox[\textwidth]{\includeframedtext{Text}%
  [insertionwidth=0.5\linewidth, fboxrule=0pt,
  backgroundcolor=cyan, textcolor=white,
  fontstyle=\Huge\bfseries]}

\caption{Some framed text boxes with different dimensional
 parameters, different font size, and different colours}\label{fig:framed text}
\end{figure}
```

Notice the all framed texts have a width of `0.45\linewidth`, so that, where there are two framed texts in a line, there is enough space between them to fill an `\hfill` so as to push them at the line extrema.

Notice that the default background color is a very light grey one, the default setting, that discreetly emphasises the wrapped text; nevertheless if the frame thickness was too large, it would become an exaggeration.

In the second line a framed text is coloured in red, while the other frame contains the same text in `\Large` size.

In the third line both frames have a yellow background and both have a blue frame; may be the yellow frame, although being a light colour, is too much, but both contain contrasting colours that match well. If, for example, those colours were a blue frame and a dark brown background, there would not be enough contrast.

The fourth line displays a white text on a cyan background, but there is no frame. The result is fine, but it might be used for other purposes than emphasising a test to be commented, for example, in a critical edition. May be it is good for a title, but certainly not for a wrapped text.

Figure 2. Two framed images. On the left the Todi stela written in Gallic and Latin;Gregorian Etruscan Museum in Rome. On the right a color wheel from Wikipedia.

The curved frame corners have a visible radius change that matches the size of the used font. On the opposite, in figure 2 there is no text, therefore the radius of the corner arcs has been specified.

Speaking of contrasting colours it is possible to choose from the colour wheel,[2] in figure 2. Contrasting colours are at the end points of each diameter. In any case colours can be selected even by using the "colour expressions" of package xcolor.

As an example of framing an object that is not a real ⟨*text*⟩ see figure 2, wich is typeset with the following code.

```
\begin{figure}
\makebox{\textwidth}{%
%
\includeframedtext%
  {\includegraphics[width=\hsize]{stele-todi-small}}%
  [fboxsep=1em, fboxrule=1mm, framecolor=blue,
   backgroundcolor=white,radius=1em,
   insertionwidth=0.45\linewidth]
%
\hfill
%
\includeframedtext}%
  {\includegraphics[\width=\hsize]{ColorWheel}}
[fboxsep=1em, fboxrule=1mm, framecolor=blue,
   backgroundcolor=white,radius=1em,
   insertionwidth=0.45\linewidth]
\caption{Two framed images. On the left the Todi stela written
```

2. From Wikipedia.

```
  in Gallic and Latin;Gregorian Etruscan Museum in Rome. On the
  right a color wheel from Wikipedia}
\label{txt:todi-stela}\end{figure}
```

4. Conclusion

The possibility of framing small texts, single words, or short phrases, even short paragraphs, may be useful even for humanists; as a matter of fact the whole wrapfig2 package was assembled explicitly on request of a humanist who wanted to emphasise some short phrases.

The original wrapfig package by Donald Arseneau did not contain anything that could frame what was wrapped; Arseneau had already developed another package, framed, but his package created just a frame with right angle corners.

On `tex.stackechange` another user asked for a similar solution and he was pleased to use the framed coloured box of package tcolorbox. But the former humanist who asked us a simpler solution, could not use tcolorbox that is too heavy for what concerns PC memory requirements; the curious reader who wants to examine the inner workings of wrapfig2 can see that the macro that draws the frame with curved corners is about a dozen lines long, and there are no problems with any PC working memory.

Certainly these functionalities can be done much better by more experienced "TEXperts"; we hope that in the future better solutions can come.

Claudio Beccari
Rivoli, TO, Italy
claudio(dot)beccari(at)gmail(dot)com