Instrumentation of METAFONT with Lua

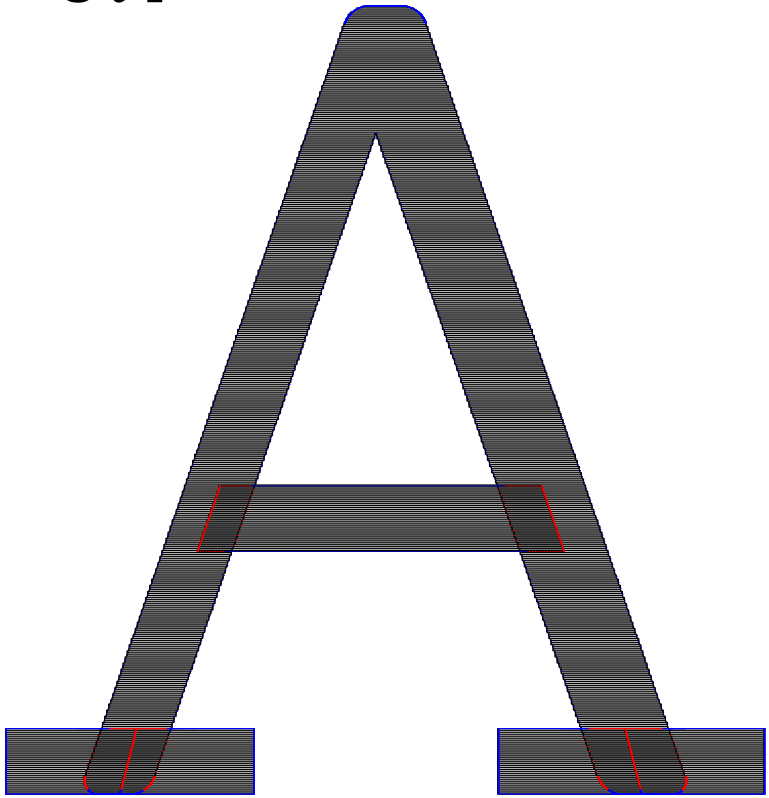
MFLua

- instrumentation of the METAFONT source code (PASCAL-WEB) with Lua functions (embedding of a Lua interpreter)
- completely compatible with METAFONT 2.718281
- original outlines of a glyph
  (no po/autotrace of the bitmap)
- https://github.com/luigiScarso/mflua

Basically, MFLua runs a METAFONT source; the mode used is

```
mode_def otcff =
  mode_param (pixels_per_inch,3600+3600);
  mode_param (blacker, 0);
  mode_param (fillin, 0);
  mode_param (o_correction, 1);
  mode_common_setup_;
enddef;
```

During the run, the functions collect the data, i.e. the cubic Bezier curves p_i, c_{1i}, c_{2i}, q_i and the pixels of the glyph.

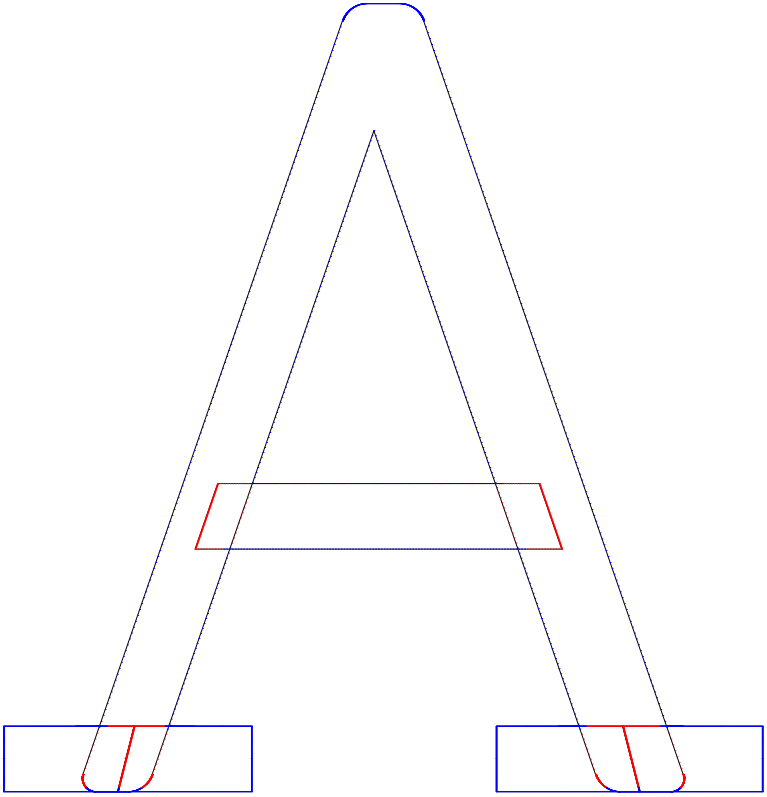


Each internal "sensor" call exactly one external Lua file:

