Danlan type by Adriaan Goddijn

(quick font hack)

Abstract
When Frans Goddijn first showed me the Danlan font article in September 2019, I immediately thought that it would be fun to play with those letters a bit in TeX and MetaPost.

But then the almost inevitable thing happened that so often happens to me: I got distracted by other things, and forgot about Danlan completely. Until this spring, when Frans reminded me that I had promised an article for the Maps. This is that promised article: it will show what a few days playing around with a specification and MetaPost, FontForge, and ConTeXt got me. I have not created a complete font by any means, but it is just enough of one to show off a little bit and document how the creation process worked out for me.

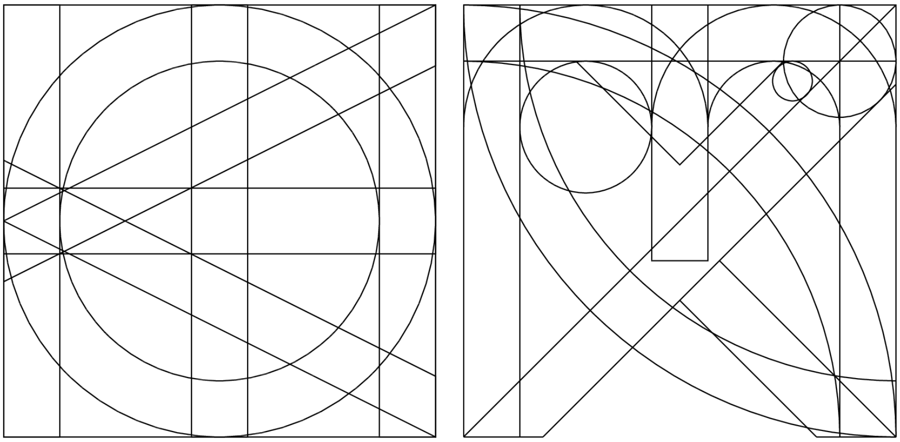
Preliminaries

My first attempt was to bitmap-trace the seven glyphs from the demonstration image straight into FontForge. I tried that first because at the time I did not quite understand the design helper images. The plan was to have the glyphs drawn in FontForge, then convert back to MetaPost input format afterwards for playing around with them.

However, the results were very underwhelming. The resolution of the bitmaps was just not high enough for a clean trace. So, just using FontForge did not work out. But as I originally wanted to play with MetaPost anyway that was not a big deal. In this early stage I was clearly trying to cut too many corners.

In preparation for a second attempt, I decided to recreate the two design images in MetaPost, and then continue work from those. I should probably apologize for the sloppiness of the MetaPost code shown below, but I won't! This messy stuff is how I typically work, and then if something needs publishing, I clean it up after its functionality is already at 100%.

Here are the two drawings I created:



Not quite the same as the example drawings, but that is OK, as these contain all the lines that I needed. The few extra lines and curves are fine, I just needed to recreate the image to understand the design. They were never intended to be used as anything else than a quick reference for myself, and the MetaPost code clearly shows that.

This is the left image; this one is used for H and I as well as A, K, V, and X:

```
d = 130;
w = cosd(22.5);
e = d * 1/(w*w);
path leftup;
pickup pencircle scaled 3;
draw unitsquare scaled 1000;
leftup = (0, 500-(d/w))--(1000,1000-(d/w))--(1000,1000)--(0,500)--cycle;
draw leftup;
draw leftup reflectedabout ((0,500),(1000,500));
draw (0,0)--(d,0)--(d,1000)--(0,1000)--cycle;
draw (500-(d)/2,0)--(500+(d)/2,0)--(500+(d)/2,1000)--(500-(d)/2,1000)
                                                             --cycle;
draw (1000-d,0)--(1000,0)--(1000,1000)--(1000-d,1000)--cycle;
draw (0,500-e/2)--(1000, 500-e/2)--(1000, 500+e/2)--(0,500+e/2)--cycle;
draw fullcircle scaled 1000 shifted (500,500);
draw fullcircle scaled (1000-2*d) shifted (500,500);
```

The interesting things in this drawing are:

`d`, which is the width of the vertical bars. This becomes an actual font meta-ness parameter (the only one).
`w`, which is a MetaPost shortcut. It makes the diagonals the same apparent width as the vertical bars.
`e`, which is the width of the horizontal middle bar. The definition is a bit odd, but this most closely matches Adriaan Goddijn's drawing.
And the units are simply set up for ease of use.

