The `lroundrect` Package, v1.0

Donald P. Goodman III

February 19, 2016

Abstract

The `lroundrect` package is the LaTeX companion for the `roundrect` macros for METAPOST; they provide an easy interface for a few specific formats, and a good example for how to write your own rounded-rectangle macros for use in LaTeX documents or elsewhere. The idea with this package and with the METAPOST `roundrect` macros was to provide a METAPOST-based replacement for the incredibly versatile `tcolorbox` package; this package is far from achieving that goal. But it is nevertheless extremely useful.

Contents

1 Introduction

While Ti*k*Z and its many accompanying packages, particularly `tcolorbox`, are wonderful and powerful tools, whenever using them I inevitably feel completely lost, and I exert great effort doing comparatively simple things. Contrariwise, thanks to my experience with the `drm` and `dozenal` packages, writing in METAPOST is quite straightforward for me. So I decided to try to write some generalized macros to provide functionality similar to that of `tcolorbox`. This package (along with `roundrect`, its accompanying METAPOST package) is not even close to that kind of flexibility or power, but it's still quite useful and versatile, so I make it available for anyone who might be interested.

This document was typeset in accordance with the `docstrip` utility, which allows the automatic extraction of code and documentation from the same document.

2 Prerequisites and Conventions

Some prerequesites for using this package are METAPOST itself (obviously). If you're using the package with LaTeX, the `gmp` package would probably be helpful; be sure to use the `latex` package option. Finally, the package internally calls `TEX.mp`, so that is also required. All of these should be packaged in any reasonably modern LaTeX system, such as TeXLive or MikTeX.

This documentation assumes nothing about your personal TeX or METAPOST environment. ConTeXt and the various forms of LuaTeX have METAPOST built-in; with pdfLaTeX, the author's choice, one can use the `gmp` package to include the source directly in one's document (that's what's been done in this documentation) or develop a simple script to compile them afterwards and include them in the source via `\includegraphics` (probably the quickest option, since compilation is done in advance). Here, we simply post the plain vanilla METAPOST code, and let you work out those details however you prefer.

3 Basic Usage

`lroundrect` utilizes the METAPOST `roundrect` macros to perform typesetting within a LaTeX document. To do this, it has a number of predefined styles which can be selected. These styles consist of different names and, often, different numbers of arguments. Here is the simplest, `\rralertbox`:

`\rralertbox`

```
\rralertbox{\unexpanded{\Huge This is an rralertbox!}}
```

yields the following:

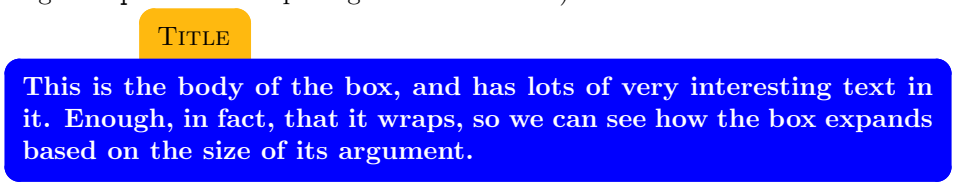
The `\unexpanded` is simply to protect the font commands when they are used with the `gmp` package in pdfLaTeX; it is not necessary otherwise. `\rralertbox` is produced with the following code:

```
\def\rralertbox#1{%
\begin{mpost}
rrborderpen(pencircle scaled 3);
rrbodytext := "#1";
rrbordercolor(0.8red);
rrtextcolor := 0.8red;
rrinnercolor := 0.5[red,white];
roundrect(1in,2in)(alertbox);
draw alertbox;
\end{mpost}
}%
```

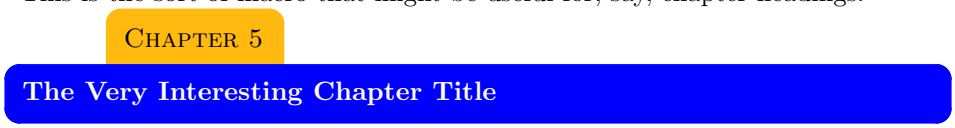
It's not claimed that this particularly good style; it serves merely as an example of what we can do.