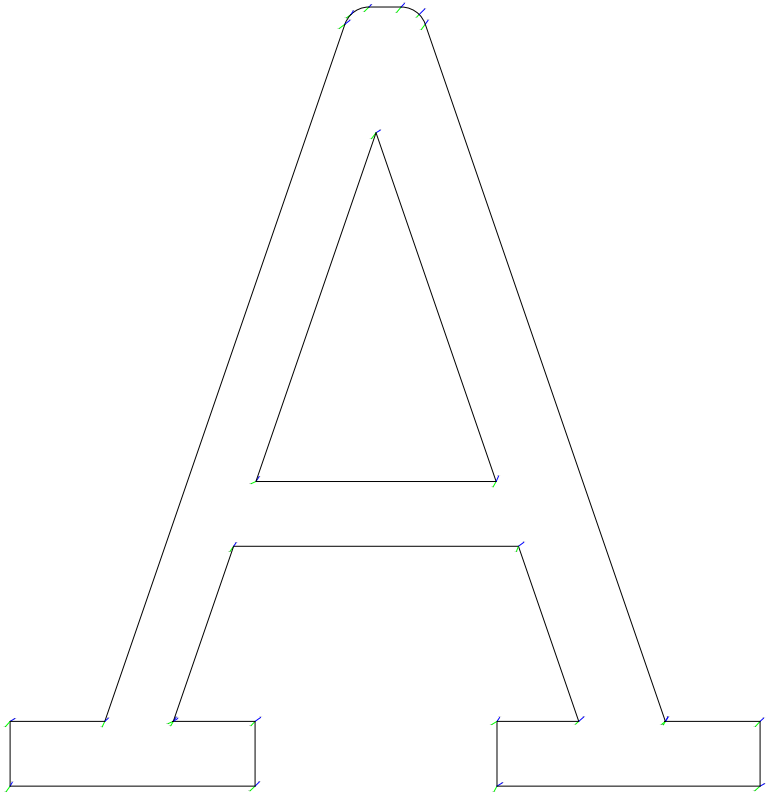
```
begin_program.lua
bezier.lua
do_add_to.lua
end_program.lua
end_program_poly_to_bezier.lua
fill_envelope.lua
fill_spec.lua
main_control.lua
make_ellipse.lua
mfluaini.lua
mflua_svg_backend.lua
namelist.lua
offset_prep.lua
parse-log.lua
pen_curves.lua
poly_to_bezier.lua
print_edges.lua
print_path.lua
scan_direction.lua
simplify.lua
skew_line_edges.lua
start_of_MF.lua
tfm.lua
transition_lines.lua
```

(some of names come from the METAFONT source code)

When the run ends, `end_program()` cleans the data from this

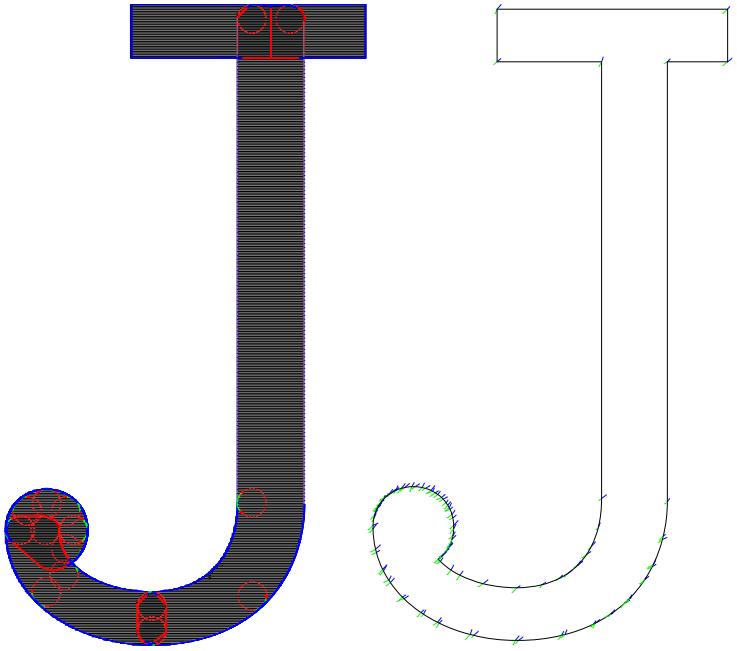


to this



and produce a SVG font file.

The set of all curves of a glyphs can be complicated:



Strategies

- design the glyph in METAFONT, and only when it is OK clean out the curves (manually, if necessary)

- use METAFONT to solve a geometric problem

- use METAPOST to show the curves produced by MFLua

- use the best tool to produce the font in otf format

FontForge (the program) can be used to convert a SVG font file into an OpentType font file.

