HOWTO: Fancy TTF and stick TTF fonts

(toolchain e filtro FontFilter per font TTF) Ver. 1.0

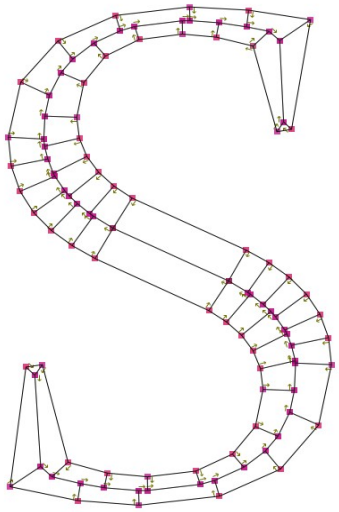
Il problema

Per molte applicazioni (cnc, incisione, laser, etc) sono richiesti i cosiddetti "**stick fonts TTF**" (vedi note 2) ma si trovano veramente pochi di questi font sul web.

Viceversa si trovano molti font TTF, anche con licenze Open Source o Public Domain (see resources).

*Questa **toolchain** ed il filtro **FontFilter** sono stati creati per sperimentare con i font TTF, per creare facilmente nuovi font o una famiglia di font, ma soprattutto come aiuto nella trasformazione di font TTF standard in "stick TTF ".*

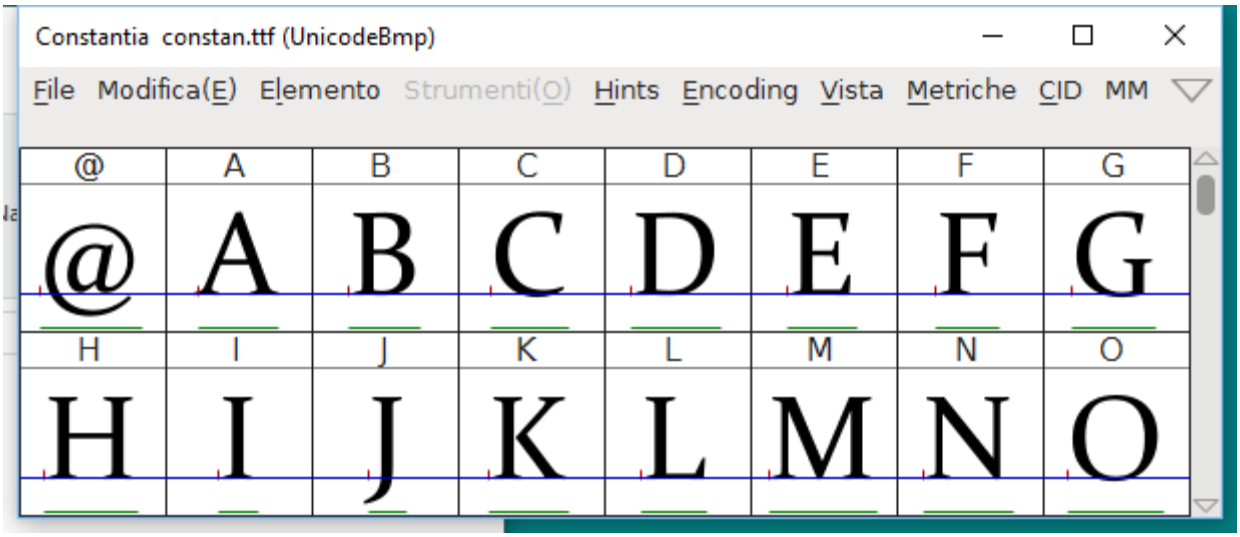
Il fitro può modificare tutti i glifi presenti in un font in un solo passo, risultando quindi veloce e di grande aiuto nella creazione di nuovi font TTF.



'SKELETON' di 'S'

Prestazioni

Il font usato nei test, **Constantia**, qui visto con ***FontForge***:



Trasformazione: biquadratica.

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

Quindi 12 numeri definiscono una trasformazione biquadratica, Questa trasformazione può alterare i glifi nella dimensione, nell'inclinazione etc.

