Preliminary report
UDK: 655.24:655.262

Font hinting techniques and the importance of applying these techniques for high-quality display of fonts on the output device screen

Banjanin Bojan[1], Nedeljkovic Uroš[1]

[1]University of Novi Sad, Faculty of Technical Sciences, Department of Graphic engineering and design, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia,

Corresponding author: Banjanin Bojan
Email: bojanb@uns.ac.rs

Abstract:
In the era of contemporary and rapid way of life and with advancing digital technology, the display of electronic content on different types of portable devices becomes a part of everyday life. Whether it is on the screen of a Tablet PC, mobile phone or e-book reader, the font needs to be designed in such a way that the displayed message is received and understood as easy and efficiently as possible.
When it comes to digital font, intended for display on screen, it is necessary to take into account the properties of the output device and font size to be used. Since the text is intended for display on small screens (especially in case of portable devices), the used font should be adapted to such conditions, namely, it should be designed so as to be readable and legible even at small sizes and at different resolutions of the device.
The integral part of contemporary outline fonts are additional instructions on how rasterizer is to render letters at lower resolutions and lower font sizes. These instructions are known as hints, or hint mechanisms, and the process of defining these instructions is called hinting.
The aim of this paper is to provide a theoretical basis for understanding the issues of the display of small sizes fonts on screen. The paper will also elaborate the hinting techniques with emphasis on TrueType hint mechanisms that are most suitable for high-quality display on the output device screen, as well as some methods of automatic hinting. Theoretical basis introduced here, represent foundation on which further exploration will lay on. It is important for broadening the knowledge in the field of rasterization and automatic hinting but also for finding new solutions for the simpler and better hinting.

Keywords: Font rasterization, TrueType hinting, Font scaling, Gridfitting, Outline font

First received: 23.04.2012.
Accepted: 06.06.2012.

Introduction

The most basic categorization of digital fonts is into bitmap fonts and outline (vector) fonts. Bitmap fonts are described in a series of pixels that define the letter character. Unlike bitmap fonts, which are used on older computers and in the memory of older dot matrix printers, the advantage of vector fonts is that they can be used on devices of different resolutions and in different sizes, without having to design separate files for each size and resolution. Whereas with bitmap font, a specially designed set of characters for each size and each section was necessary, with vector font it is sufficient to have one file for each style (regular, bold, italic, ...), which can be used for any desired size.

Given that the output device screen is made up of many elements (pixels) stored in a discrete rectangular grid, each image (including letter characters) is represented by a series of these elements. As for the vector fonts, the form of the letter character (defined by the outline) is filled with pixels of the monitor or raster printer dots.

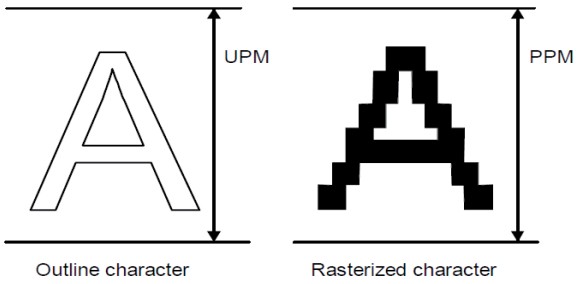

Figure 1. Schematic representation of the outline font (left) and the bitmap font (right)

Output devices must convert the outline record of the letter characters into the bitmap in some way. A separate program, rasterizer, is in charge of that. Its task is to calculate, using the coordinates of the outline, where on the discrete grid of pixels the outline will be set and which pixels will be included, in order to obtain as faithful representation of the letter characters as possible. Since this grid is defined by the integer pixel values, with each change in the size of letter characters, the real coordinates of the outline need to be rounded up to integer values. This process is called *grid fitting*.

For example, if a dot on the outline has the coordinates 80 and 115, during the reduction of the letter sign 6 times, these coordinates change value to 13.3333 and 19.1666. After rounding to integers, the coordinates will be 13 and 19 with errors 0.3333 and 0.1666. This error increases proportionally with the reduction of the outline.

In order to minimize errors while reducing the size of letter characters, font rasterizers use special algorithms that correct the scaled outline in order to get better results on devices with low and medium resolution. To enable this, the algorithms use the instructions built in within the font during design. These instructions are called hints (indications) or hint mechanisms. They define the most important parts of the letter character, such as the thickness of strokes, the position of the key elements of the letter character, the proportion of letter character, as well as the set of rules for modifying the outline.