- import the SVG font produced by MFLua and open a traditional editing session.
- execute a FontForge script from inside `end_program()`, by means of `os.execute(cmd);`
- extend Lua with a FontForge binding

- MFLua was able to produce a SVG font `ConcreteOT.svg` from `ccr10.mf` (at 4000 dpi)
- FontForge was used to simplify the curves of the SVG font and convert it to `ConcreteOT.otf`, a CFF OpenType font
- `ConcreteOT.otf` is the font used in these slides, and a Type1 version was used for the paper for the BachoTEX 2012 meeting

! 33 exclam	# 35 numbersign	$ 36 dollar	% 37 percent	& 38 ampersand
’ 39 quotesingle	(40 parenleft	) 41 parenright	* 42 asterisk	+ 43 plus
, 44 comma	- 45 hyphen	. 46 period	/ 47 slash	0 48 zero
1 49 one	2 50 two	3 51 three	4 52 four	5 53 five
6 54 six	7 55 seven	8 56 eight	9 57 nine	: 58 colon
; 59 semicolon	= 61 equal	? 63 question	@ 64 at	A 65 A
B 66 B	C 67 C	D 68 D	E 69 E	F 70 F
G 71 G	H 72 H	I 73 I	J 74 J	K 75 K
L 76 L	M 77 M	N 78 N	O 79 O	P 80 P
Q 81 Q	R 82 R	S 83 S	T 84 T	U 85 U

[91 bracketleft	] 93 bracketright	` 96 grave	a 97 a	b 98 b
c 99 c	d 100 d	e 101 e	f 102 f	g 103 g
h 104 h	i 105 i	j 106 j	k 107 k	l 108 l
m 109 m	n 110 n	o 111 o	p 112 p	q 113 q
r 114 r	s 115 s	t 116 t	u 117 u	v 118 v
w 119 w	x 120 x	y 121 y	z 122 z	¡ 161 exclamdown
¯ 175 macron	´ 180 acute	¸ 184 cedilla	¿ 191 questiondown	Æ 198 AE
Ø 216 Oslash	Œ 338 OE	ˇ 711 caron	˘ 728 breve	˚ 730 ring
ˆ 770 uni0302	˜ 771 tildecomb	˙ 775 uni0307	¨ 776 uni0308	˝ 779 uni030B
̷ 823 uni0337	Γ 915 Gamma	Δ 916 Delta	Θ 920 Theta	Λ 923 Lambda

Ψ 936 Psi	Ω 937 Omega	– 8211 endash	— 8212 emdash	‘ 8216 quoteleft
“ 8220 quotedblleft	” 8221 quotedblright	ff 64256 uniFB00	fi 64257 uniFB01	fl 64258 uniFB02
ffi 64259 uniFB03	ffl 64260 uniFB04	□ 983040 .notdef		

Problems

- `end_program()` too complicated
- polygonal approximation of the pens create many curves

$\Rightarrow$ change strategy

end_program() too complicated

- process contours, envelopes and pen's curves separately

- less functions, but with more code

```
1) _remove_useless_curves
2) _simplify_curves
3) _remove_loops
4) _merge_envelopes_and_pens
5) _merge_envelopes_and_contours
6) _simplify_merged_curves
7) _build_cycle
```

Plus:

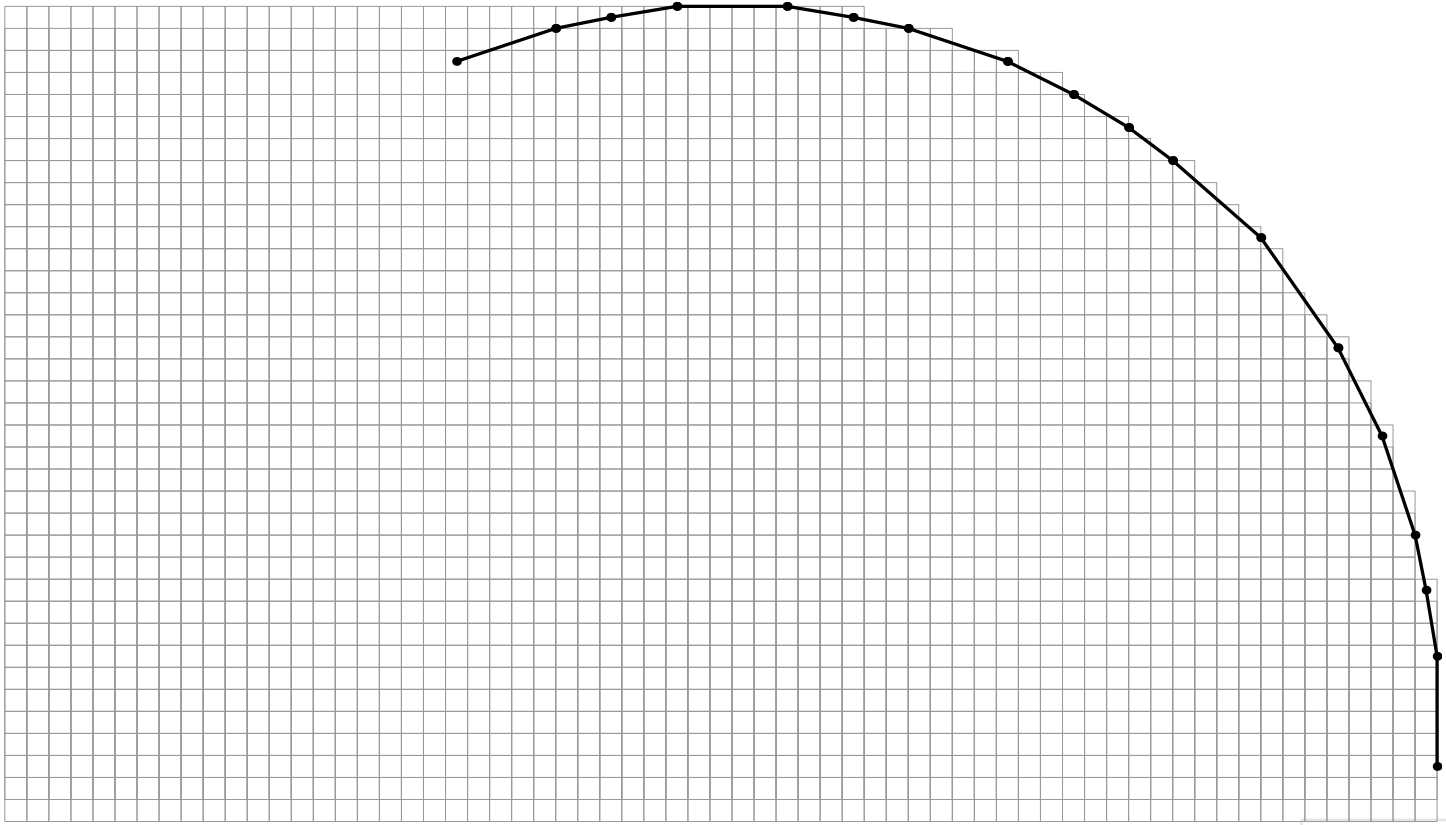
```
_manually_remove(valid_curves_e,{15 }) – can be a function
_dump_curves(final_curves,'final_curves.lua')
```

end_program() too complicated

- implement in Lua:
  the De Casteljau's algorithm ($x(t), y(t)$, left(t), right(t));
  De Casteljau's bisection algorithm (trace a curve);
  intersection of two curves;

- `_exec_fixes(final_curves, _sf("fix_files/fix_%04d.lua",index))`

polygonal approximation of the pens create many curves



polygonal approximation of the pens create many curves

METAFONT takes `majoraxis`, `minoraxis` and the angle of rotation `theta` from the pen's specification and then calls the `make_ellipse` subroutine. If we put a sensor around it, we can store the axis and theta into a Lua table.

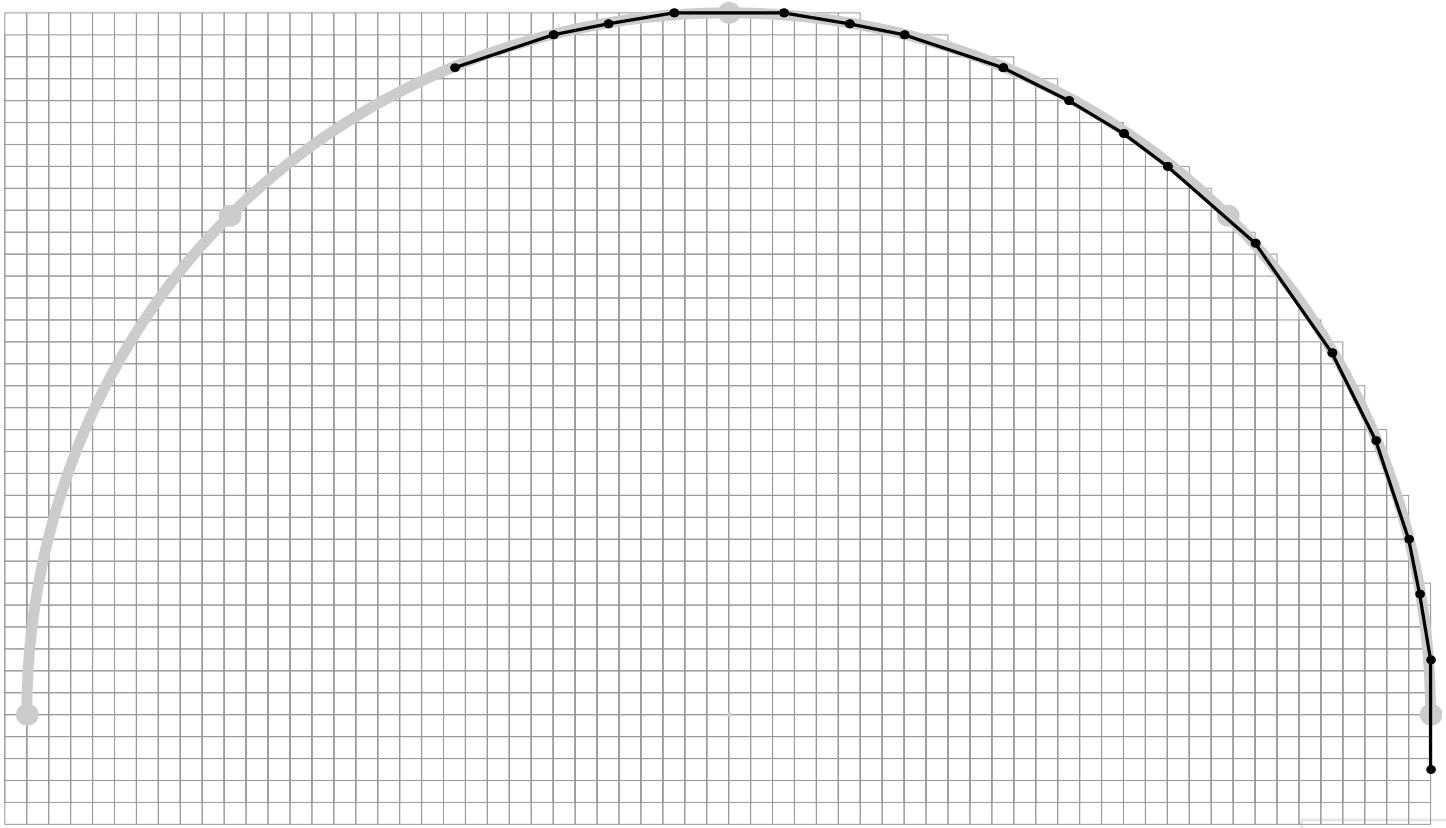
Trick: call MFLua with the pen's spec.

```
batchmode;
fill fullcircle
    xscaled (majoraxis) yscaled (minoraxis) rotated (theta)
shifted (0,0);
shipit;bye.
```