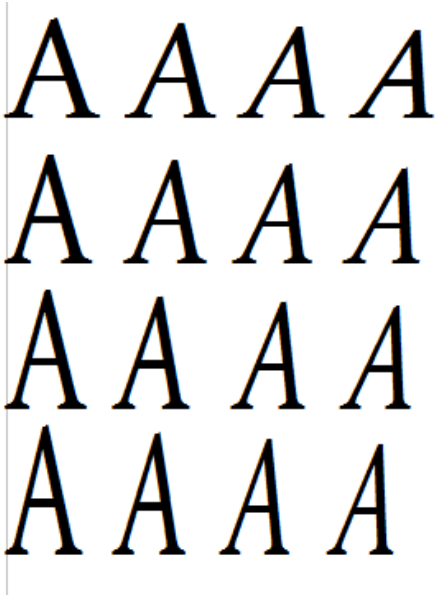
FontFilter *offre due opzioni*:

Function QUAD: La stessa trasformazione biquadratica è applicata a tutti i glifi. *Per produzione.*

Function MANYQUAD: Trasformazioni biquadratiche differenti sono applicate a specifici glifi. *Per test.*

Nell'esempio a lato, la lettera 'A' con quattro altezze e quattro inclinazioni differenti.

L'originale è in alto a sinistra.



Trasformazione: non-lineare

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

In questo caso *f()* e *g()* sono espressioni matematiche in JavaScript, valutate al runtime. Per le funzioni matematiche usabili (radici quadre, funzioni trigonometriche etc,) vedere `Dmath.java`.

Nelle espressioni sono ammesse strutture di controllo, espressioni multilinea, definizioni di parametri etc.

nota: l'ultima espressione deve essere una pura espressione matematica, non un'assegnazione: il valore calcolato è il risultato.

Function EVAL**:** Le stesse due funzioni sono applicate a tutti i glifi presenti nel font.

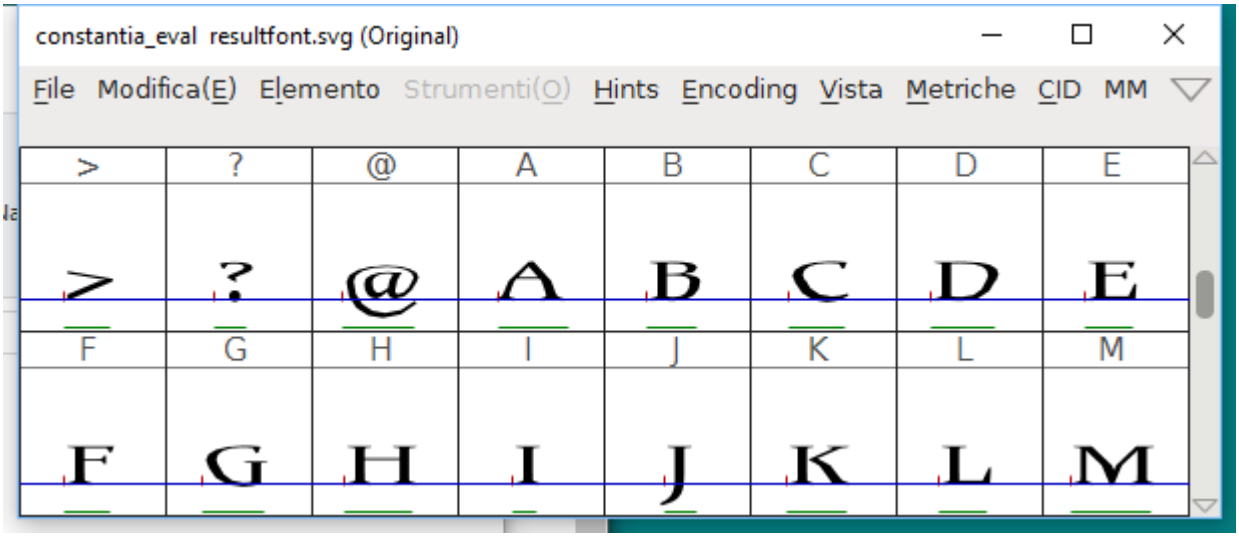
Esempo 1:

f: x

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

Questa funzione riduce la *'y'* del 60%, solo quando *y* supera il valore del parametro *h* (6).

