HOWTO: Fancy TTF and stick TTF fonts

(font toolchain and Fontfilter)

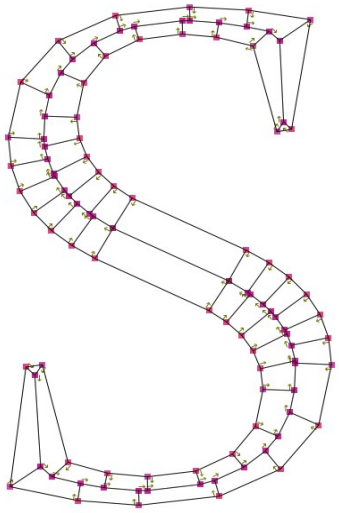
Ver. 1.0

The problem

For many applications (engraving, laser cut etc.) are required “stick” TTF fonts (see note 2) but on web we can found only few of such fonts. Viceversa we can found many Open Source or Public Domain standard TTF fonts (see resources).

*This **tollchain** and the **Fontfilter** java program are created to experiment with TTF fonts, to produce new fancy TTF font or font family, and as help to transform TTF fonts to stick TTF fonts.*

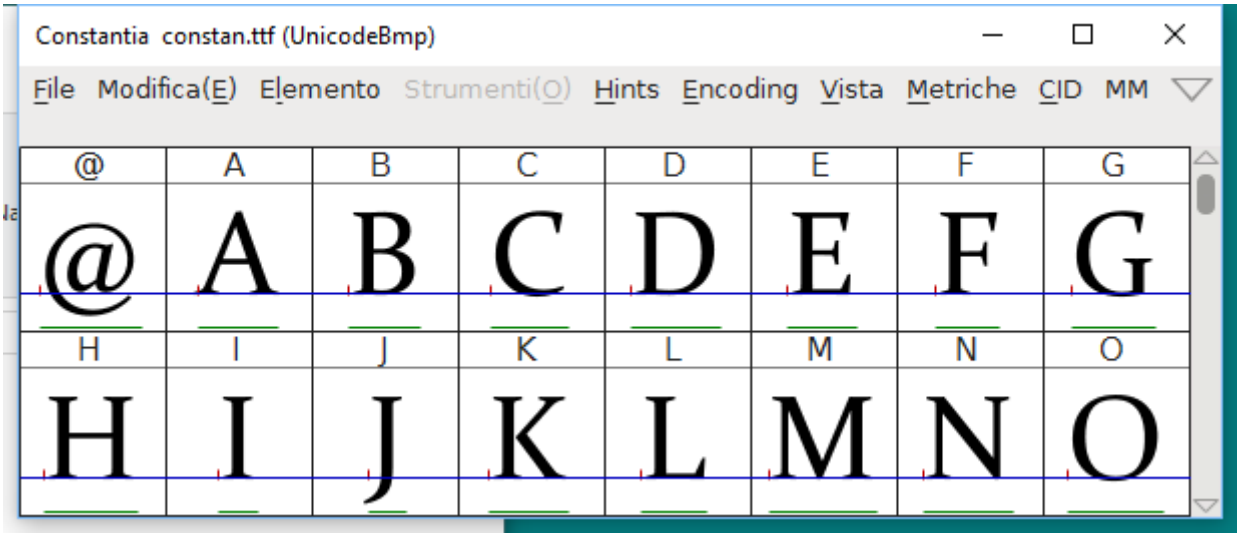
The **Fontfilter** can process all glyphs presents in a font, so it is very fast and it is a great help on design new fonts.



The 'S' SKELETON

Features

The test font: **Constantia**, here inside *FontForge*.



Transformation: biquadratic.

$$x_1 = f(x,y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2$$
$$y_1 = g(x,y) = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2$$

So 12 number defines a biquadratic transformation. This transformation can alter the glyphs slant, size, shape etc.

