Variable Fonts

we’re ready for them

Hans Hagen

BachoTUG 2017

A Summary

- **the macro package's view:** just a font but with many possible variations in shapes (width, weight, slope, etc) and therefore a bit more complex user interface
- **the engine's view:** an abstraction not different from other fonts but that needs a special treatment in the backend
- **the viewer's view:** a font to be displayed like any other with outlines in cff of ttf format
- **the user's view:** an opentype font with possibly surprising shapes of which you need to know a bit more than usual if you want to profit from it

So, in practice, for most TEX users it's just a font that has to be supported by TEX and friends.

Starting point

- The OpenType 1.8 specification at the MicroSoft website defined the extra tables and explains bits and pieces.
- There a few fonts that have relevant tables (not all) and implement variants as well as features.
- There are some posts on the internet that show a bit about axis and other things that go on in these fonts.
- Luckily we have ways (in CONTEXT) to explore what goes on in these fonts and how they could look.
- Condition: no tricks, no fuzzy heuristics, just the specification should be enough.

Implementation steps

- First try to render variants in order to see what we're dealing with. This was not too hard (starting with cff) because we have already virtual font support.
- Next try to load the relevant tables and figure out what these deltas and such really mean and how axis and regions and ... have to be applied.
- Try to make it all work on a real piece of text, so not only shapes but also features and dimensions.
- Finally make sure that the font can get embedded as a normal font and not as inline (tagged) graphic.
- Also, try to generalize the helpers and methods in such ways that we can experiment with additional tricks (after all, TEX is about control).
- Todo: once there are more fonts (with the right data tables), check the code with the specification.

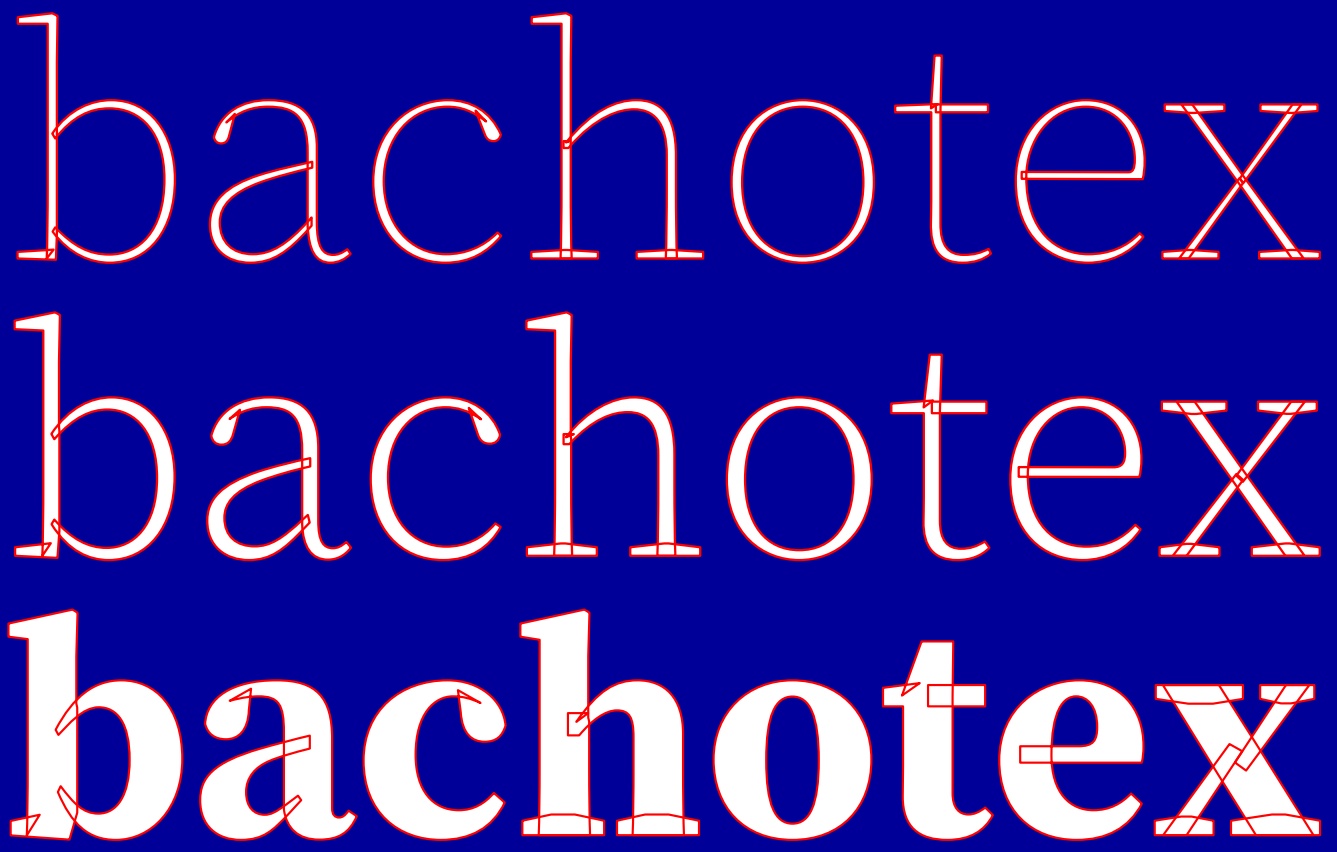
Adobe Variable Font Prototype (cff)