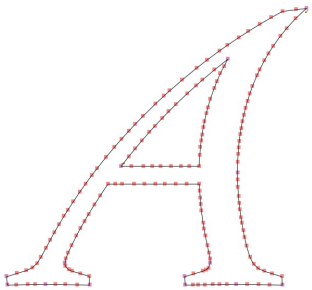
L'effetto é una compressione nella parte superiore dei caratteri.



Esempio 2:

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

g: y



Questa funzione trasforma una linea verticale in un arco di cerchio, verso destra.

Nota: la trasformazione EVAL è lenta: quando possibile preferire QUAD

trasformazione SKELETON

L'obiettivo di questa trasformazione è quello di ottenere *"***2 lines stick TTF fonts***"* (vedi note2)

N.B. Sperimentale, può essere inesatto in caso di glifi complessi. !

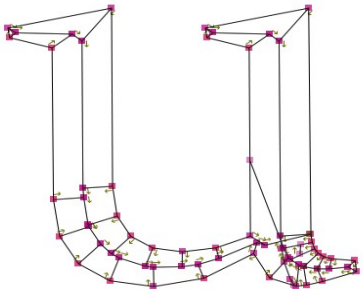
Ritengo che anche all'attuale stato di sviluppo possa rappresentare un utile aiuto. Auspichiamo di ottenere risultati migliori in versioni successive.

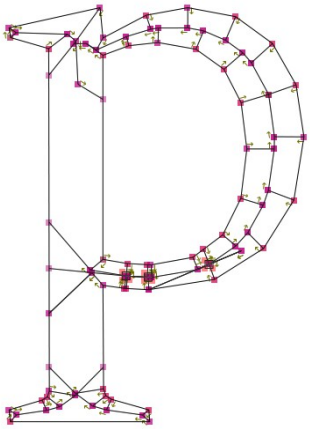
Come primo passo, questa funzione cerca di trovare lo 'skeleton' di tutti i glifi.

Function SKELETON, output TEST

Questo output presenta (in *FontForge*) lo 'skeleton' trovato dal filtro.

Nell'esempio 'skeleton' della 'u'. E' completo e corretto.





In questo caso (glifo 'p') lo 'skeleton' è incompleto (manca un segmento verticale).

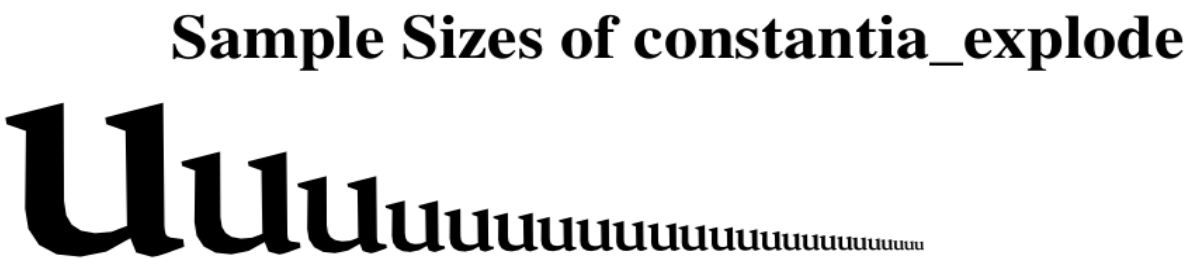
Ma è comunque utilizzabile: vedi oltre.

L'output TEST è utile solo in fase di sviluppo o per documentazione.

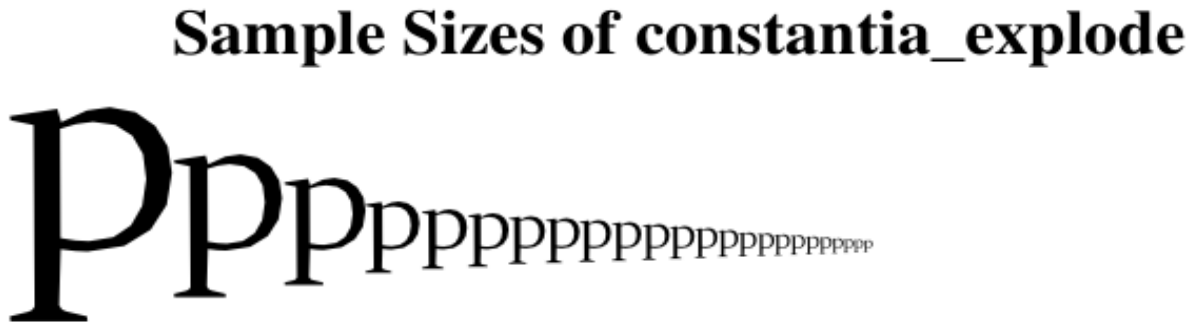
Function SKELETRON, output SHRINK

Questo output permette di ottenere versioni 'bold' o 'light' del font originale, in funzione di un fattore numerico.