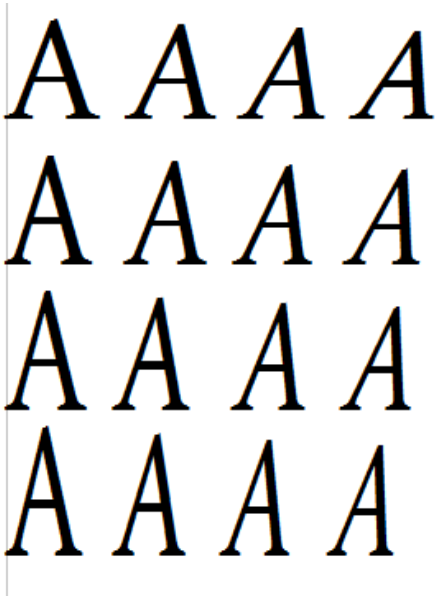
The *Fontfilter* offers 2 options:

Function QUAD: the same biquadratic transformation is applied to all glyphs in the font. For production.

Function MANYQUAD: different biquadratic transformations are applied to specific glyphs. For tests.

Here the 'A' with 4 slant (0..0.3) and 4 narrow (1.. 1.6) steps, in total 16 different biquadratic transformations

Top left the original glyph.



Transformation: not linear

$$x_1 = f(x, y)$$
$$y_1 = g(x, y)$$

Where *f()* and *g()* are expressions in JavaScript, valued at runtime. For the math functions (square root, trigonometric etc.) allowed see `Dmath.java`.

In expressions are allowed also control structures and multiline code, parameter definitions etc.

note: the last expression must be a pure expression, not an assignment statement: its value is the result.

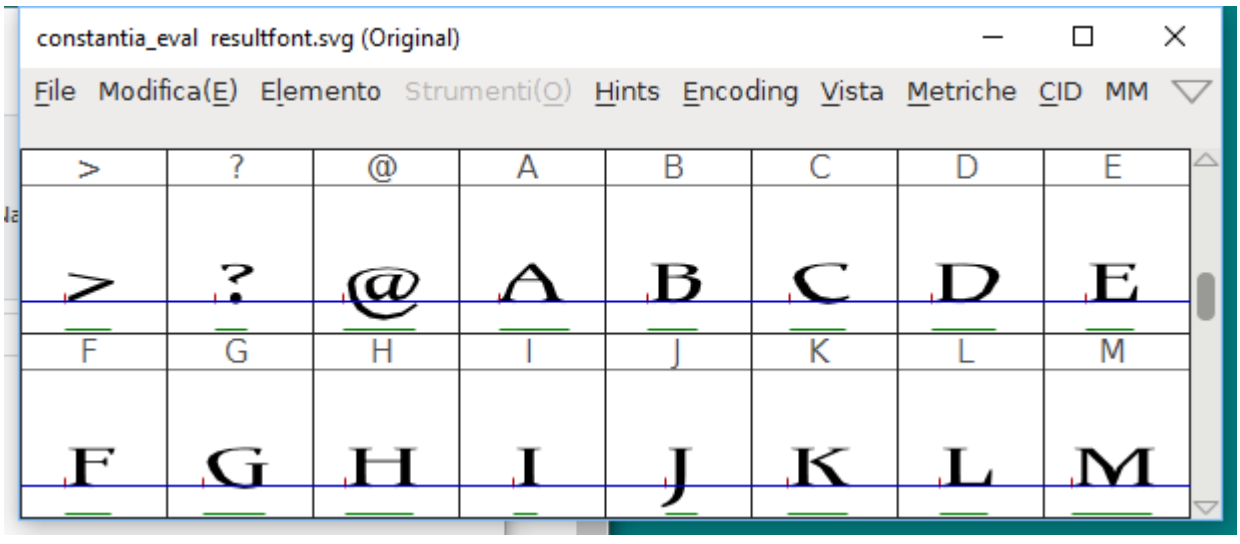
***Function EVAL*:** the same 2 functions are applied to all glyphs in the font.

Example 1:

f: x

*g: h=6; (y > h)? (h + (y-h)*0.40): y*

This function reduces 'y' of 60% when 'y' is bigger than the parameter h (6). The result is the vertical compression of the top of glyphs.