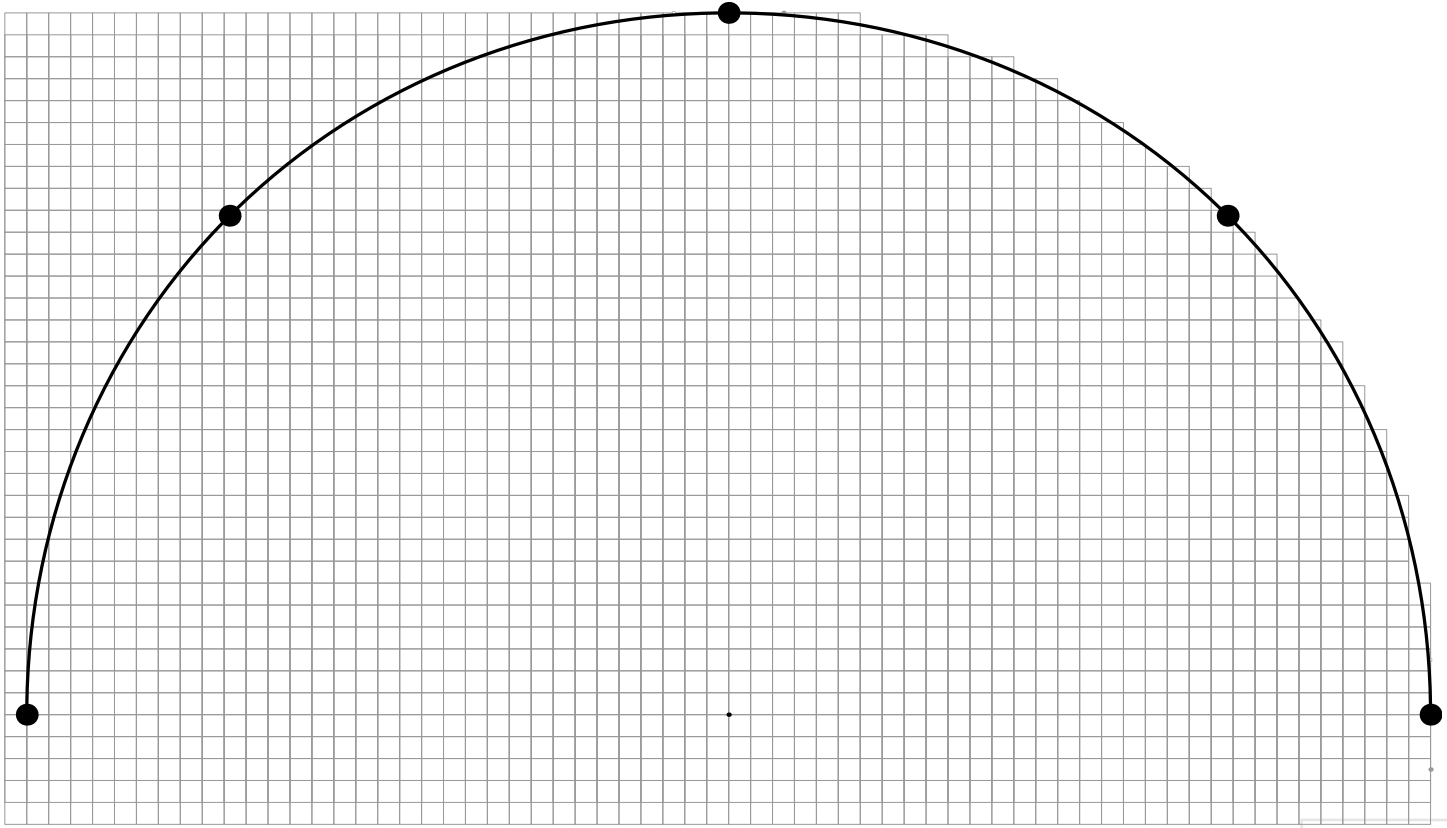
polygonal approximation of the pens create many curves

