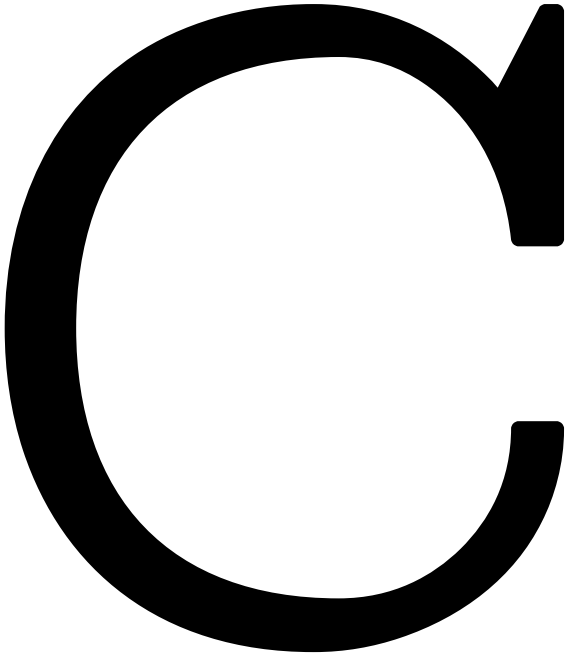


MFLUA

- *instrumenting* the METAFONT source code with Lua functions
- completely compatible with METAFONT 2.718281
- original outlines of a glyph (no {po/auto}trace of the bitmap)
- https://github.com/luigiScarso/mflua

Basically, MFLUA runs a METAFONT source with a particular mode:

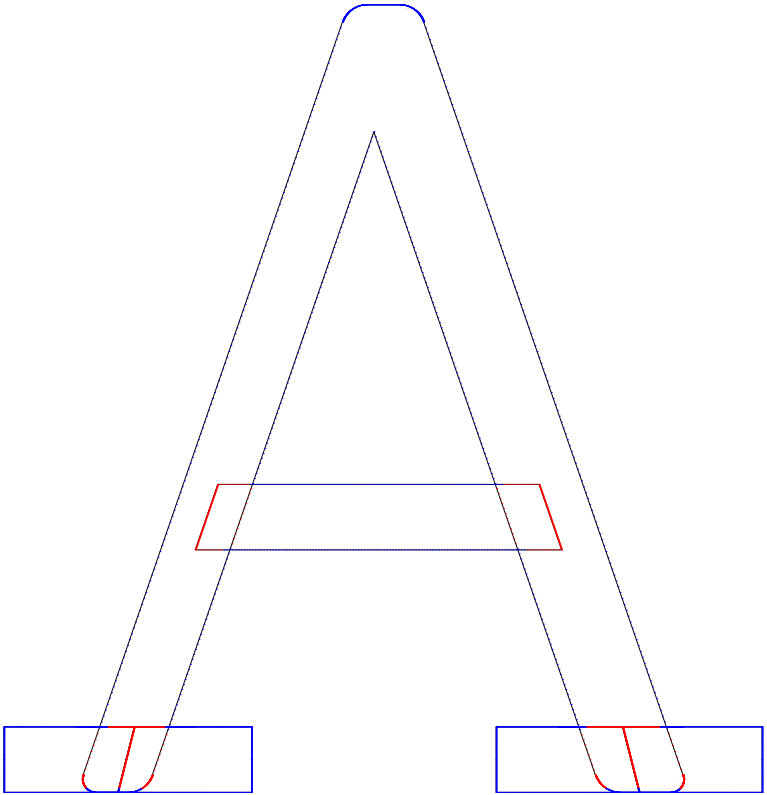
```
mode_def otcff =
  mode_param (pixels_per_inch, 4000);
  mode_param (blacker, 0);
  mode_param (fillin, 0);
  mode_param (o_correction, 1);
  mode_common_setup_;
enddef;
```

During the run, the ‘sensors’ collect the data. Sensors are callback functions and they are organised in files:

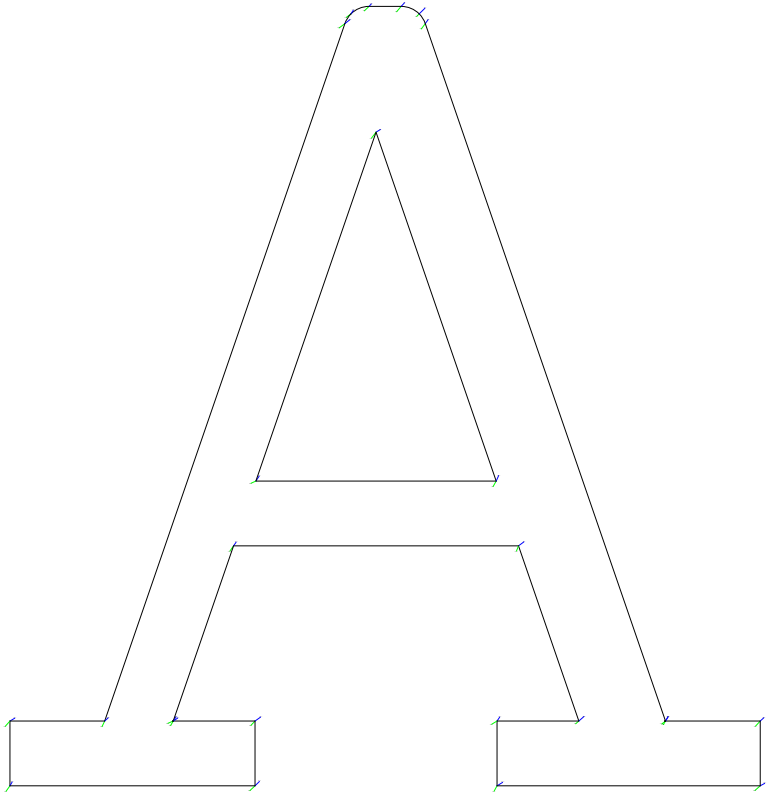
```
begin_program.lua      mfluaini.lua
do_add_to.lua          offset_prep.lua
end_program.lua        parse-log.lua
envelope.lua           print_edges.lua
fill_envelope.lua      print_path.lua
fill_spec.lua          scan_direction.lua
final_cleanup.lua      skew_line_edges.lua
main_control.lua       start_of_MF.lua
```

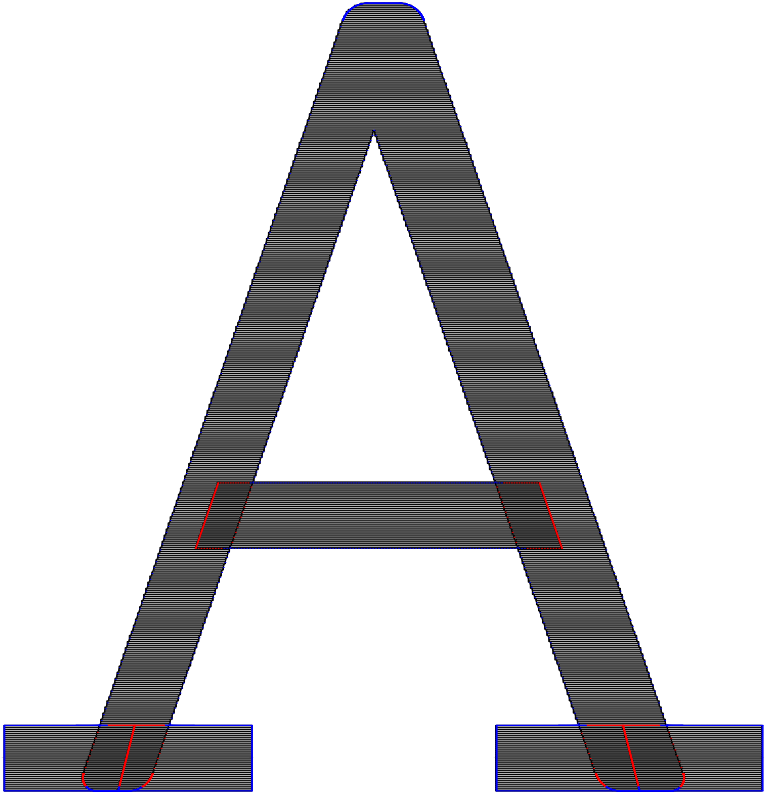
(the names come from the METAFONT source code)

When the run finishes, `end_program()` ‘clean’ the data from this

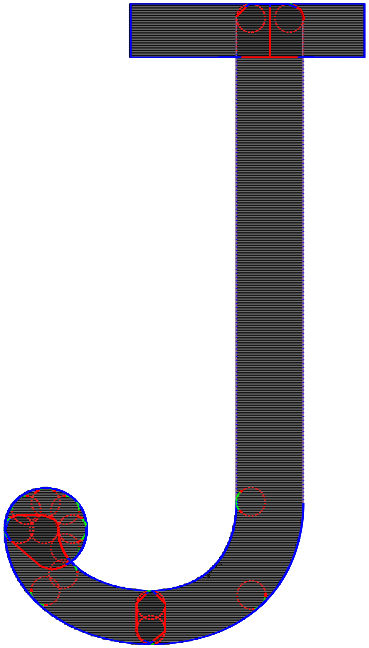


to this





Of course there are problems...and some of them I believe that cannot be solved in an automatic way



but I still like to investigate

As of today, the ‘data’ is just a set of Bezier curves p_i,c_{1i},c_{2i},q_i so it’s natural to translate them into a SVG picture (a closed path need a bit more attention)