Example 2:

*f: r=25; x+(r-Math.sqrt(r*r-y*y))*

g: y

This function transforms a vertical line in a circle arc to right:

Note: EVAL is a slow transformation, for linear functions better to use QUAD function.

transformation SKELETON

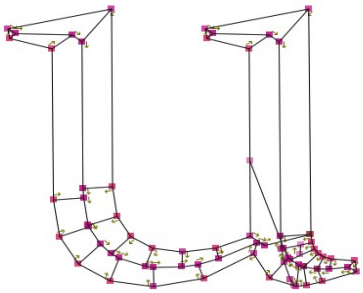
The goal of this transformation is to get *2 lines stick TTF fonts* (see note2)

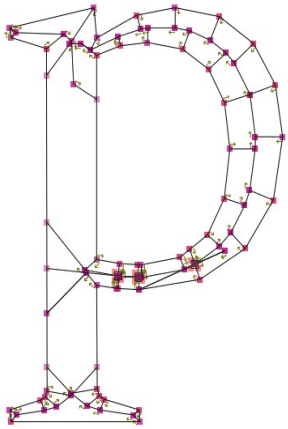
Very experimental, it can makes mistakes on complexes glyphs !! But I think it is still a great help, I hope to get better results in future versions.

As first step, this transformation try to find the ***skeletron*** of all gyphs.

Function SKELETON, output TEST

The output shows (in ***FontForge***) the found skeleton, for 'u' glyph, complete and correct.





In this case ('p' glyph) the skeleton is imperfect (missed the vertical segment).

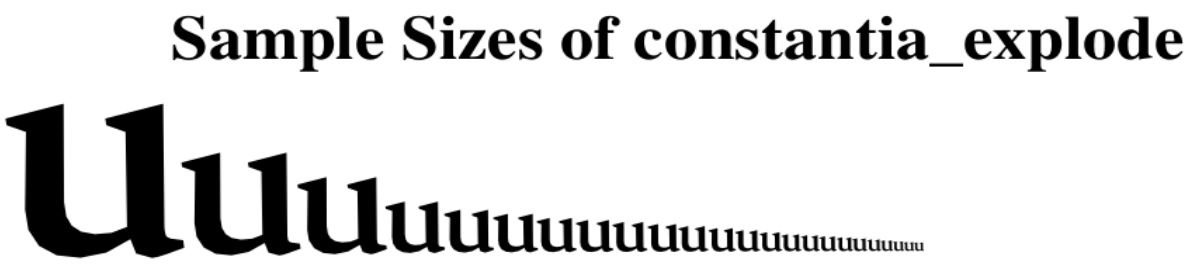
But it is useful... see later

The TEST output is used only on development or for documentation purpose.

Function SKELETON, output SHRINK

This is for bold or light version of the original font: a factor controls the variation.

Bold 'u' (1.3 factor)



Light 'p' (0.8 factor)

Sample Sizes of constantia_explode

Ppppppppppppppppppppppp

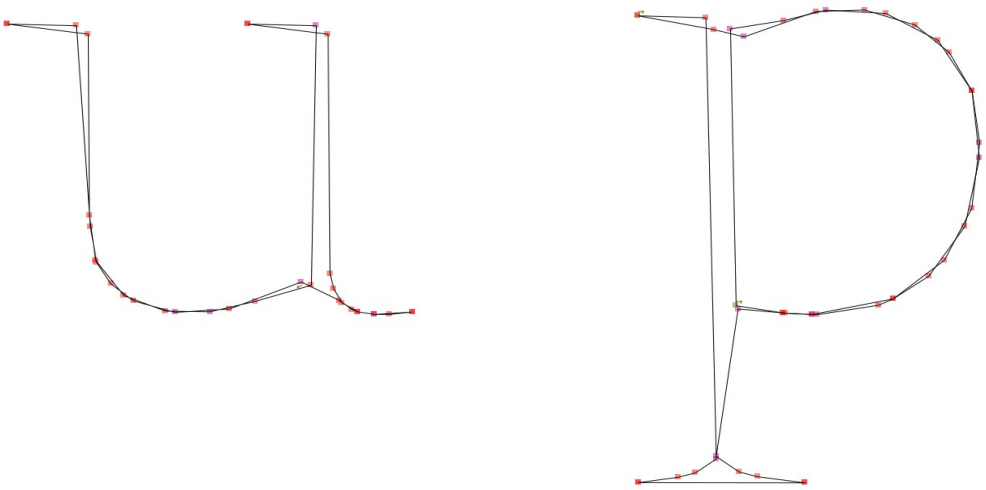
This 'p' is good, also if the skeletron was not perfect (see before, TEST output)