\rrorangebluebox

Another simple example (and likely equally ugly color-wise) is `\rrorangebluebox` (using the `mpcolornames` package for color names):



This is the sort of macro that might be useful for, say, chapter headings:



This provides an interesting way to display things, particularly items like headings and definitions. Both the title and the body will expand to fit their contents as necessary, though if the title is too long its positioning won't work well. (By "title" I mean the orange box.) The code producing these boxes is below.

```
\newlength{\rrorangewid}
\newdimen\rrblueht
\newdimen\rrorangeht
\newbox\rrbluebox
\newbox\rrorangebox
\def\rrorangebluebox#1#2{%
\settowidth{\rrorangewid}{\ \scshape #1\ }
\savebox\rrbluebox{\parbox{\linewidth}{#2}}
\advance\rrorangewid by6pt
\savebox\rrorangebox{\parbox{\rrorangewid}{#1}}
\rrorangeht=\ht\rrorangebox
\advance\rrorangeht by\dp\rrorangebox
\advance\rrorangeht by\baselineskip
\rrblueht=\ht\rrbluebox
\advance\rrblueht by\dp\rrbluebox
\advance\rrblueht by\baselineskip
\begin{mpost}
input mpcolornames;
rrborderrad(10pt);
rrbotlftborderrad := 0pt;
rrbotrtborderrad := 0pt;
rrbordercolor(DarkGoldenrod1);
rrinnercolor := DarkGoldenrod1;
rrtextalign := "\unexpanded{\raggedright}";
rrtextfont := "\unexpanded{\scshape}";
rrbodytext := "#1";
roundrect(\mpdim{\rrorangeht})(\mpdim{\rrorangewid})(orangebluebox);
rrrestorevals;
rrborderrad(10pt);
```

```
rrbordercolor(blue);
rrinnercolor := blue;
rrbodytext := "#2";
rrtextcolor := white;
rrtextfont := "\unexpanded{\bfseries}";
rrtextalign := "";
roundrect(\mpdim{\rrblueht})(\mpdim{\linewidth})(mainbox);
draw orangebluebox shifted (-\mpdim{0.3\linewidth},0);
draw mainbox shifted (0,-\mpdim{\rrblueht}/2-\mpdim{\rrorangeht}/2);
\end{mpost}
}%
```

As we can see, the hardest thing about this macro is ensuring that the heights and widths are correct; that done, all is really rather simple.

We can also imitate the sort of boxes that `tcolorbox` produces by default with `\rrdefaultbox`:

\rrdefaultbox

Title

This is the body of the box, and has lots of very interesting text in it; so much, in fact, that it will have to be wrapped, thus displaying the way that this macro produces boxes which will expand appropriately according to their contents. Even more, we'll *put in some font changes* to show that we can STYLE OUR TEXT without causing trouble.

The code producing these boxes is here:

```
\newlength{\rrtitlewid}
\newlength{\rrtitleht}
\newlength{\rrmainht}
\newbox\rrtitlebox\newbox\rrmainbox
\def\rrdefaultbox#1#2{%
\settowidth{\rrtitlewid}{\ \scshape #1\ }
\savebox\rrmainbox{\parbox{\linewidth}{#2}}
\advance\rrtitlewid by6pt
\savebox\rrtitlebox{\parbox{\rrtitlewid}{#1}}
\rrtitleht=\ht\rrtitlebox
\advance\rrtitleht by\dp\rrtitlebox
\advance\rrtitleht by\baselineskip
\rrmainht=\ht\rrmainbox
\advance\rrmainht by\dp\rrmainbox
\advance\rrmainht by\baselineskip
\begin{mpost}
input mpcolornames;
rrborderrad(10pt);
rrbotlftborderrad := 0pt;
rrbotrtborderrad := 0pt;
rrbodytext := "#1";
roundrect(\mpdim{\rrtitleht})(\mpdim{\linewidth})(titlebox);
rrrestorevals;
```

```
rrborderrad(10pt);
rrtoplftborderrad := 0pt;
rrtoprtborderrad := 0pt;
rrtopborderpen := pencircle scaled 1;
rrbodytext := "#2";
rrtextalign := "";
roundrect(\mpdim{\rrmainht})(\mpdim{\linewidth})(mainbox);
draw titlebox;
draw mainbox shifted (0,-\mpdim{\rrmainht}/2-\mpdim{\rrtitleht}/2);
\end{mpost}
}
```