Bold 'u' (fattore 1.3)



Light 'p' (fattore 0.8)



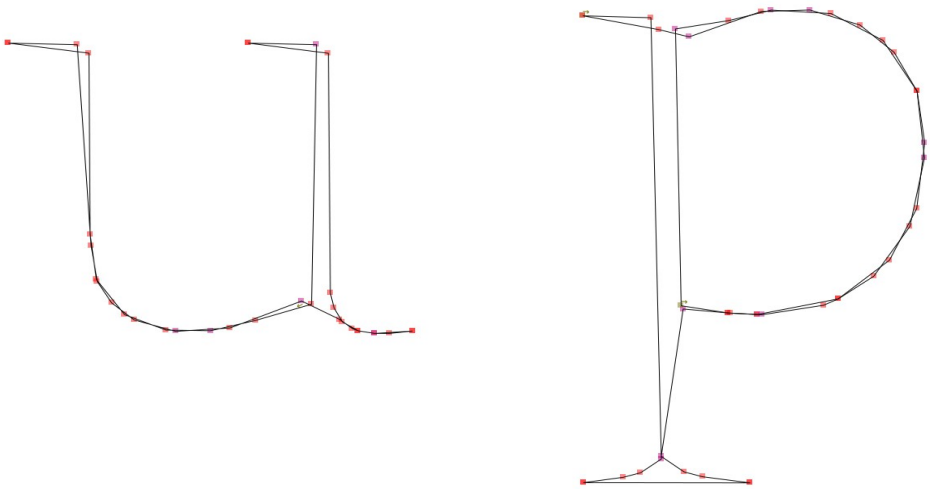
Il risultato per la 'p' è comunque valido, anche se, come visto, lo skeleton della 'p' presentava un difetto.

Function SKELETRON, output LINE

Questo output è la base per la creazione di font *"2 lines stick TTF".*

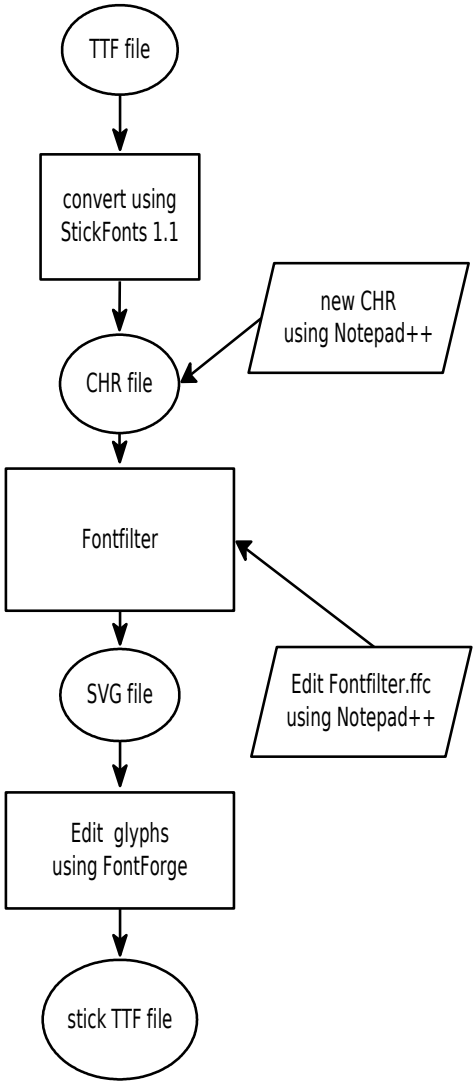
E' sempre richiesto qualche aggiustamento a mano, in *FontForge*, ma questi risultano, in generale, di semplice e rapida esecuzione.