extralight 0/0	It looks like this!
light 150/0	It looks like this!
regular 394/0	It looks like this!
semibold 600/0	It looks like this!
bold 824/0	It looks like this!
black high contrast 1000/100	It looks like this!
black medium contrast 1000/50	It looks like this!
black 1000/0	It looks like this!

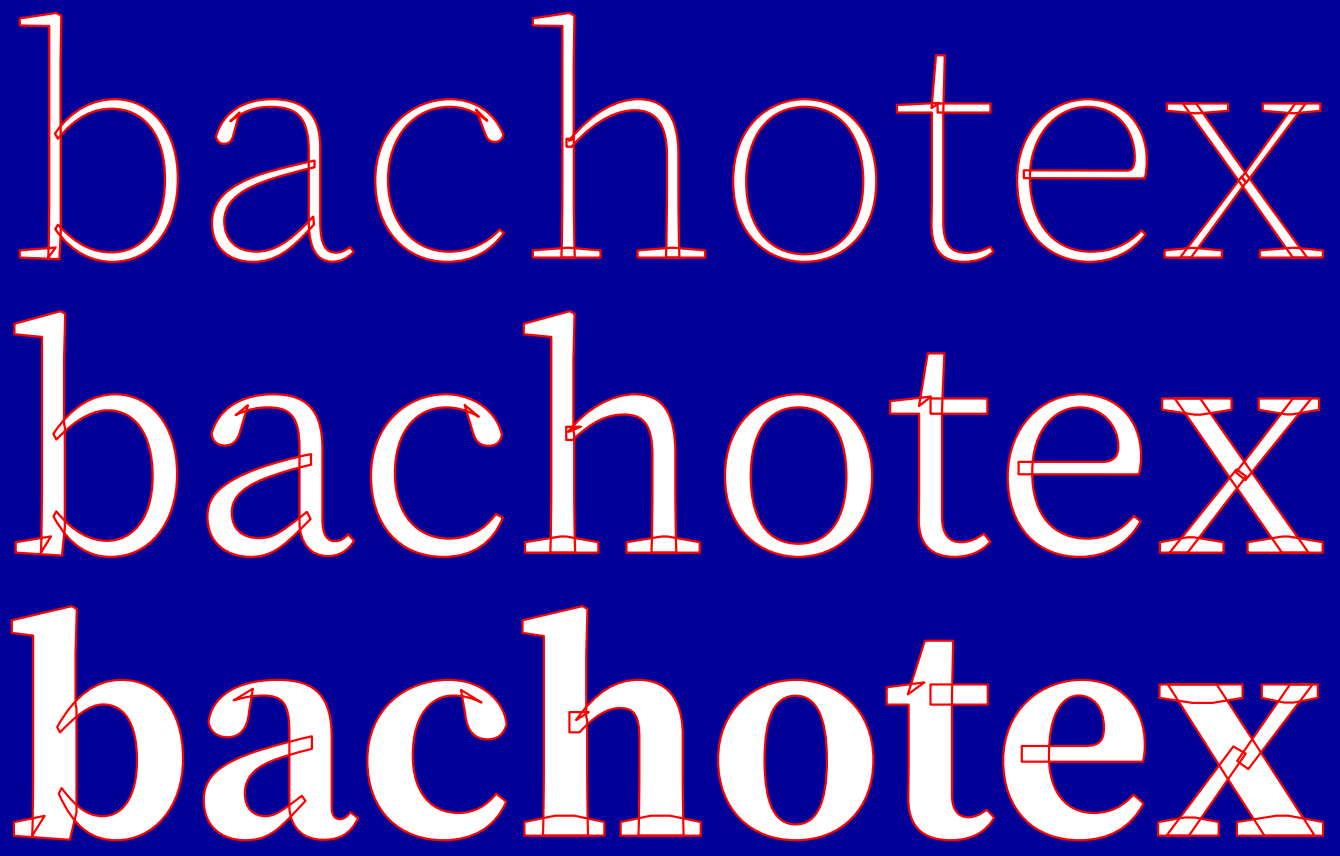
Avenir Next Variable (ttf)

regular 400/100	It looks like this!
medium 500/100	It looks like this!
bold 700/100	**It looks like this!**
heavy 900/100	**It looks like this!**
condensed 400/75	It looks like this!
medium condensed 500/75	It looks like this!
bold condensed 700/75	**It looks like this!**
heavy condensed 900/75	**It looks like this!**