Here is the right image, which is used for all the other letters in the alphabet:

```
v = cosd(45);
save diam,x,y,p;
path p[];
p1 = ((0+d/v,0)--(1000,1000-(d/v)));
z5 = ((1000,0)--(0,1000)) intersectionpoint p1;
z6 = ((1000-d/v,0)--(0,1000-d/v)) intersectionpoint p1;
z1 = (d+d, 1000-d); z1' = (x1 + 1000,y1 - 1000);
z2 = (1000-d-d,y1); z2' = (x2 - 1000,y2 - 1000);
z3 = (z1--z1') intersectionpoint (z2--z2');
z4 = (z1--z1') intersectionpoint ((0,0)--(1000,1000));
diam = arclength (z3--z4);
draw (0,0)--(0,1000)--(1000,1000)--(1000,0);
draw (0,0)--(d/v,0);
draw (1000,0)--(1000-d/v,0);
draw quartercircle scaled 2000;
draw quartercircle scaled (2000-2*d);
draw (quartercircle scaled 2000) rotatedaround((500,500), 180);
draw (quartercircle scaled (2000-2*d)) rotatedaround((500,500), 180);
draw (0,0)--(1000,1000);
draw (0+d/v,0)--(1000,1000-(d/v));
draw (d,0)--(d,1000);
draw (1000-d,0)--(1000-d,1000);
```

```
draw (0,1000-d)--(1000,1000-d);
draw (1000,0)--z5;
draw (1000-d/v,0)--z6;
x7 = 500-d/2;
x7'= 500+d/2;
y7'= 1000 -(x7'/2);
draw (x7, 1000)--(x7, y5)--(x7',y5)--(x7',1000);
draw halfcircle scaled x7' shifted (x7'/2,y7');
draw (halfcircle scaled x7' shifted (x7'/2,y7')) shifted (x7,0);
draw fullcircle scaled (x7'-2*d) shifted  (x7'/2,y7');
draw (halfcircle scaled (x7'-2*d) shifted (x7'/2,y7')) shifted (x7,0);
draw z1--z3--z2;
draw fullcircle scaled (2*d) shifted  (1000-d, 1000-d);
draw fullcircle scaled diam shifted (x2+v*(diam/3.1415), 1000-d-diam/2);
```

That one is a bit more involved. By far the most interesting object in that drawing is that little circle in the top right. Deducing its diameter and (especially) its location was a bit tricky. I am still not sure I have done it completely right, but the current definition produces a result that – I think – is good enough.

Implementation

With the two design images done, it became a simple exercise to code the seven main glyphs in MetaPost. For example, here is the basic definition of one of them:

```
def letter_k =
  begingroup
  save e,leftup,leftdown,vb,vt;
  path leftup, leftdown;
  e = d * 1/(w*w);
  leftup = (0, 500-(d/w))--(1000,1000-(d/w));
  leftdown = leftup reflectedabout ((0,500),(1000,500));
  vb = 500-e/2;
  vt = 500+e/2;
  z1 = ((d,0)--(d,1000)) intersectionpoint ((0,vb)--(1000,vb));
  z2 = ((0,vb)--(1000,vb)) intersectionpoint ((0,500)--(1000,0));
  z3 = ((d,0)--(d,1000)) intersectionpoint ((0,vt)--(1000,vt));
  z4 = ((0,vt)--(1000,vt)) intersectionpoint ((0,500)--(1000,1000));
  z5 = leftup intersectionpoint leftdown;
  (0,0)--(d,0)--z1--z2--(1000,0)--(1000,(d/w))--z5--(1000,1000-(d/w))--
     (1000,1000)--z4--z3--(d,1000)--(0,1000)--cycle
  endgroup
enddef;
```