Questo è l'output ottenuto per i glifi 'u' e 'p':



Dopo i ritocchi a mano, in *FontForge* (essenzialmente spostamento di punti) questi sono i due glifi finali: ***2 lines stick TTF font***

Note generali sulla toolchain (windows)



1. I font TTF usano curve di Bézier. Per semplificare il filtro, i font TTF sono linearizzati nel formato CHR. Questo step è eseguito da **StickFonts 1.1** (free – vedi note 1)

   Nota: la linearizzazione peggiora la qualità del font, soprattutto ad elevate dimensioni, Due parametri (`smallLine` e `longLine`) nel file di configurazione permettono un certo controllo nel numero di segmenti usati.

2. Un ulteriore vantaggio nell'uso del formato CHR è che si tratta di un semplice formato ASCII. Questo rende oltremodo agevole creare i propri font, logo o icone, semplicemente scrivendo le coordinate dei punti (vedi note 3).

3. I filtro **FontFilter** richiede come input un file CHR e produce file CHR o SVG, Il filtro non ha interfaccia utente, prendendo tutte le opzioni da un file di configurazione (file `fontfilter.ffc`, modificarlo usando ***Notepad***++)

4. Il file SVG ottenuto può essere aperto usando **FontForge**, un programma grafico per font, dove i glifi possono essere modificati interattivamente. *FontForge* può esportare font in molti formati PS, SVG, OpenType, in particolare nel formato TTF.

   *Nota: in **FontForge** è anche semplice ricreare (se si desidera) le curve di Bézier (selezionare 3 o più punti, poi menu: modifica: merge).*

5. Il file finale TTF può essere usato in qualsiasi applicazione. Basta copiarlo in `/Windows/fonts`.

Downloads con istrizioni dettagliate qui: https://github.com/msillano/FontFilter

Nota 1 – Usare il punto come separatore decimale.

Le impostazioni della lingua Italiana (e in molte altre nazioni) prevedono la virgola (,) come separatore nei numeri decimali.
In particolare il programma **StickFonts** usa la virgola (usando l'impostazione locale) e questo rende inutilizzabile il file CHR prodotto.
Non si può modificare il file CHR usando un editor ASCII e cambiando tutte le virgole (,) in punti (.), perchè la virgola è usata anche come separatore tra i numeri (vedi nota3), quindi si deve procedere in un altro modo: per ottenere in output numeri che utilizzano il punto (.) come separatore decimale basta cambiare temporaneamente le impostazioni per la lingua italiana.

HOWTO:

1. Raggiungere `Pannello di Controllo/Orologio e opzioni internazionali – Cambia data ora e formato dei numeri`
   oppure:
   `Pannello di Controllo/Tutti gli elementi del Pannello di Controllo/Lingua/Cambia data ora e formato dei numeri`

2. In `Impostazioni aggiuntive` scegliere `Separatore decimale` "." (punto).

3. Ricordarsi poi di ripristinare la situazione Italiana standard.

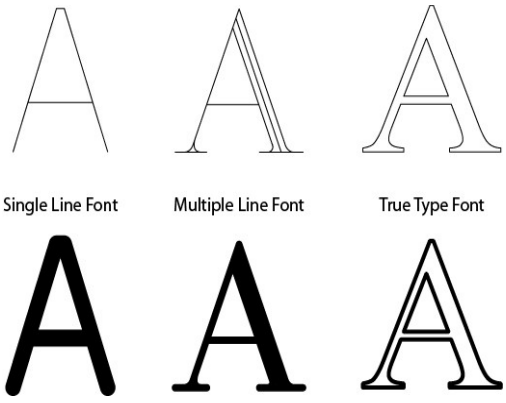
Nota 2 – Font True Type

I **fonti True Type** (TTF) sono sempre formati da uno o più percorsi chiusi, con l'interno pieno.

Per l'incisione invece si utilizzano linee, percorse dall'utensile.

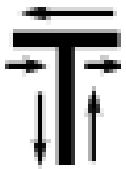
Usando per l'incisione i normali font TTF, viene usato il contorno del glifo e quindi si ottiene sempre un doppio perimetro interno/esterno.

Esistono però speciali font True Type pensati proprio per incisione con cnc, per laser, cutter etc.



TTF stick

Questi speciali font TTF sono formati da una linea chiusa che percorre il glifo due volte, ottenendo così un tragitto lineare adatto all'incisione.