and then parse the log to read the curves. Put the curves in place of the polygonal:

polygonal approximation of the pens create many curves



polygonal approximation of the pens create many curves

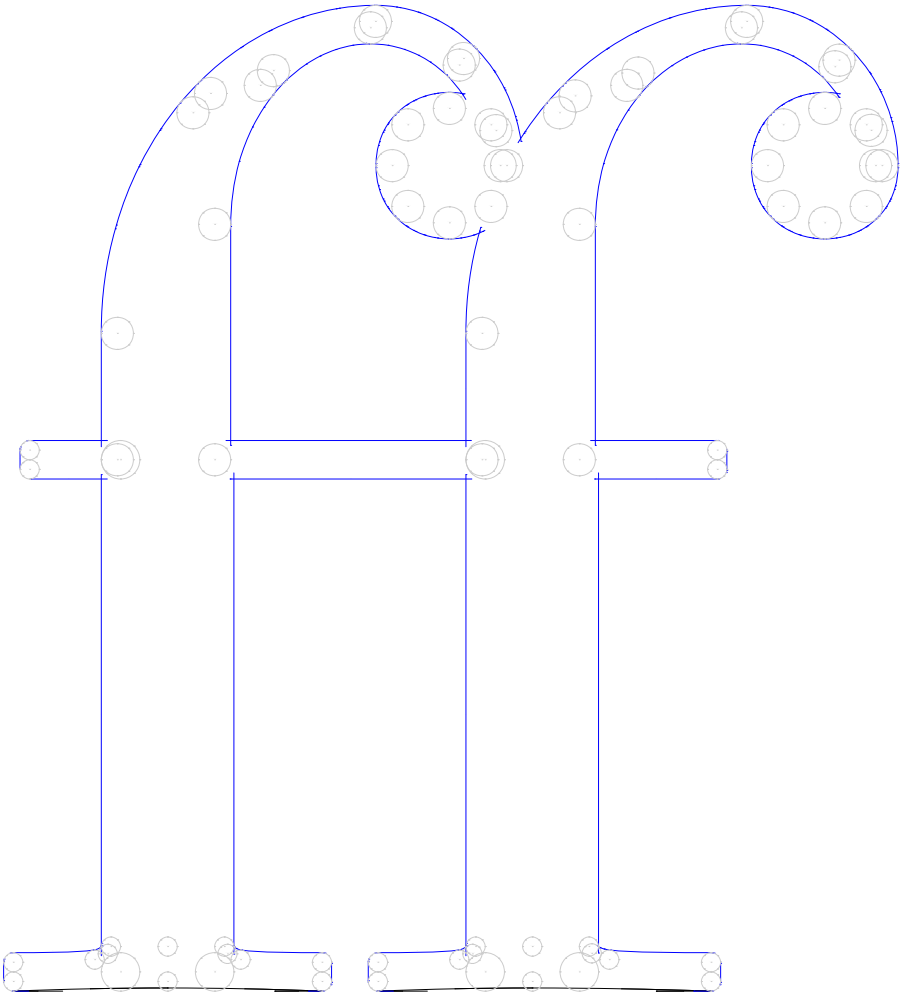


