Hints & Tricks

“Hey — it works!”

Jeremy Gibbons

Welcome again to *“Hey — it works!”*, a column devoted to (LA)TEX and META tips and tricks. This issue is devoted to a single topic, a sequel to an earlier article on creating ornamental rules: we show how to construct ornamental boxes out of individual symbols.

Please note that I have moved. My new contact details are given at the end of this article. Unfortunately, mail is not being forwarded from my old address; I apologize profusely for any inconvenience that this may have caused. This column is being archived at `http://users.comlab.ox.ac.uk/jeremy.gibbons/hiw/`.

Ornamental boxes

In this column in *TUGboat* 19:4, Christina Thiele showed how to produce ornamental rules constructed from ordinary characters:

```
***************
```

It is also fun to generate ornamental *boxes* out of ordinary characters:

```
*********************
*                   *
*  shake the yoke of  *
*                   *
* inauspicious stars *
*                   *
*********************
```

This article shows how. We start off with a simple macro, and elaborate on it in stages.

First attempt: a single symbol

Our first attempt constructs an ornamental box out of copies of a single symbol, as in the example above. The macro `\boxitA` takes two arguments: the contents to be boxed, and the symbol (in fact, any horizontal material) that will be used to surround it. The first step is shift these two boxes vertically, if necessary, so that they have zero depth.

```
\def\boxitA#1#2{{%
  \setbox0=\hbox{#1}% the box contents
  \setbox0=\hbox{\raise\dp0\box0}%
  \setbox1=\hbox{#2}% the ornament
  \setbox1=\hbox{\raise\dp1\box1}%
```

Unfortunately, Christina's elegant use of leaders doesn't work as well for boxes as it does for rules; we have to achieve the same effects manually. We compute precisely how many instances of the symbol are required, horizontally and vertically, to exceed the dimensions of the contents; call these two numbers m and n. Each number is the size of the contents divided by the size of the symbol, rounded up to the nearest integer; we round upwards by first adding the size of the ornament less one.

```
  \count0=\wd0 \advance\count0 by \wd1
  \advance\count0 by -1 \divide\count0 by \wd1
  \count1=\ht0 \advance\count1 by \ht1
  \advance\count1 by -1 \divide\count1 by \ht1
```

The dimensions of the contents may not be exact multiples of the size of the symbol, so we wrap the contents in the smallest enclosing box with such dimensions:

```
  \setbox0=\hbox to \count0\wd1{%
    \hfil\vbox to\count1\ht1{%
      \vfil\box0\vfil}\hfil}%
```

Finally, we construct the ornamental box, with $m+2$ instances of the symbol at the top and bottom, and $n+2$ instances at the left and right:

```
  \hbox{\vbox{\offinterlineskip
    \hbox{\copy1%
          \duplicate{\count0}{\copy1}%
          \copy1}
    \hbox{\vbox{\duplicate{\count1}{\copy1}}%
          \copy0%
          \vbox{\duplicate{\count1}{\copy1}}}
    \hbox{\copy1%
          \duplicate{\count0}{\copy1}%
          \copy1}
  }}%
}}
```

Here, the macro `\duplicate` generates a given number (its first argument) of copies of a given text (its second argument):

```
\def\duplicate#1#2{{% #1 copies of #2
  \count2=#1%
  \loop
    #2%
    \advance\count2 by -1
  \ifnum \count2>0 \repeat}}
```

For example, the box at the start of this article was generated by the code

```
\boxitA{\begin{tabular}{c}
          shake the yoke of \\
          inauspicious stars
        \end{tabular}}
       {$\ast$}
```

Second attempt: multiple symbols

The first attempt gave the general idea; however, it would be nice to be able to use different symbols for the four edges and the four corners. In this second attempt, we provide such a facility. However, for simplicity we assume that all eight symbols are the same size. For example:

```
↘ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↙
→                   ←
→    bring me my    ←
→  arrows of desire ←
↗ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↖
```

As before, we start by making the contents and all eight of the symbols sit on the baseline:

```
\def\boxitB#1#2#3#4#5#6#7#8#9{{%
  % contents TL T TR L R BL B BR
  \setbox0=\hbox{#1}% the box contents
  \setbox0=\hbox{\raise\dp0\box0}%
  \setbox1=\hbox{#2}% #2 to #9 the ornaments
  \setbox1=\hbox{\raise\dp1\box1}%
  \setbox2=\hbox{#3}%
  \setbox2=\hbox{\raise\dp2\box2}%
  \setbox3=\hbox{#4}%
  \setbox3=\hbox{\raise\dp3\box3}%
  \setbox4=\hbox{#5}%
  \setbox4=\hbox{\raise\dp4\box4}%
  \setbox5=\hbox{#6}%
  \setbox5=\hbox{\raise\dp5\box5}%
  \setbox6=\hbox{#7}%
```

```
    \setbox6=\hbox{\raise\dp6\box6}%
    \setbox7=\hbox{#8}%
    \setbox7=\hbox{\raise\dp7\box7}%
    \setbox8=\hbox{#9}%
    \setbox8=\hbox{\raise\dp8\box8}%
```