Remember, if you are including these in a (pdf)LaTeX document using `gmp`, you need to wrap your font changing commands in `\unexpanded`, or LaTeX will give METAPOST a bunch of gobbledygook that it can't understand. `\unexpanded` ensures that METAPOST gets something it can work with. With other methods of inclusion, however, `\unexpanded` is not necessary.

`lroundrect` also provides a box similar to that used in the `tcolorbox` documentation for examples:

```
\rrtcolorex{\texttt{$\backslash$tcbuselibrary{listings,theorems}}}
```

\tcbuselibrarylistings,theorems

The following code produces these boxes (and assumes that the contents will be only one line):

```
\def\rrtcolorex#1{%
\begin{mpost}
rrborderrad(10pt);
rrbordercolor(blue);
rrtopbordercolor := 0.5[green,black];
rrinnercolor := 0.8[blue,white];
rrbodytext := "#1";
roundrect(2\mpdim{\baselineskip},\mpdim{\linewidth})(rrexampbox);
draw rrexampbox;
\end{mpost}
}%
```

`\rrtcoltag`

The tags from the `tcolorbox` documentation can also be closely reproduced using the `\rrtcoltag` macro:

```
\rrtcoltag{LIB}{skins}
\rrtcoltag{ALT}{rasterized}
\rrtcoltag{DEF}{wow, this one is really long}
```

LIB skins ALT rasterized DEF wow, this one is really long

This is one of the more complicated definitions in this little package; but it works well when the first argument is short and the second is limited to one line.

```
\newdimen\rrtcoltagwd
\newbox\rrtcoltagbox
\def\rrtcoltag#1#2{%
\settowidth{\rrtcoltagwd}{#2}
\advance\rrtcoltagwd by12pt
\begin{mpost}
input mpcolornames;
rrborderrad(5pt);
rrbotlftborderrad := 0pt;
rrbotrtborderrad := 0pt;
rrnobot := true;
rrinnercolor := 0.5[green,white];
rrbodytext := "#1";
rrtextfont := "\unexpanded{\sffamily}";
rrtextcolor := Khaki1;
roundrect(\mpdim{1.5em},2.5\mpdim{\baselineskip})(rrtaglabel);
rrrestorevals;
rrborderrad(5pt);
rrtoplftborderrad := 0pt;
rrbotlftborderrad := 0pt;
rrinnercolor := 0.7[green,white];
rrnolft := true;
rrbodytext := "#2";
roundrect(2.5\mpdim{\baselineskip},\mpdim{\rrtcoltagwd})(rrtextlabel);
draw rrtaglabel rotatedaround((0,0),90);
draw rrtextlabel shifted (\mpdim{\rrtcoltagwd}/2+0.625\mpdim{\baselineskip},0);
\end{mpost}
}%
```

4 Using metafun Extensions

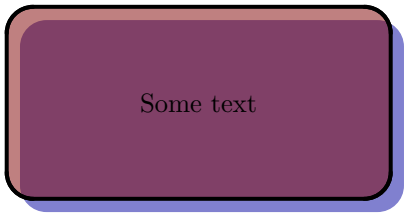
We can also create interesting effects with drop shadows and transparency, if we're willing to use `metafun` macros. Note that these won't work with `gmp` by default, which includes Metapost `.mps` files rather than `.pdf`; transparency effects require conversion to `pdf` first. However, `gmp` does perform conversion to `pdf` when the engine is XeTEX, so it's certainly possible; we simply need to redefine `\gmp@innermpost` to run `mptopdf` and include the `pdf` rather than the `mps`.