Function SKELETON, output LINE

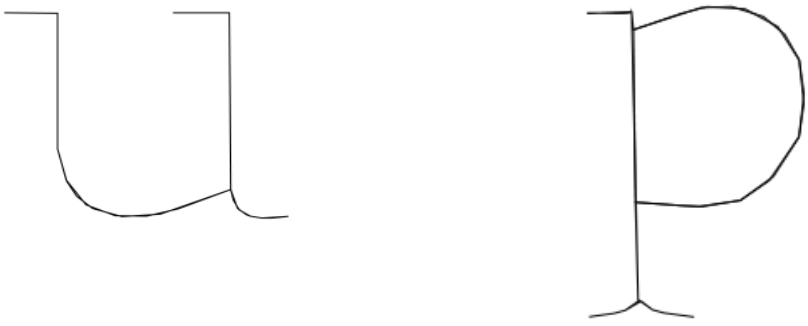
This output is the base for the creation of *2 lines stick TTF fonts*.

Some manual tuning is required, but it is very simple and limited.

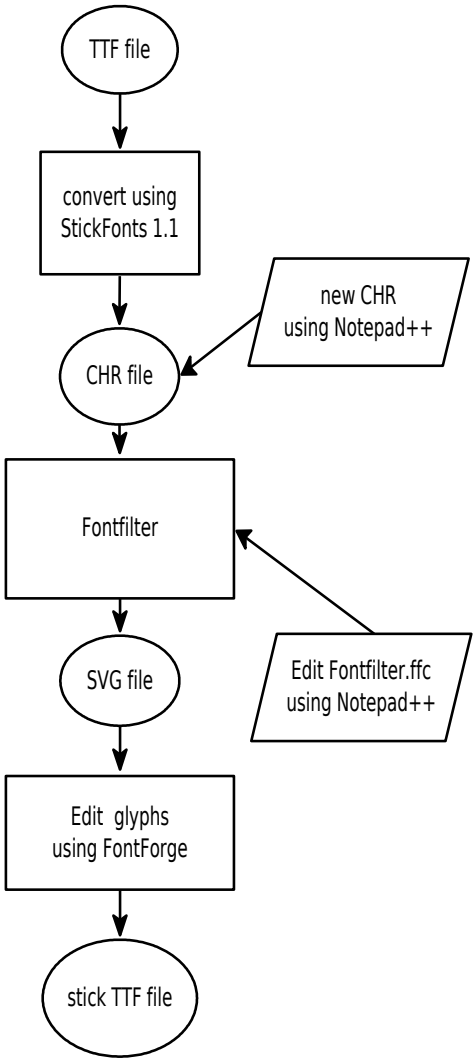
This is the original output for 'u' and 'p' gyphts:



After the manual tuning in *FontForge ,* these are the resulting glyphts, ***2 lines stick TTF font***.