In total, there were seven of these definitions. Each of these is called two times: once for a filled-in style, and once for an outline style:

```
beginchar(107, "k")
fill letter_k;
endchar;

beginchar(75, "K")
fill letter_k withcolor white withpen pencircle scaled ministrokew;
draw letter_k withpen pencircle scaled strokew;
currentpicture := currentpicture shifted(strokew/2, strokew/2);
endchar;
```

Finally, there is a bit of a preamble to that file needed:

```
d := stem;
w := cosd(22.5);
v := cosd(45);
ministrokew := (stem/11); % just to give width to the eraser

linejoin := mitered;
miterlimit := infinity;
linecap := butt;

def beginchar(expr c, s) =
  if string s:
    outputtemplate := "%j-%c-"&s&".%o";
  else:
    outputtemplate := "%j-%c.%o";
  fi
  beginfig(c)
enddef;

def endchar =
  currentpicture := currentpicture scaled 0.8;
  currentpicture := currentpicture shifted (lsidebearing,0) ;
  endfig
enddef;
```

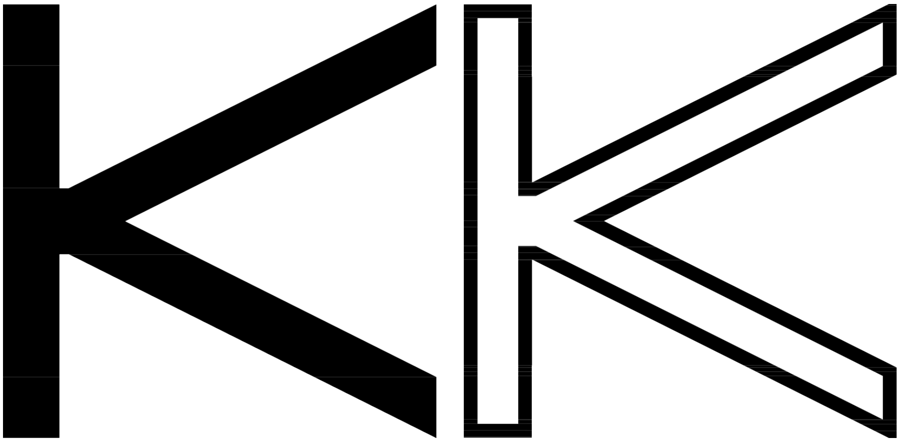
All those macros were saved in a MetaPost file, and then a driver file is used to set the small amount of variables:

```
stem = 130;
lsidebearing = 70;
strokew = (stem/4);

input danlan-design;

end.
```

Processing one of these driver files produces the raw EPS images of the base glyphs. For example, here are the 'k' and 'K' versions in the regular width:



At that point, the 'real' work was done. What was left was to import to glyphs in FontForge, do some visual manipulation like adding the rotated versions, adding points at extrema, rounding all points to integers, and setting the right sidebearings.