`\rrincludepdf` `lroundrect` provides the `\rrincludepdf` macro, which does all this for you. Just issue:

```
\rrincludepdf
```

After doing this, you can use all the `metafun` tricks you want, including transparency, and the resulting `pdf` will be included rather than the `mps`. `\rrtransbox` takes a single argument, the text you want included, and it contains oddly-colored transparent `rrinnercolor` and `rrshadowcolor`s:

```
\rrtransbox{Some text}
```



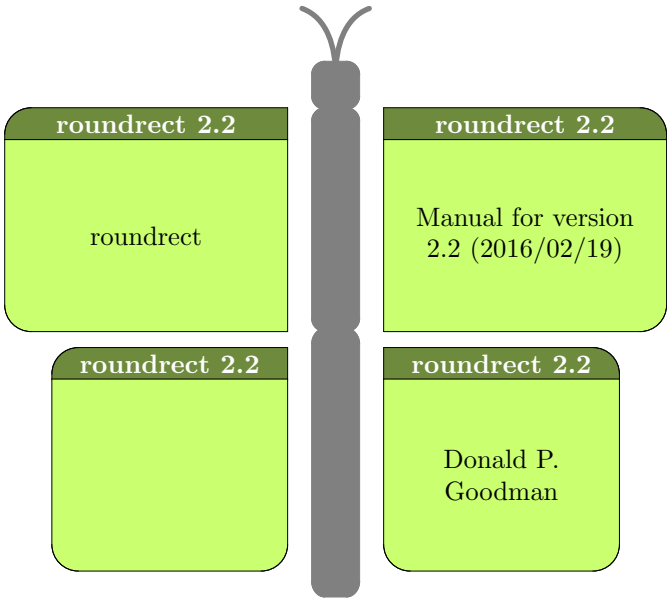
The code which produces this is:

```
\def\rrtransbox#1{%
\begin{mpost}[mpmem=metafun]
rrbodytext := "#1";
rrdropshadow := true;
rrborderrad(20pt);
rrinnercolor := transparent(0.5,0.5,0.5red);
rrshadowcolor := transparent(0.5,0.5,0.625blue);
rrshadowx := rrbotlftborderrad/4;
rrshadowy := -rrbotlftborderrad/4;
roundrect(1in,2in)(rectangle);
draw rectangle;
\end{mpost}
}%
```

The transparency here is interesting, if oddly colored; and this serves as a good example of what can be done with judicious use of shadows and transparency.

`\rrincludemps` To turn off the inclusion of `pdf`s (including them rather than `mps` does slow down compilation, since `mptopdf` needs to be run), issue `\rrincludemps`, and `mps` files will be included as before.

Using transparency from `metafun` enables us to do some really impressive things. For example, the logo from the cover of the `tcolorbox` manual, modified to encompass `roundrect` (almost the same, anyway; `roundrect` still doesn't really support gradients, and I picked different colors):



