`\starttext`: Swelled rules and MetaPost

Steve Peter

Abstract

A swelled rule has tapered edges and a thicker midsection, thus blending better with the overall color of a page than a normal rule. In this column, we explore methods for creating them, using ConTEXt and MetaPost.

1 Introduction

Recently I was enjoying Ari Rafaeli's *Book Typography* and noted that he writes (page 93), "Swelled rules, adopted by English printers in the late eighteenth century, out of fashion in the early twentieth century, but revived by Stanley Morison and Nonesuch and The Curwen Press, are sadly missing from the typography of contemporary books." So I made a note to myself: use a swelled rule (also known as an English rule), in an upcoming project.

Coincidentally I read the column by Dave Walden in the 2005-4 issue of *The PracTEX Journal*, where he talks about thought groups. He discusses various ways to break text at a point where the thought transitions somewhat, but not enough for a section break. One traditional printer's technique for that is to use spaced out asterisks, as here.

* * *

Walden also suggests a horizontal rule. A swelled rule, by contrast, has a bit more presence on the page than a simple rule, but the tapered edges allow it to blend in with the overall color of the page. (Typographical color, that is, which comes down to how gray the page appears, not whether you use mauve or teal.)

So how might we go about swelling a rule? Punching it in the midsection doesn't seem to work.

2 Stairway to heaven

An early attempt, using Plain TEX, to provide for swelled rules is the `swrule` package by Tobias Dussa. It builds up the swelled rule by butting thicker or thinner rules together to create the illusion of a single swelled rule.

In an era of bitmapped fonts, there was nothing against such an approach. However, in the present era of zoomable PDFs, such rasterized solutions are suboptimal.

3 Quite a character

Another solution, probably the first one used in the computer typesetting era, and certainly back in the era of metal type, is to use a character or characters from a font to set the rule. Of course, getting and installing a specialized font for this one character can be annoying.

Given the flexibility, or rather distortability, of PostScript technology, we can still use a character from a font to achieve our goals, without precisely having a swelled rule character per se. Specifically, you might take a diamond dingbat and reduce the vertical axis while stretching the horizontal.

As a stopgap, this works well. The rule can be scaled at will, and it is a reasonable approximation of a swelled rule. It is not perfect, though, for a true swelled rule is like a plane journey. It rises and levels off before coming back down to the original level.

This seems like more of a graphics issue than a font issue. As you would expect, ConTEXt can use external graphics, and in fact you could simply place a picture of a swelled rule in your file. However, ConTEXt also has access to a far tidier setup via MetaPost. Let's take a look at that.

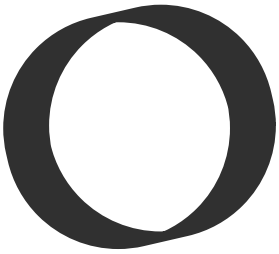
4 MetaPost in ConTEXt

Using MetaPost in ConTEXt is very straightforward, thanks to tight integration. You need not write the MetaPost code separately, nor do you need to compile it separately (although you can if you are so inclined). ConTEXt contains MetaPost interface macros, and the ConTEXt executable `texexec` will automatically call MetaPost for you. For example, try out the following as ConTEXt source:

```
\startuseMPgraphic{test}
   draw fullcircle scaled 3cm
      withpen pensquare xscaled 1mm
        yscaled 6mm rotated -77
      withcolor darkred ;
\stopuseMPgraphic
\useMPgraphic{test}
```

The result is approximately a calligraphic *o*, suitable for use as a stunning versal or dropcap, as shown:

Editor's Note: This article is reprinted, with additions, from *The PracTEX Journal*, 2005-4, `http://tug.org/pracjourn`.



For a much fuller discussion of MetaPost in ConTEXt, see Hans Hagen's *MetaFun* manual. Some amazing things can be done with the software combination described here.

One detail to note before we move on is that `\startuseMPgraphic` runs MetaPost each time the graphic is placed via `\useMPgraphic`. If you use the graphic hundreds of times in your document, the processor overhead can be intensive, so there is another command, `\startreusableMPgraphic`, which calculates the graphic once and reuses it. There is also `\startuniqueMPgraphic`, which makes the calculations based on arguments passed to the command. We'll see this below.