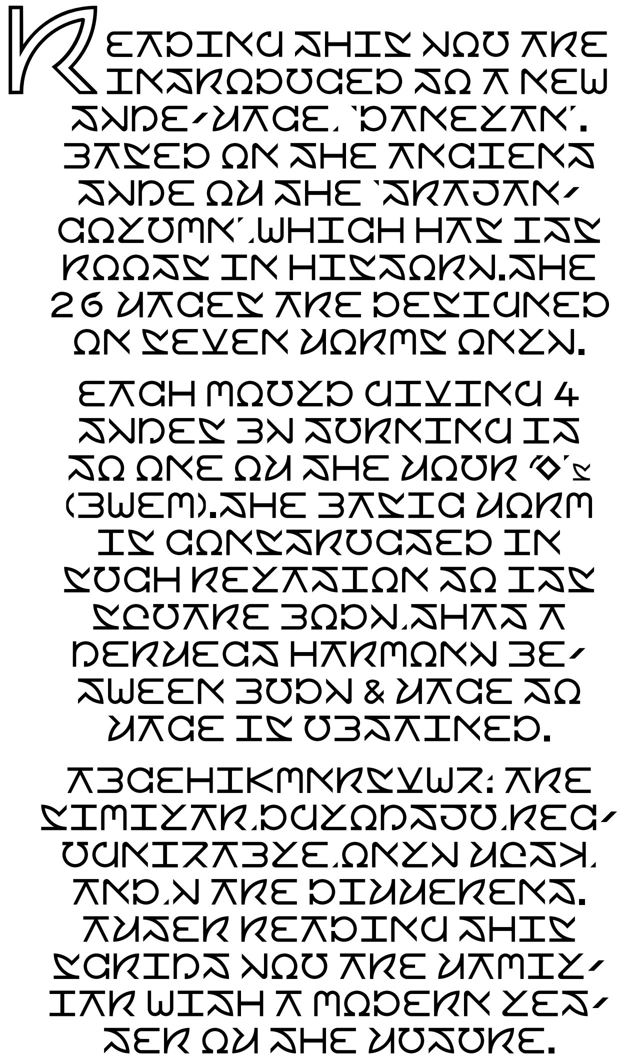


Figure 1. Regular version.

Figure 2. Bold version.

Final touches and result

There are a few glyphs in the demonstration text that were not based on the design, like the punctuation marks, the digits, and the rotation symbol. Those were simply created in FontForge directly for the 'regular' version of the font. My capital letters are less elaborate than single 'R' initial from the demonstration text, but I wanted to create all of the uppercase letters and it was not obvious from the demonstration how many and which of the helper lines should be added as the decorative backdrop to the initial, so I chose to skip all of them.

```
\definefont[danlannormal][Danlan at 12pt]
\definefont[danlaninit]  [Danlan at 40pt]
\definefont[danlanx]     [Danlan at 8pt]
```

After generating an OpenType font from FontForge, it was time to prepare a TeX version of that textual demonstration:

```
\danlannormal

\setbox1=\hbox
  {\kern -18pt
   \smash{\hbox{\danlaninit R}}}

\blank[2*line]
\setuplocalinterlinespace
  [line=14pt]

\startlines
\dontleavehmode %
\kern\wd1 eading this you are
\dontleavehmode %
\box1 introduced to a new
type-face, `danelan'.
based on the ancient
type of the `trajan-
column',which has its
roots in history.the
26 faces are designed
on seven forms only.
each mould giving 4
types by turning it
to one of the four @'{\danlanx s}
(bwem).the basic form
is constructed in
such relation to its
square body,that a
perfect harmony be-
tween budy & face to
face is ubtained.

abcehikmnrsvwz: are
similar,dgloptju,rec-
ugnizable,only fqtx,
and,y are different.
after reading this
script you are famil-
iar with a modern let-
ter of the future.
\stoplines
```

The text above in a faithful representation of the demonstration image. Interestingly, it has a number of typos. The typeface name is actually OK (it was initially named 'danelan' instead of 'danlan') but there are a few mistakes with u instead of o, and in the alphabetic list at the end, the 't' is mentioned twice!

The typeset result is in figure 1.

Using a slightly different driver file with a wider stem, it was possible to create a bold version of the font as well. This one only has the 52 main glyphs, and its effect can be seen in figure 2.

Afterthoughts

Having spent two days on the fonts in the end creating something that is of extremely limited use, I have to wonder whether this was worthwhile. The chance of anyone extending my MetaPost code to a complete, usable font family seems infinitesimally small.

On the other hand, it did result in this article and perhaps most importantly: I had a lot of fun playing around!

Taco Hoekwater