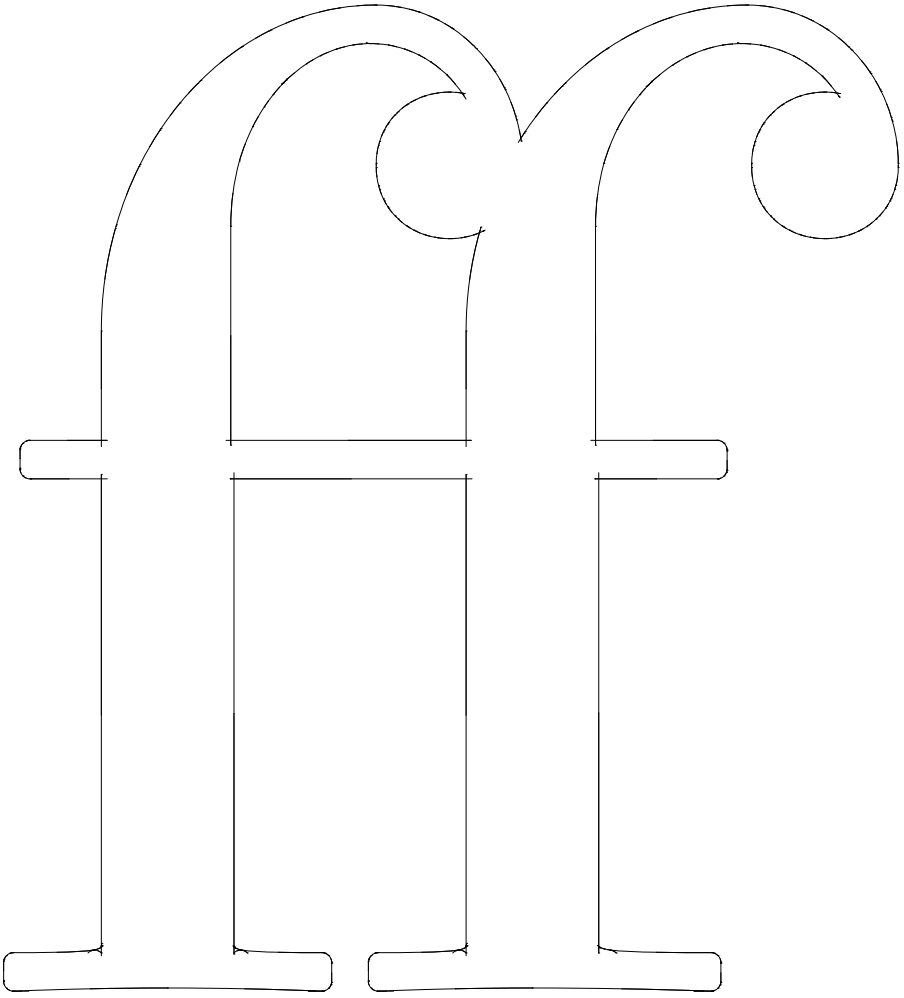
polygonal approximation of the pens create many curves

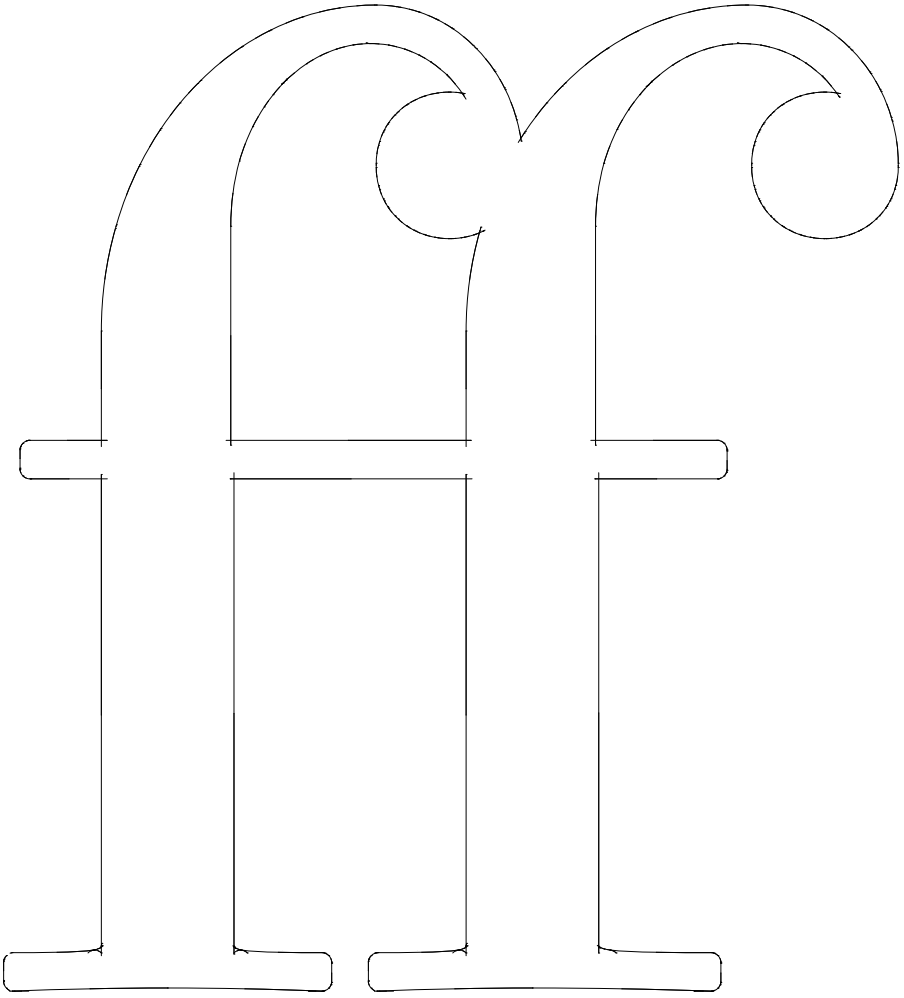
Problems:

- find the point where to place the center of the ellipse
- manage intersections

(see next pictures)







Still many curves: why?

