The \texttt{roundrect} Macros, v2.1

Donald P. Goodman III

August 7, 2015

Abstract

The \texttt{roundrect} macros for \textsc{MetaPost} provide extremely configurable, extremely versatile rectangles (including rounded corners), intended primarily for inclusion in documents produced by \TeX{} and friends. The idea was to provide a \textsc{MetaPost}-based replacement for the incredibly versatile \texttt{tcolorbox} package; the macros are far from achieving that goal. But they are nevertheless extremely useful.

Contents

1 Introduction 1
2 Prerequisites and Conventions 2
3 Shapes and Styles 2
4 Coloring the Parts 4
5 Drop Shadows 6
6 Including Text 8
7 Implementation 9

1 Introduction

While \textsc{TikZ} and its many accompanying packages, particularly \texttt{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 \texttt{drm} and \texttt{dozenal} packages, writing in \textsc{MetaPost} is quite straightforward for me. So I decided to try to write some generalized macros to provide functionality similar to that of \texttt{tcolorbox}. It’s 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 prerequisites 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. Con`TEXt` and the various forms of `LuaTEX` have `METAPOST` built-in; with `pdfTEX`, 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 \texttt{\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 Shapes and Styles

### `roundrect`

The core of all the action is the `roundrect` macro; this will set up your rounded rectangle in the plainest way possible. The first argument is the box’s height, the second its width, and the third its name, by which you will draw it later:

```plaintext
roundrect(1in,2in)(rectangle);
draw rectangle;
```

All the corners don’t have to be rounded; we can make them square if we want. To do things like this, we use the macro `rrborderrad()`, which takes a single argument giving the border radius we want; that is, how rounded we want the corners of our rectangle. Higher values will be more rounded, lower values will be less:

---

### `rrborderrad()`

---
Notice that the corners in this, with \texttt{rrborderrad()} set to 10pt, are much less rounded than the previous example. The default border radius is 40pt, which is quite rounded.

\texttt{rrborderrad()} provides an easy way to set the border radius of all four corners at once; however, we can also control each corner individually, with \texttt{rrtoplftborderrad}, \texttt{rrbotlftborderrad}, \texttt{rrtoprtborderrad}, and \texttt{rrbotrtborderrad}, which are parameters rather than macros; that is, we define them using := rather than as an argument in parentheses:

\begin{verbatim}
rrtoplftborderrad := 20pt;
rrbotlftborderrad := 40pt;
rrtoprtborderrad := 10pt;
rrbotrtborderrad := 60pt;
roundrect(1in,2in)(rectangle);
draw rectangle;
\end{verbatim}

As you can see, this makes it possible to create a large variety of shapes, including the ability to arbitrarily flatten any side of the rectangle desired just by setting the border radius of the appropriate corners to 0pt:
Here, we've flattened the top border by setting the top right and top left corners' border radii to \textit{0pt}. This ability to flatten any given side of the rectangle makes it much easier to combine multiple rectangles into interesting forms, which we'll see a bit more about later.

4 Coloring the Parts

The colors of the \texttt{roundrect} are extremely configurable, both on the whole and for individual parts. The background color of the \texttt{roundrect} is controlled by \texttt{rrinnercolor}, while the border is colored by \texttt{rrbordercolor()}.

```latex
rrinnercolor := red;
rrbordercolor(blue);
roundrect(1in,2in)(rectangle);
draw rectangle;
```

By default, \texttt{rrinnercolor} is white and \texttt{rrbordercolor} is black. Notice that \texttt{rrbordercolor} is a \textit{macro}, not a parameter; that’s because each border can be individually colored, and this macro simply does all of them at once. We’ll see more about that later.