For now, let's see how we might draw a swelled rule using straightforward MetaPost code.

5 Brute force MetaPost

The most straightforward way to define a swelled rule is to set up points for the left and right edges and the points where the rule reaches its "cruising altitude". Consider (or better yet, try out) the following code:

```
\startuseMPgraphic{FirstSwell}
  z1 = (0,0); z2 = (100,1);
  z3 = (200,1); z4 = (300,0);
  z5 = (200,-1); z6 = (100,-1);
  fill z1--z2--z3--z4--z5--z6--cycle;
\stopuseMPgraphic
\useMPgraphic{FirstSwell}
```

First, we define the various coordinates in (x, y) pairs, assigned to variables named $z1$–$z6$ (as is typical with METAFONT and MetaPost). $z1$ is the leftmost point, while $z4$ is the rightmost point. The rule rises 1pt to point $z2$ and stays there through $z3$. The `fill` command draws a line through the points, cycles back to the origin and fills the resulting shape. (If you are typing this at home, note that each command in MetaPost is terminated by a semicolon, and you can put multiple statements per line.) Let's first have a look at the points we've set up:

Now, let's connect the dots:

Here's the resulting graphic when the points are connected and filled:

This is pretty good, in that you can zoom in as much as you want, you can change the color at will, and you can modify the length of the rule and how thick it is. Unfortunately, those last changes aren't as tranparent as they should be. Finding the correct coordinates is somewhat of a guessing game, and the dimensions of the rule are fixed. If you switch to a two-column layout, you will have to return to this code to hack in new values. We can do better.

6 A more refined MetaPost solution

The first thing we must do to get away from hard-coded solutions is to create variables, which is done in the standard ConTEXt setup manner.

```
\setupMPvariables[SwelledRule]
    [height=2pt,breadth=.667\localhsize]
```

`SwelledRule` is the name of the graphic, and we declare and assign two variables, a height and a breadth. The ConTEXt variable `\localhsize` gives the width of the text block. A good looking swelled rule isn't as broad as the text, so we'll make it 2/3 as broad.

Having set up these variables, we can write some MetaPost code that uses their values.

```
\startuniqueMPgraphic{SwelledRule}%
    {height,breadth}
  x1 = 0;
  x2 = x6 = .333x4; x3 = x5 = .667x4;
  x4 = \MPvar{breadth};
  y1 = y4 = \MPvar{height}/2;
  y2 = y3 = \MPvar{height};
  y5 = y6 = 0;
  fill z1--z2--z3--z4--z5--z6--cycle;
\stopuniqueMPgraphic
```

In many ways this is similar to the definition we wrote in the last section. The main difference is that we pass in the variables `height` and `breadth` and use them to calculate points on the rule. The $z1$–$z6$ coordinates are separated into their x and y

components in the first two lines, but no additional definition is needed. The pair $x1$ and $y1$ is identical to $z1$.

There are two ways to define the y points. The first, which I show here, is to place the endpoints at half-height and allow the rule to sit on the baseline ($y = 0$). Another approach is to place the endpoints on the baseline and have two of the y points ($y5$ and $y6$) as negative values.

Once we have the graphic, we can use it. We need to call `\setlocalhsize` each time to use the `\localhsize` variable, so let's make a macro to do that and center the rule:

```
\def\SwelledRule{%
  \setlocalhsize
  \midaligned{%
    \reuseMPgraphic{SwelledRule}}}
```

Now we use the rule and show the output:

```
\SwelledRule
```

Now to change the dimensions of the rule, all we have to do is modify the `MPvariables` via the normal ConTEXt setup mechanism. We could add additional variables—for instance to control the color of the rule (though I prefer black). I leave that addition as an exercise for the reader.

◇ Steve Peter
Beech Stave Press,
310 Hana Road,
Edison, NJ 08817
speter (at) beechstave.com