Metafontisch overlap (1)



Metafontisch overlap (2)



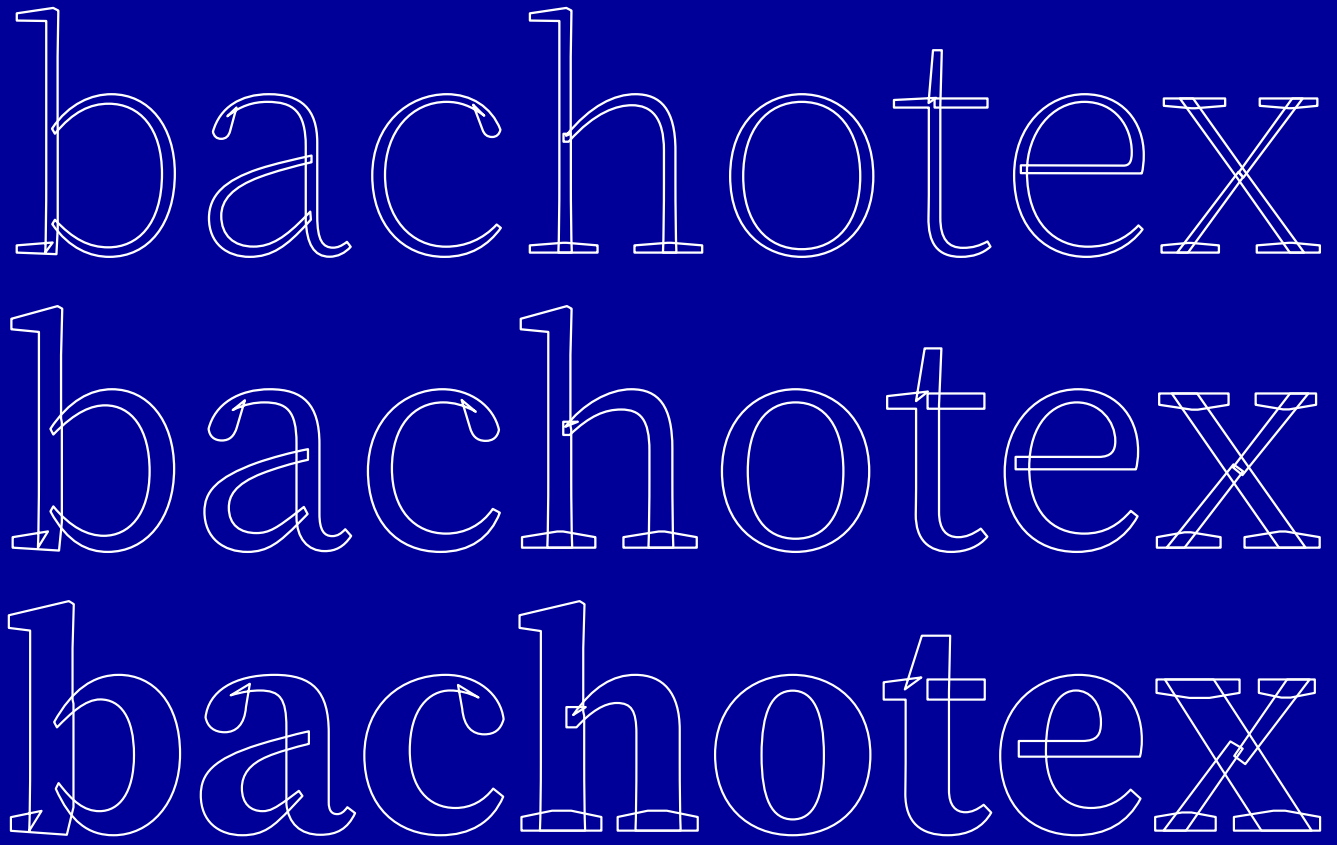
Fills hide the details

bachotex

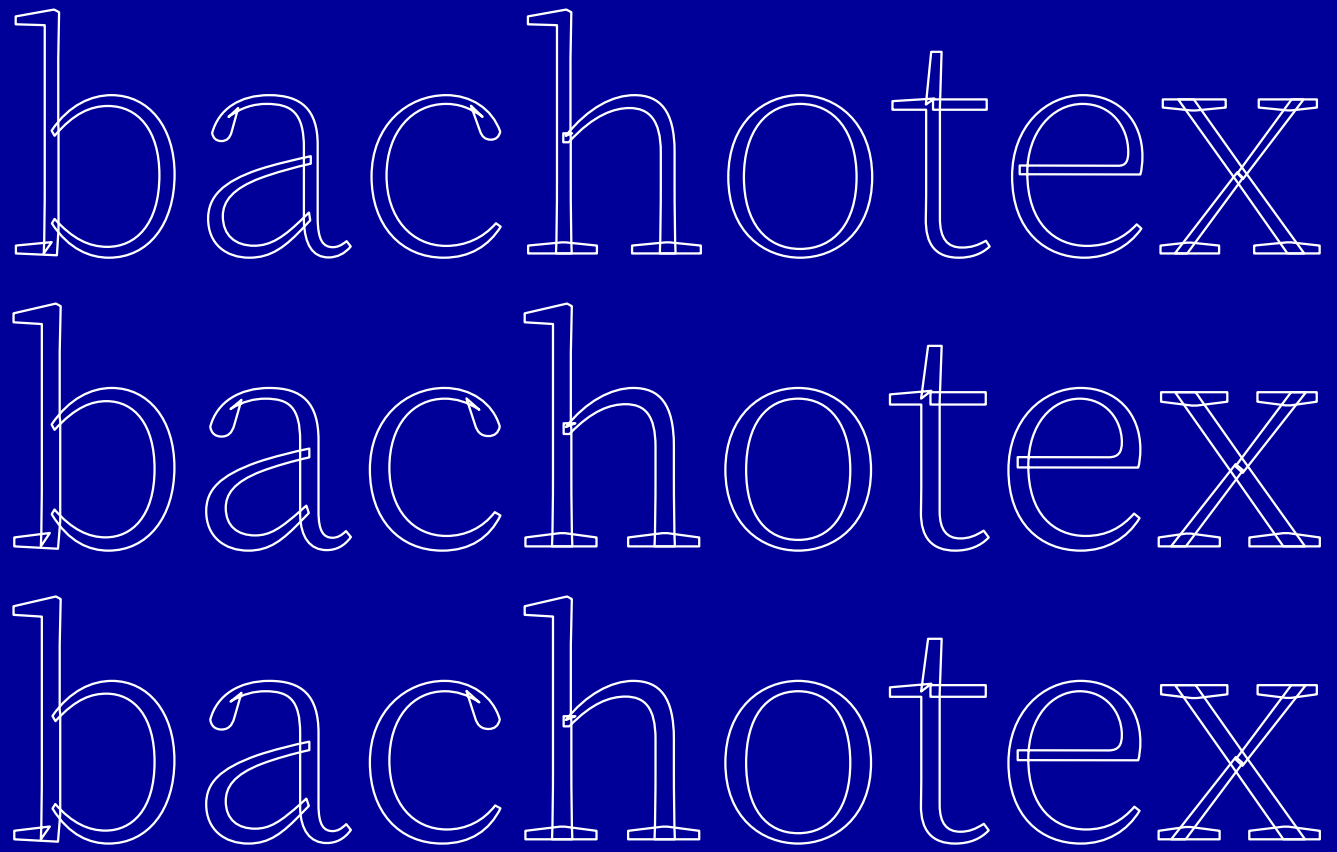
bachotex

bachotex

Unsuitable outlines



Stay within specification



Subjective choices

bachotex
bachotex
bachotex

Difficult choices

Definitions (1)

```
\definefontfeature
  [default:shaped]
  [default]
  [axis={width:10}]

\definefont
  [SomeFont]
  [file:avenirnextvariable*default:shaped]
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manualls which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on.

Hermann Zapf

Definitions (2)

```
\definefontfeature
  [default:shaped]
  [default]
  [axis={width:100,weight=200}]

\definefont
  [SomeFont]
  [file:avenirnextvariable*default:shaped @ 12pt]
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on.

Hermann Zapf

Transformations

correction:

$$x' = x + s_{x1} \cdot x_1 + s_{x2} \cdot x_2 + s_{x3} \cdot x_3 + s_{x4} \cdot x_4$$

$$y' = y + s_{y1} \cdot y_1 + s_{y2} \cdot y_2 + s_{y3} \cdot y_3 + s_{y4} \cdot y_4$$

internal cff:

```
1 <setvstore>
120 [10 -30 40 -60] 1 <blend> ... <operator>
100 120 [10 -30 40 -60] [30 -10 -30 20] 2 <blend> .. <operator>
```

external ttf:

```
apply x deltas [10 -30 40 -60] to x 120
apply y deltas [30 -10 -30 20] to y 100
```

Follow up

- Performance is quite okay because we cache instances. I might come up with an alternative way but there is not much to gain.
- Once fonts show up alternative interfaces to axis and scaling can be explored and provided.
- I will look into ways to do all the backend font code in CONTEXT in LUA (easier to update and more flexible).
- Luigi and I will play with variable fonts defined in the traditional meta tools that come with TEX.