You can also completely suppress the border by using \texttt{rrnotop}, \texttt{rrnobot}, \texttt{rrnolft}, and \texttt{rrnort}, which is particularly useful when you want to combine multiple rectangles without making an obvious border between them. You can combine these in any way you like:
Here we’ve squared all the corners to make it easier to see what’s going on. Each border can be colored individually and separately from the others, using the commands you’d expect:

There is obviously some difficulty in determining what part of each rounded corner should be colored how; this ability is typically more useful with a single, flattened side, to help it blend in better when combined with other constructs:
Perhaps you don’t like the border; you’d like it thicker, or drawn with a square rather than a circular pen. You’re in luck; \texttt{rrborderpen()} takes the single argument of the pen you’d like to draw the border with, defined like any other \textsc{MetaPost} pen:

\begin{verbatim}
rrborderpen(pensquare scaled 3);
roundrect(1in,2in)(rectangle);
draw rectangle;
\end{verbatim}

The default border pen is \texttt{pencircle scaled 1.5}, so this results in a square pen rather than a circular one, twice as thick. You can also use individual pens for each border, as expected:

\begin{verbatim}
rrbotlftborderrad := 0pt;
rrbotrtborderrad := 0pt;
rrbotbordercolor := green;
rrbotborderpen := pensquare yscaled 6;
roundrect(1in,2in)(rectangle);
draw rectangle;
\end{verbatim}

Here we’ve flattened the bottom border, colored it green, and drawn it with a square pen scaled on the y-axis only by 6. Clearly, there are huge possibilities here.

5 Drop Shadows

\texttt{rrdropshadow} We can also put a shadow on the boxes using \texttt{rrdropshadow}, a boolean value which defaults to \texttt{false}: 

\begin{verbatim}
The drop shadow always mimics the shape of the box itself; there is presently no way to avoid that. If for some reason you want to, you'll have to create a separate \texttt{roundrect} and place it manually.

We can control the size and direction of the drop shadow fairly easily, however, along with its color. Its color is controlled by \texttt{rrshadowcolor}, which can be set to any arbitrary \texttt{METAPOST} color:

```
rrdropshadow := true;
rrshadowcolor := blue;
roundrect(1in,2in)(rectangle);
draw rectangle;
```

The position of the drop shadow is governed by \texttt{rrshadowx} and \texttt{rrshadowy}, which will shift the \texttt{roundrect} on the \texttt{x} or \texttt{y} axis, respectively. By default, these are set to one quarter of the border radius in effect for the bottom left corner:
rrdropshadow := true;
rrshadowcolor := blue;
roundrect(1in,2in)(rectangle);
draw rectangle;

6 Including Text

Finally, we can put text in the rectangles; this is as configurable as everything else:

rrbodytext := "Let's put some text into this rectangle and see if it typesets correctly!"
roundrect(1in,2in)(rectangle);
draw rectangle;

rrtextfont
rrtextcolor The font and style of the text can be controlled with \texttt{rrtextfont}, and the
color of the text can be controlled with \texttt{rrtextcolor}:
We also used, without explaining it first, \texttt{rrtextalign}, which allows insertion of text alignment commands. This can also be inserted in the \texttt{rrtextfont} variable, but it seemed logical to have a separate parameter for it. It’s default is \texttt{\centering}.

The width of the text is governed by \texttt{rrtextwd}, which defaults to the same width as the rectangle with a 3pt buffer on either side. The buffer is not directly controllable, but the width can be set however you like:

Finally, to restore all these values to the default, use the \texttt{rrrestorevals}; directive. This will clear everything to default so you can have a completely different \texttt{roundrect} in the same figure.