This "butterfly" logo is quite similar (again, but for different colors and the lack of gradients), and it's got the advantage that the antennae are actually curved, since nicely curved lines are so much easier to do in METAPOST than in Ti*k*Z. This is a nice demonstration of the power that `roundrect` puts at your fingertips.

```
\def\rrincludepdf{%
 \long\def\gmp@innermpost##1\end##2{%
\ifgmp@nowrite\else
\gmp@write\gmp@out{%
##1^^Jendfig;^^Jend.%
\gmp@doiflatex{^^Jverbatimtex^^J\string\end{document}^^Jetex}}%
\gmp@closeout\gmp@out
\count@=\gmp@runs
\loop\ifnum\count@>\z@
\gmp@shellcommand{\gmp@command\space
-tex=\gmp@mpxprogram\space\gmp@jobname\gmp@ext
\gmp@fourdigits{\gmp@number}}%
\advance\count@\m@ne
\repeat
\ifxetex
\gmp@shellcommand{epstopdf --hires
\gmp@jobname\gmp@ext
\gmp@fourdigits{\gmp@number}.mps}%
\fi
\gmp@shellcommand{mptopdf
--result=\gmp@jobname\gmp@ext
\gmp@fourdigits{\gmp@number}.pdf
\gmp@jobname\gmp@ext
\gmp@fourdigits{\gmp@number}.mps}%
\gmp@shellcommand{mv
```

```
\gmp@jobname\gmp@ext
\gmp@fourdigits{\gmp@number}-mps.pdf
\gmp@jobname\gmp@ext
\gmp@fourdigits{\gmp@number}.pdf}%
\egroup
\end{##2}\ifgmp@use
\gmp@usempost{\thegmp@count}\fi}
  \def\gmp@usempost##1{%
    \edef\gmp@thempsfile{\gmp@jobname\gmp@ext\gmp@fourdigits{##1}}%
    \IfFileExists{\gmp@thempsfile.pdf}%
    {\includegraphics{\gmp@thempsfile.pdf}}%
    {\gmp@nemessage\gmp@box}}
  \renewcommand\usempost[2][]{%
    \IfFileExists{\gmp@jobname\gmp@ext\csname gmp@fig##2\endcsname.pdf}%
    {\includegraphics[##1]
      {\gmp@jobname\gmp@ext\csname gmp@fig##2\endcsname.pdf}}%
    {\gmp@nemessage\gmp@box}}
}%
\def\rrincludemps{%
\long\def\gmp@innermpost##1\end##2{%
  \ifgmp@nowrite\else
    \gmp@write\gmp@out{%
      ##1^^Jendfig;^^Jend.%
      \gmp@doiflatex{^^Jverbatimtex^^J\string\end{document}^^Jetex}}%
    \gmp@closeout\gmp@out
    \count@=\gmp@runs
    \loop\ifnum\count@>\z@
      \gmp@shellcommand{\gmp@command\space
        -tex=\gmp@mpxprogram\space\gmp@jobname\gmp@ext
        \gmp@fourdigits{\gmp@number}}%
    \advance\count@\m@ne
    \repeat
    \ifxetex
      \gmp@shellcommand{epstopdf --hires \gmp@jobname\gmp@ext
        \gmp@fourdigits{\gmp@number}.mps}%
    \fi
  \fi
  \egroup
  \end{##2}\ifgmp@use
  \gmp@usempost{\thegmp@count}\fi}
\ifxetex
  \def\gmp@usempost##1{%
    \edef\gmp@thempsfile{\gmp@jobname\gmp@ext\gmp@fourdigits{##1}}%
    \IfFileExists{\gmp@thempsfile.pdf}%
    {\includegraphics{\gmp@thempsfile.pdf}}%
    {\gmp@nemessage\gmp@box}}
  \renewcommand\usempost[2][]{%
    \IfFileExists{\gmp@jobname\gmp@ext\csname gmp@fig##2\endcsname.pdf}%
    {\includegraphics[##1]
      {\gmp@jobname\gmp@ext\csname gmp@fig##2\endcsname.pdf}}%
```

```
    {\gmp@nemessage\gmp@box}}
\else
  \def\gmp@usempost##1{%
    \edef\gmp@thempsfile{\gmp@jobname\gmp@ext\gmp@fourdigits{##1}}%
    \IfFileExists{\gmp@thempsfile.mps}%
    {\includegraphics[hiresbb]{\gmp@thempsfile.mps}}%
    {\gmp@nemessage\gmp@box}}
\renewcommand\usempost[2][]{%
  \IfFileExists{\gmp@jobname\gmp@ext\csname gmp@fig##2\endcsname.mps}%
  {\includegraphics[hiresbb,##1]
    {\gmp@jobname\gmp@ext\csname gmp@fig##2\endcsname.mps}}%
  {\gmp@nemessage\gmp@box}}
\fi
}%
```