Victor Eijkhout

TEXniques

Typesetting CD labels

Introduction

Now that CD burners are becoming standard equipment in personal computers, there is a need for software to typeset CD labels. Of course, one can just squeeze a normal paragraph of text in the confines of a label, but it would be much more elegant to set the text to use all the available space. In this short article I will explain the macros that I wrote during a Christmas holiday, and that contain a few neat tricks.

The full macros can be found on CTAN under

`http://tug.ctan.org/cgi-bin/CTANfilesearch.pl?FILESTRING=cd_labeler`

you will also need

`http://tug.ctan.org/cgi-bin/CTANfilesearch.pl?FILESTRING=repeats`

The plan of attack, and the obvious stumbling block

Since labels are circular, and TEX has a general paragraph mechanism in the `\parshape` primitive, it seems logical to compute line lengths and indentations to get text to fit on a label. In fact, we would need to fit the text in between the large circle of the outer rim, and the small circle of the hole of the label. Basically, from the number of lines that we are below the top or the middle of the CD, plus the `\baselineskip`, we get our y coordinate, and from that we compute the line length and indentation.

Eh, compute exactly how? TEX has only integer arithmetic, and even that is limited to adding, multiplying, and dividing. We need something like a square root to compute $x = \sqrt{r^2 - y^2}$.

In fact, the problems start before the square root computation: I decided to solve the integer problem by using 'scaled points', TEX's smallest measure, of which there are some thousands in a lowly printers point. Now, the radius of a CD label is a little over the square root of the maximum integer representable in TEX; in other words, computing r^2 gives integer overflow. This set me back for a while. I tried scaling down the quantities, taking the square root and scaling them up again, but this is not very elegant. The solution I found was to compute

$$r^2 - y^2 = (r + y)(r - y),$$

which is far less likely to overflow.

Next the problem of computing the square root. Given that TEX does have division the solution is clear: I had to implement Newton's method. (In fact, if TEX hadn't had division I could have implemented that too with Newton's method. In the early days of computing this was how the hardware did things even.) For the uninitiated: Newton's method is an idea of computing the solution of an equation by making successively better approximations. Formally, to compute the value x for which $f(x) = 0$, one computes a series of approximations x_i from

$$x_{i+1} = x_i - f(x_i)/f'(x_i).$$

In the case of square roots, where we want to compute $x = \sqrt{a}$, we let $f(x) = x^2 - a$, giving the iteration

$$x_{i+1} = (x_i + a/x_i)/2.$$

In TEX terms, using the \repeat macro I just explained, this can be written as

```
\count1=#1
\repeat \do {
   \count2=#1\relax \divide\count2 by \count1
   \advance\count2 by \count1 \divide\count2 by 2
   \ifnum\count2<\count1 \count1=\count2
     \else \expandafter\breakrepeat \fi}
```

This is obviously taken from a macro where the first argument is the number we need the root of. (Mathematically inclined TEXnicians may be amused to note the crude but effective stopping test of this iteration.)

User interface

For the user interface I decided on making a box, which actually gets printed, in which all material gets superimposed.

```
\CDLlabel{ <label content> }
```

Available commands for filling the label are:

```
\CDLbackground{
     <picture material> }
\CDLunderhole{<setup>}{
     <material printed below the hole> }
\CDLlowerhalf{<setup>}{
     <material on the lower half of the label,
      left of and below the hole> }
\CDLleft{<setup>}{
     <material printed on the left side of the label> }
\CDLright{<setup>}{
     <material printed on the right side of the label> }
```

Any combination of these can be given; it is up to the user not to give too much text. One thing I didn't implement is filling the whole label with text that would flow around the hole. It can be done, but I have never seen it, so the need must not be overwhelming.

The 'setup' part of the macros is where the \baselineskip setting has to be done; font changes can go there or in the actual text.

The resulting label is in essence a \vbox, so it can be positioned on the paper by the usual shift commands. I found it easiest to shift the \hoffset.

Of the above commands, the 'left' and 'right' are convenient for printing the disc title. However, since they start all the way at the top, the following command may come in handy:

```
\CDLblank{ <number of lines to skip> }
```

Just a little more about the internals

Since any of the above macros can be given in arbitrary combinations, internally they are all put in a box of zero width:

```
\long\def\CDLlabelbox#1{%
     \setbox\CDLbox=\vbox to \CDLdiscsize{\hsize=\CDLdiscsize
         #1\vfil
         \hrule width \CDLdiscsize height 0pt depth 0pt}%
     \wd\CDLbox=0pt \box\CDLbox}
```

The surrounding box is an `\hbox`, so these boxes are all ‘lined up’.
Internally, most of the hard work is done by this macro:

```
%% CDLline
%% from #1 line number compute #2 length, #3 inner gap #4 outer gap
%%
\def\CDLline#1#2#3#4{...]
```

Apart from the first argument, the arguments to this macro are control sequences that get defined in the macro definition. This is then used in such top level macros as

```
\def\CDLleft#1#2{\CDLlabelbox{
   \CDLtoks={}#1\relax
   \begingroup
     \repeat \for{CDLcount} \from{1} \do {%
       \CDLline\CDLcount\a\b\c
       \if\CDLerror \expandafter\breakrepeat \fi
       \edef\temp{\CDLtoks={\c\space\a \space\the\CDLtoks}}\temp
       }%
     \noindent
     \parshape \CDLcount 0pt 0pt \the\CDLtoks \hfill\break #2\par
   \endgroup
   }}
```

Note how the paragraph shape gets built up by gradually adding to a token list.

Conclusion

There are always new challenges for TEX to tackle. So far the primitives seem up to the task: typesetting in a circle was quite easy. Now I should really write a floating point arithmetic package to make the computations a bit more elegant :-)