PRO:

- Sono font TTF standard, usabili quindi in ogni applicazione
- Esistono versioni a 1 linea, a 2 linee, a 3 linee... per simulare il riempimento di zone larghe

CONTRO:

- il doppio passaggio della fresa raddoppia il tempo di esecuzione e può provocare difetti.

Nella figura, aspetto di un *Stick font* (in *Inkscape*) :

sopra: riempimento: sì, contorno: no (menu Oggetti/Riempimento, Contorno)

sotto: riempimento: no, contorno: sì

Chiamati anche "Single Line Fonts with 2 lines"
esempio

*L'obiettivo di questa **toolchain** e di **FontFilter** è proprio quello di semplificare la conversione di qualsiasi font TTF standard in font TTF stick.*

Single line open path

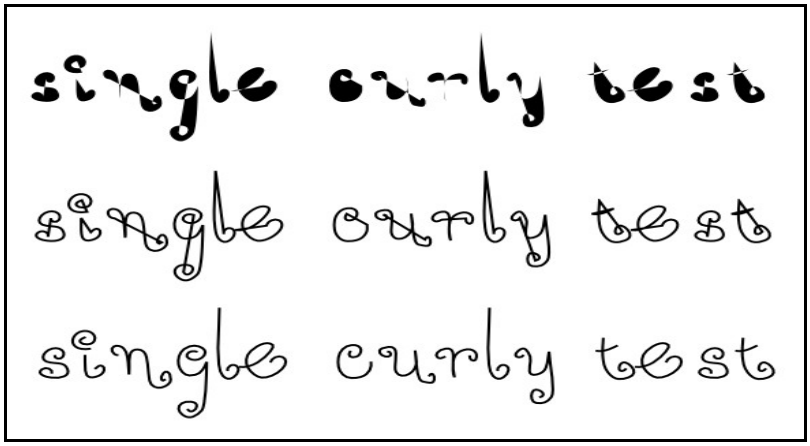
Questi font TTF contengono dei percorsi chiusi che deve essere "aperti" con l'eliminazione dell'ultimo segmento. Questa operazione aggiuntiva può essere eseguita all'interno di un programma di grafica vettoriale, dopo aver trasformato il testo in oggetti grafici e dopo aver diviso gli oggetti. Per ***Inkscape*** esiste la speciale estensione automatica reopenSingleLineFont

Nella figura, aspetto di un *Single line open* font (in *Inkscape*) :

1.riempimento: sì, contorno: no (menu Oggetti/Riempimento, Contorno)

2.riempimento: no, contorno: sì

3.dopo aver eliminato il segmento di chiusura



esempio

PRO

- veri font con linea singola (e non doppia)

CONTRO

- Richiedono un'operazione in più eseguibile solo in un programma di grafica vettoriale
- Dopo il taglio non sono più font ma path. (ovvero, il taglio deve essere l'ultima operazione eseguita per poter sfruttare le funzionalità "testo" dell'editor grafico)
- Pochi font rintracciabili via Web.

Note 3 – formato CHR

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.

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

NOTA: Creando a mano fonti CHR da trasformare in TTF, ogni percorso deve essere chiuso (ultimo punto uguale al primo punto).

References:

Il precedente: **HOWTO engrave text using 3Drag/K8200**
(http://www.thingiverse.com/thing:1235469)

Algoritmi:

- 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:

Un punto critico, si trovano poche soluzioni per stick TTF fonts:

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

- Vendita di *Font stick :* a partire da 10$ (example)
- Tutorial *Single line open TTF fonts* usando Inkscape
- Free font TTF: Google, Fontspace, 1001fonts, daFont
  (nota: attenzione al tipo di licenza dei font free)
  .

Windows:

- ***Notepad++*** *(ASCII editor) si trova qui:* https://notepad-plus-plus.org/
- ***StickFonts v. 1.1.*** *si trova qui:* http://www.ncplot.com/stickfont/stickfont.htm
- ***FontForge*** *si trova qui:*https://fontforge.github.io/en-US/
- ***Inkscape*** *si trova qui:* https://inkscape.org/en/
- *WIN 10: Per riavere un menu programmi con tutte le applicazioni, a due livelli, ho installato e trovato molto utile "**classic start menu**" da http://www.classicshell.net/*