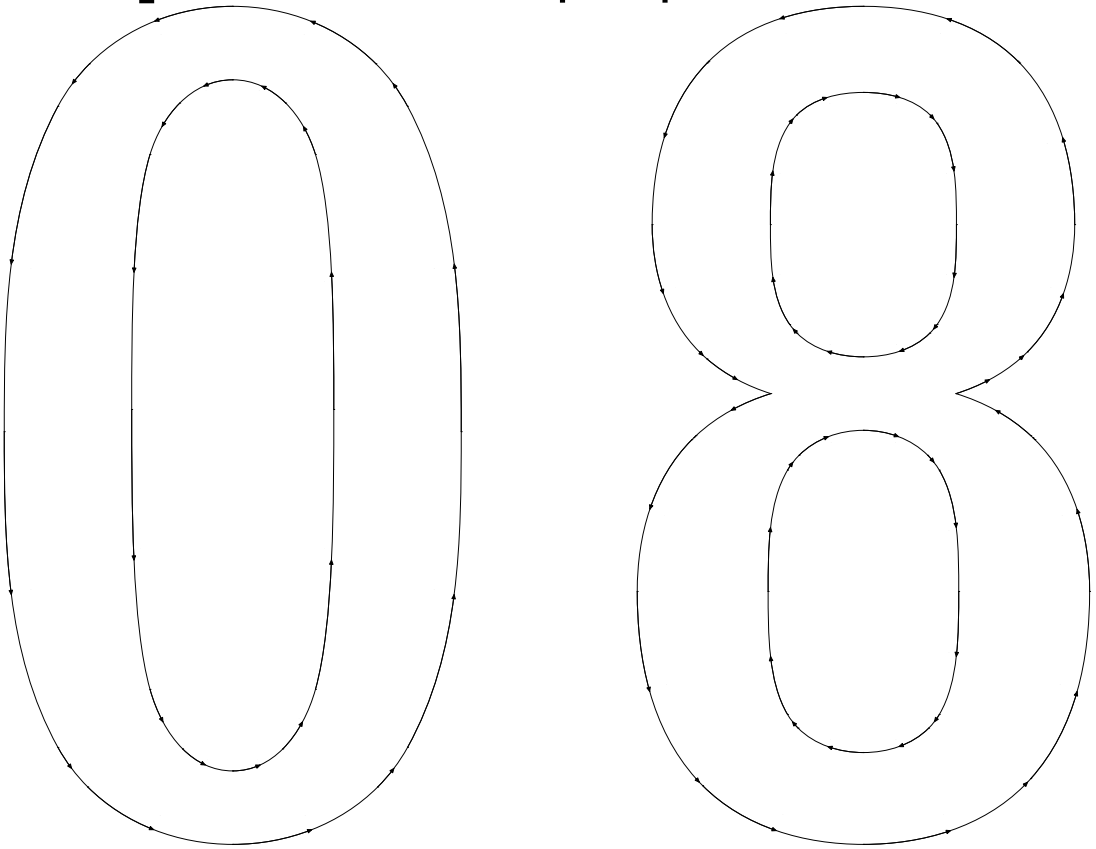
It's the pen:

"Given a convex polygon with vertices $\mathbf{w_0}, \mathbf{w_1}, ..., \mathbf{w_{n-1}}, \mathbf{w_n} = \mathbf{w_0}$ a in counterclockwise order ...(and a curve $\mathbf{B(t)}$) the envelope is obtained if we offset $\mathbf{B(t)}$ by $\mathbf{w_k}$ when the curve is travelling in a direction $\mathbf{B'(t)}$ lying between the directions $\mathbf{w_k - w_{k-1}}$ and $\mathbf{w_{k+1} - w_k}$. At times t when the curve direction $\mathbf{B'(t)}$ increases past $\mathbf{w_{k+1} - w_k}$, we temporarily stop plotting the offset curve and we insert a straight line from $\mathbf{B(t) + w_k}$ to $\mathbf{B(t) + w_{k+1}}$; notice that this straight line is tangent to the to the offset curve. Similarly, when the curve direction decreases past $\mathbf{w_k - w_{k-1}}$, we stop plotting and insert a straight line from $\mathbf{B(t) + w_k}$ to $\mathbf{B(t) + w_{k+1}}$; the latter line is actually a "retrograde" step which will not be part of the final envelope under the METAFONT's assumptions. The result of this construction is a continuous path that consist of alternating curves and straight line segments."

METAFONT: The Program, chapter 469.

Still many curves: why?

It's not a problem with penpos:



Conclusion

- the pens cause a large number of curves, post-processing is complicated
- it is more easy if a glyph has only contours
- FontForge is still indispensable (as a program or a module)
- currently work on
  `https://github.com/luigiScarso/mflua` on
  `/tree/master/test/xmssdc10`
  only to produce the outlines.

That's all
Thank you !