General notes on toolchain (windows)



1. TTF fonts uses Bezier curves and splines. To simplify the filter task, the TTF font is linearized in the format **.CHR**. This is done using ***StickFonts v. 1.1.*** (free – see note 1)

   note: The linearization of TTF reduces the quality of font, mostly at big sizes. Two parameters (`smallLine` *and* `longLine`*) on config file allows some control on the number of points used..*

2. One more advantage using CHR format is that this is a simple text format, so it is very easy to create custom fonts, logos or icons, only writing the points coordinates (see note 3)

3. The ***Fontfilter*** requires a file CHR as input and produces an output in CHR or SVG formats. The filter as not an user interface, it gets options form a configuration file (file `fontfilter.ffc`, edit it using *Notepad++*)

4. The resulting SVG file can be open using ***FontForge,*** a free program with graphic interface, for manual tuning of glyphs. *FontForge* can export fonts in many formats: PS, SVG, OpenType, etc. The most important for us is TTF.

   note: in ***FontForge*** *it is also easy to recreate (if you wont) the Bezier curves: select 3 points, then* menu:modify:merge.

5. The resulting TTF font can be used in any application. Place it in `/Windows/fonts` directory.

Downloads with installation instructions are here: https://github.com/msillano/FontFilter

Note 1 – Use the point as decimal separator.

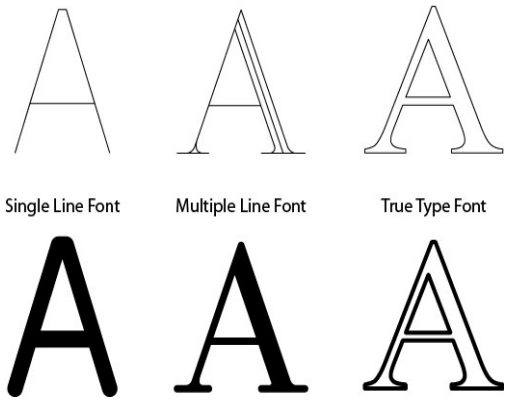
Many countries uses the comma (“,”) as decimal separator.
Some programs do not take into account local settings and expect instead the period (.).
In particular, ***StickFonts v. 1.1.*** can use commas as separator (from local settings), and that make the output CHR file unusable (see note 3).
You can change the CHR file using an ASCII editor and changing all the commas (,) in points (.), but because commas are used also as separator, that is not easy.
We can proceed in another way: to get numbers that use the period (.) as the decimal separator we must change just temporarily the settings for the local language.

HOWTO:

1. Go to Control Panel / Clock, Language and Region / Language - Change date, time or number formats.
   or:
   Control Panel / All Control Panel Items / Language / Change date, time or number formats.

2. In Additional settings choose Decimal "." (Period).
3. Remember at end to restore the local standard.

Note 2 – Font True Type



True Type fonts always consist of one or more closed paths, with the interior filled.

For the incision lines are used instead.

Using for engraving normal TTF fonts, it is used the outline of the glyph and then you always get a double perimeter inside / outside.

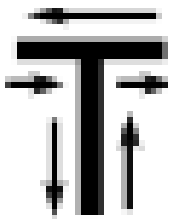
However, there are special TrueType fonts designed just for cnc engraving, for laser cutter etc.

TTF stick

These special TTF are formed by a closed line that runs through the glyph twice, thus obtaining a route suitable for linear incision.

PRO:

- Stick fonts are TTF fonts standard, usable in every application
- There are versions in 1 line, 2 lines, 3 lines ... to simulate the filling of wide areas



AGAINST:

- The double passage of the bit doubles the running time and can fail.

In the figure, you can see how looks a Stick fonts (using Inkscape):

above: Filling: yes, outline: No (Objects menu / Fill, Outline)