(It would be nice to do this with a loop, but unfortunately you cannot use a counter to access a macro parameter.) Then we compute the number of symbols required, horizontally and vertically, and pad the contents accordingly:

```
    \count0 = \wd0 \advance\count0 by \wd1
    \advance\count0 by -1 \divide\count0 by \wd1
    \count1 = \ht0 \advance\count1 by \ht1
    \advance\count1 by -1 \divide\count1 by \ht1
    \setbox0=\hbox to \count0\wd1{%
      \hfil\vbox to\count1\ht1{%
        \vfil\box0\vfil}\hfil}%
```

Finally, we construct the ornamental box, taking care to use the correct symbol for each position:

```
    \hbox{\vbox{\offinterlineskip
      \hbox{\copy1%
            \duplicate{\count0}{\copy2}%
            \copy3}
      \hbox{\vbox{\duplicate{\count1}{\copy4}}%
            \copy0%
            \vbox{\duplicate{\count1}{\copy5}}}
      \hbox{\copy6%
            \duplicate{\count0}{\copy7}%
            \copy8}
    }}%
  }}
```

In order to use this macro, we need a means of making all eight symbols the same size. The macro `\resizeW` solves this problem: it yields its first argument, but centred in the width of its second argument. (Fortunately, all eight arrows are the same height, so no vertical adjustment is necessary.)

```
  \def\resizeW#1#2{{% #1, but to width of #2
      \setbox0=\hbox{#2}%
    \rlap{\hbox to \wd0{\hfil#1\hfil}}%
    \phantom{\box0}%
  }}
```

Then the box constructed out of eight arrows can be generated by

```
  \boxitB{\itshape
          \begin{tabular}{c}
            bring me my \\
            arrows of desire
          \end{tabular}}%
    {\resizeW{$\searrow$}   {$\searrow$}}
    {\resizeW{$\downarrow$} {$\searrow$}}
    {\resizeW{$\swarrow$}   {$\searrow$}}
    {\resizeW{$\rightarrow$}{$\searrow$}}
    {\resizeW{$\leftarrow$} {$\searrow$}}
    {\resizeW{$\nearrow$}   {$\searrow$}}
    {\resizeW{$\uparrow$}   {$\searrow$}}
    {\resizeW{$\nwarrow$}   {$\searrow$}}
```

Third attempt: different shapes

A little reflection suggests that there is no need for all eight ornaments to be the same size; all that is required is for those symbols that will be aligned together to have matching sizes in the appropriate dimension. Thus, if we call the four edge symbols T, B, L and R, and the four corner symbols TL, TR, BL and BR, then:

- TL, L, BL should have the same width;
- T, B should have the same width;
- TR, R, BR should have the same width;
- TL, T, TR should have the same height;
- L, R should have the same height;
- BL, B, BR should have the same height.

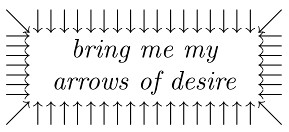
The only change required to the macro is to make sure the appropriate symbols are used when it comes to computing the number of symbols required:

```
  \def\boxitC#1#2#3#4#5#6#7#8#9{{%
    % contents TL T TR L R BL B BR
    \setbox0=\hbox{#1}% the box contents
    ...
    \count0 = \wd0 \advance\count0 by \wd2
    \advance\count0 by -1 \divide\count0 by \wd2
    \count1 = \ht0 \advance\count1 by \ht4
    \advance\count1 by -1 \divide\count1 by \ht4
    \setbox0=\hbox to \count0\wd2{%
      \hfil\vbox to\count1\ht4{%
        \vfil\box0\vfil}\hfil}%
    ...
  }}
```

Now it is possible to dispense with the resizing; we can write simply

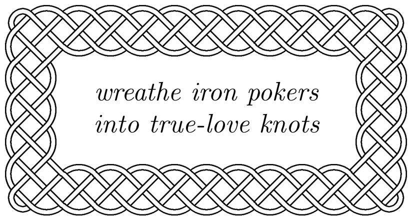
```
  \boxitC{\itshape
          \begin{tabular}{c}
            bring me my \\
            arrows of desire
          \end{tabular}}%
    {$\searrow$}{$\downarrow$}{$\swarrow$}
    {$\rightarrow$}{$\leftarrow$}{$\nearrow$}
    {$\uparrow$}{$\nwarrow$}
```

to generate

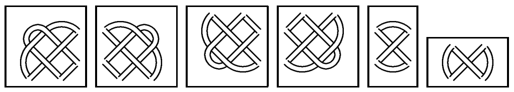


Knotwork

Sadly, there is a shortage of good symbols for creating such ornaments; not many typographic elements come in eight different orientations! However, there is nothing to stop you designing your own symbols:



This ornamental box uses a font of six different knotwork components:



(the top and bottom edges use the same symbol, as do the left and right edges). The designs are based on those in the book *Celtic Knotwork Designs* by Sheila Sturrock (Guild of Master Craftsman Publications, 1997).

◇ Jeremy Gibbons
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
`jeremy.gibbons@comlab.ox.ac.uk`
`http://www.comlab.ox.ac.uk/oucl/people/jeremy.gibbons.html`