7 Implementation

input TEX;
color rринnercolor; rринnercolor := white;
numeric rrтoprtborderrad; rrтoprtborderrad := 40pt;
numeric rrbotrtborderrad; rrbotrtborderrad := 40pt;
numeric rrbotlftborderrad; rrbotlftborderrad := 40pt;
numeric rrtopleftborderrad; rrtopleftborderrad := 40pt;
numeric rrtextwd; rrtextwd := 0;
numeric rrshadowx; rrshadowx := rrbotrtborderrad/4;
numeric rrshadowy; rrshadowy := -rrbotrtborderrad/4;
string rrtextfont; rrtextfont := "\fontsize{10pt}{12pt}\selectfont ";
color rrtextcolor; rrtextcolor := black;
string rrbodytext; rrbodytext := "\centering";
boolean rrnotop; rrnotop := false;
boolean rrnobot; rrnobot := false;
boolean rrnolft; rrnolft := false;
boolean rrnort; rrnort := false;
boolean rrdropshadow; rrdropshadow := false;
color rrtopleftbordercolor; rrtopleftbordercolor := black;
color rrbotbordercolor; rrbotbordercolor := black;
color rrleftbordercolor; rrleftbordercolor := black;
color rrrtbordercolor; rrrtbordercolor := black;
color rrshadowcolor; rrshadowcolor := black;
def rrtopleftborderrad(expr x) =
  rrtopleftborderrad := x;
def rrleftborderrad(expr x) =
  rrleftborderrad := x;
def rrbotborderrad(expr x) =
  rrbotborderrad := x;
def rrtopbordercolor(expr x) =
  rrtopleftbordercolor := x;
def rrleftbordercolor(expr x) =
  rrleftbordercolor := x;
def rrrightbordercolor(expr x) =
  rrrightbordercolor := x;
def rrcolor(expr x) =
  rrtopleftbordercolor := x;
pen rrtopleftborderpen; rrtopleftborderpen := pencircle scaled 1.5;
pen rrbotleftborderpen; rrbotleftborderpen := pencircle scaled 1.5;
pen rrleftborderpen; rrleftborderpen := pencircle scaled 1.5;
pen rrrightborderpen; rrrightborderpen := pencircle scaled 1.5;
pen rrtborderpen; rrtborderpen := pencircle scaled 1.5;
def rrborderpen(expr x) =
  rrtopleftborderpen := x;
def rrnotop = rrnotop;
def rrnobot = rrnobot;
def rrnolft = rrnolft;
def rrnort = rrnort;
def rrtextwd := 0;
rrbotborder := rrbotrcorner--rrbotlftcorner;
rrlftborder := rrbotlftcorner--rrtoplftcorner;
rrrtborder := rrtoprtcorner--rrbotrtcorner;
picture rrdropshadowpic;
if (rrdropshadow = true):
  rrdropshadowpic := image(fill rrtoplftcorner--rrtoprtcorner--
  rrbotrtcorner--rrbotlftcorner--cycle
  shifted (rrshadowx,rrshadowy) withcolor
  rrshadowcolor);
else:
  rrdropshadowpic := currentpicture;
fi
name := currentpicture;
addto name also rrdropshadowpic;
rrdropshadowpic := image(fill rrtoplftcorner--rrtoprtcorner--
  rrbotrtcorner--rrbotlftcorner--cycle withcolor
  rrinnercolor);
addto name also rrdropshadowpic;
% name := image(fill rrtoplftcorner--rrtoprtcorner--
% rrbotrtcorner--rrbotlftcorner--cycle withcolor
% rrinnercolor);
picture rrtmpborder;
border := currentpicture;
if (rrnotop = false):
  rrtmpborder := image(draw rrtopborder withcolor
  rrtopbordercolor withpen rrtopborderpen);
  addto border also rrtmpborder;
fi
if (rrnobot = false):
  rrtmpborder := image(draw rrbotborder withcolor
  rrbotbordercolor withpen rrbotborderpen);
  addto border also rrtmpborder;
fi
if (rrnolft = false):
  rrtmpborder := image(draw rrlftborder withcolor
  rrlftbordercolor withpen rrlftborderpen);
  addto border also rrtmpborder;
fi
if (rrnort = false):
  rrtmpborder := image(draw rrrtborder withcolor
  rrrtbordercolor withpen rrrtborderpen);
  addto border also rrtmpborder;
fi
rrtext :=
image(label(TEX("\parbox{\decimal(rrtextwd)}{\&rrtextalign&rrtextfont& "&rrbodytext&}")),
  addto name also rrtext;
enddef;