under: Filling: no, outline: yes

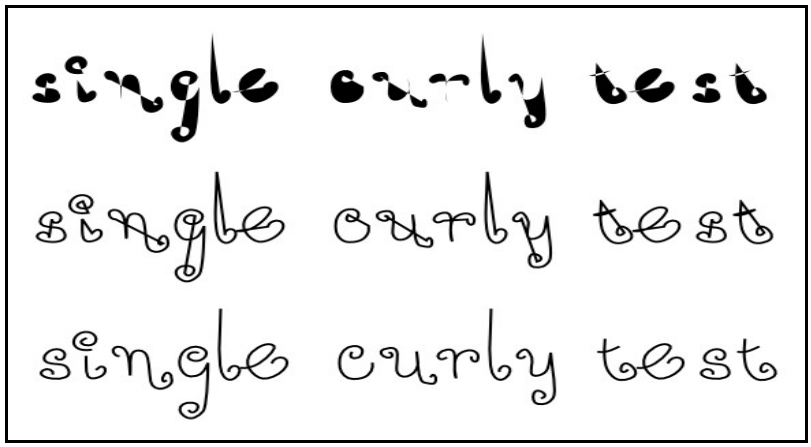
This font are also called "Single Line Fonts with 2 lines".

example

Single line open path

These TTF fonts contain closed paths which must be "open" with the elimination of the last segment. This additional step can be performed in a vector graphics program, after transforming the text in graphic objects and after ungrouping objects. For Inkscape exists a special automatic extension reopenSingleLineFont

In the figure, appearance of a single line open font (using Inkscape):



1. filling: yes, outline: No (Objects menu / Fill, Outline)
2. filling no, outline: yes
3. After removed the closing segment

example

PRO

- true font with single line

AGAINST:

- Requires an extra operation executable only in a vector graphics program
- After cutting we get paths, not more fonts. (I.e., the cut must be the last action, to be able to take advantage of "text" features in the graphic programs)
- Few fonts in the Web.

Note 3 – CHR files

The font files are in an ASCII format and can be edited with any text editor. Each line in the file describes one character. Each character is made up of a series of coordinate pairs that are the endpoints of the lines that make up the character. The coordinates are based on a grid that is 21 units high and variable in width. A character that is 21 units high will scale to be whatever you

enter as the text height.

A typical character in the .CHR file looks like this:

CHR_41 18; 1,0 9,21 17,0; 4,7 14,7

Starting from the beginning of the line:

CHR_41

The first four characters "CHR_" should always be the same. The next two characters are the characters ASCII code in hexadecimal.

18;

The remaining values are all in decimal. The first value after the character code is the characters width followed by a semicolon. This value should include any padding space before and after the character.

1,0

The first coordinate pair after a semicolon is the start point of a line segment.

9,21

This coordinate pair is the endpoint of the first line segment.

17,0;

The line continues from the last coordinate pair to this one. The semicolon indicates that it is the last point in this line segment.

4,7

This point is the start point of a new line segment.

14,7

This point is the endpoint of the line segment and the end of the character.

(from 'create fonts.txt' by NCPlot Software LLC)

NOTE: A file CHR that will be converted to TTF must contain only close paths, i.e. first point equal to last.

References:

The previous report: **HOWTO engrave text using 3Drag/K8200**
(http://www.thingiverse.com/thing:1235469)

Algoritm:

- Motorcycle graphs and straight skeletons: (2002. SW Cheng)
  http://dl.acm.org/citation.cfm?id=545401
- Computing Straight Skeletons of planar Straight-Line graphs Based on Motorcycle Graphs (20010, Stefan Huber, Martin Held)
  http://cccg.ca/proceedings/2010/paper50.pdf

Fonts:

This is the critical point, with relatively few viable solutions for stick TTF fonts:

- font TTF stick download.
- single line open font TTF download.

- Free font TTF: Google, Fontspace, 1001fonts, daFont
  (note: take care at license type of free fonts)
- Tutorial *Single line open* TTF fonts using Inkscape
- *Font stick for sell: starting from 10$ (example)*

.

Windows:

- ***Notepad++*** *(ASCII editor) can be found here:* https://notepad-plus-plus.org/
- ***StickFonts v. 1.1.*** *can be found here:* http://www.ncplot.com/stickfont/stickfont.htm
- ***FontForge*** *can be found here:* https://fontforge.github.io/en-US/
- ***Inkscape*** *can be found hrere:* https://inkscape.org/en/
- *WIN 10: To get back a programs menu with all applications, two-level, I installed and find very useful: "**classic start menu**"* http://www.classicshell.net/