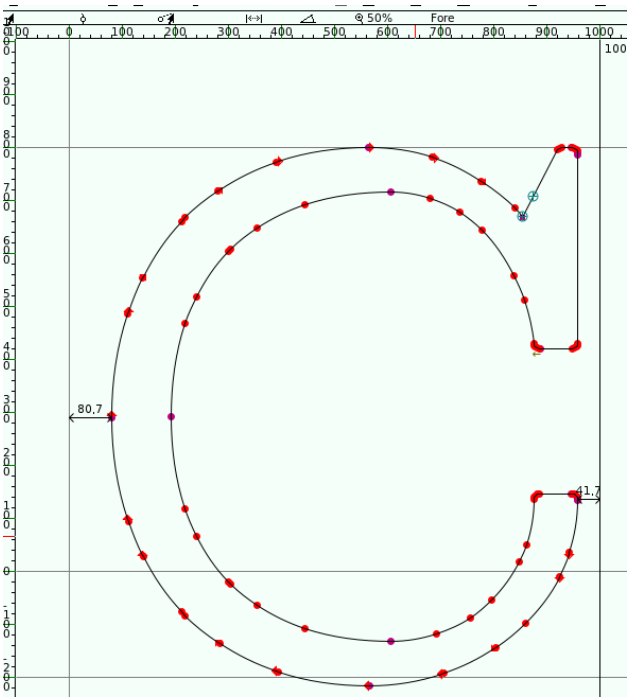
```
uni0041-raw.svg uni004e-raw.svg
uni0042-raw.svg uni004f-raw.svg
uni0043-raw.svg uni0050-raw.svg
uni0044-raw.svg uni0051-raw.svg
uni0045-raw.svg uni0052-raw.svg
uni0046-raw.svg uni0053-raw.svg
uni0047-raw.svg uni0054-raw.svg
uni0048-raw.svg uni0055-raw.svg
uni0049-raw.svg uni0056-raw.svg
uni004a-raw.svg uni0057-raw.svg
uni004b-raw.svg uni0058-raw.svg
uni004c-raw.svg uni0059-raw.svg
uni004d-raw.svg uni005a-raw.svg
```

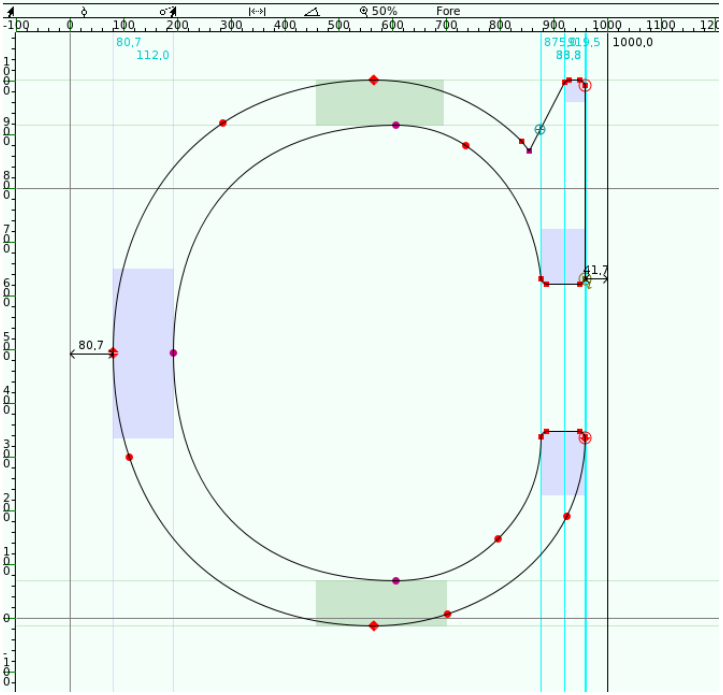
The next step is to use Fontforge (the program, or a custom Lua binding) to ‘learn’ how to produce an OpenType CFF font: simplify the paths, integral coordinates, dimensions... and much more.

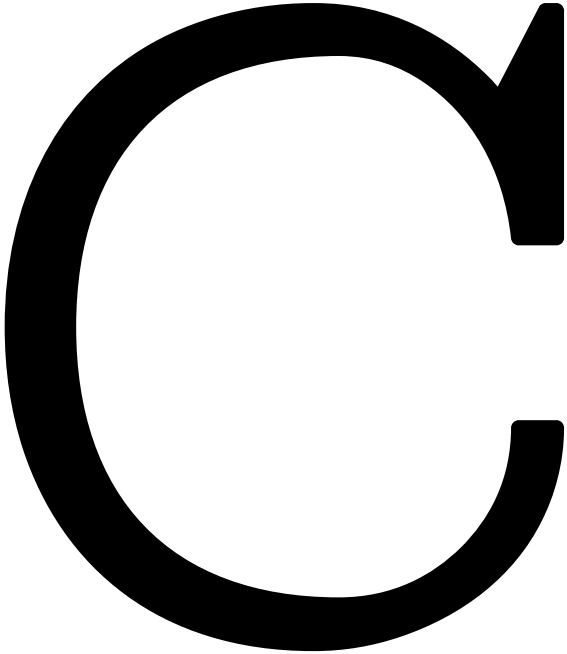
The strategy is to use Fontforge to see how to divide the responsibility: for example path simplification, integral coordinates and kerning probably should be done in `end_program` — to produce the best set of outlines. Fontforge can be scripted, but I believe that at the end a ‘visual’ interaction cannot be avoided.

Maybe ConTEXt-MkIV?

Concrete10OT is the temporary name of the OpenType version of ccr10. The first and only glyph is C




50%
Fore
80,7
112,0
1000,0
80,7
41,7



That’s all
Thank you !