For a high-quality font that looks good when displayed on the output device screen, even in smaller sizes, in addition to having the well-defined outlines, it is also necessary that the hint mechanisms are properly allocated. This process of defining hints is called *hinting*.

Hint mechanisms

Font formats can generally be divided into two groups: Type1 and TrueType. Type1 font format (also known as the PostScript font) has been developed by the Adobe company and was popular in the publishing industry. This format was developed primarily for applications related to prepress and their final aplication is the use on the printed material.

TrueType format, developed by the Apple company, is primarily intended to equalize the representation of characters on screen and paper, putting the emphasis on the screen display. The outlines which describe the characters within the font are not sufficient for a clear view on the high and low resolution output devices such as printers with the resolution of 300dpi and screens with the resolution of 72 dpi or 96 dpi.

It was necessary to develop techniques for better placement of the outline record and its rasterization within the network of pixels. These techniques have included additional information in the form of rules which determined how certain parts of letter characters would be displayed. These rules, called *hints*, refer to the specific outline and the dots it consists of, defining the width of certain parts of the letter characters, such as base strokes, junctions, serifs, terminals, but also the distance between these parts. The main role of the hints would be to maintain the original characteristics of the letter character, during its scaling and display on the raster network of the output device.

Although both formats contain outlines described with Bézier curves (Type1 with cubic Bézier curves, and TrueType with quadratic Bézier curves), their hinting styles are radically different.

Type1 Hint mechanisms

Hinting Type1 fonts works as follows: the instructions in the form of restrictions are defined, and they control the basic features of the character (the thickness of the horizontal and vertical strokes, serif, terminals, etc.). Rasterizer fills the outline with the pixels of the grid, using the instructions in rather restricted way. Hints defined for Type1 fonts have quite limited control of the letter character rasterization, because the designer can control only the position and thickness of the letter character, with no direct impact on individual pixels.

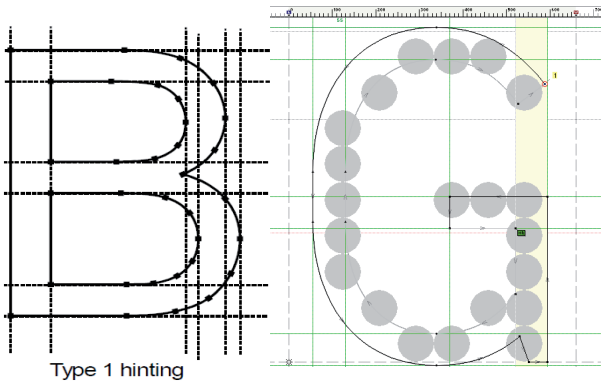

Figure 2. Schematic representation of the Type1 hints of the letter "B" (left) and the display of Type1 hints in the FontLab (right)

Also, the defined instructions for a certain size of the letter character affects the other sizes, so it is not possible to control the rasterization of the letter character for only one size. Hinting Type1 fonts includes a lot of compromising on what the quality of the letter character is desirable and on what sizes this quality needs to be preserved.

Given their inflexibility and limited manipulation of pixels, Type1 hint mechanisms are not used for the fonts to be displayed on the screen, because due to the lower screen resolution, they require more precise control over the pixels.

TrueType fonts and their hint mechanisms allow for a much more precise control over the pixels.

TrueType Hint mechanisms

Rasterizing TrueType fonts

As noted earlier, the TrueType font contains characters that are described by the outline, namely, by the quadratic Bézier curve. Location of the dots that define the outline are described in Font Units (FUnit). It is the smallest unit of EM square. Em square is an imaginary square which can fit a capital letter M with its descenders and ascenders. The letter character is placed in this square, and the number of FUnits that take up the vertical side of the Em square is defined by the UPM (Units per Em) value. For the TrueType fonts, this value is 2048 UPM. The grid, Em square is composed of, is two-dimensional, and the coordinates x and y define the movement in horizontal and vertical directions, respectively. One unit of this coordinate system is equal to one FUnit. Each dot in this coordinate system has an integer value.

When scaling, outline stored in the font file is scaled to the requested size. The positions of the dots that make up the outline are no longer described in FUnits, but their coordinates are assigned value in pixels of the output device. After scaling, certain instructions (hints) defined in the font file are performed. This process is also referred to as *grid-fitting*, and as a result it has adaptation of the original outline to the given pixel grid of the output device, in order to better maintain the original shape of the outline. Grid-fitting is followed by Scan Conversion which is to determine which pixels will be visible on a given output device, and which not.
Before being displayed on the output device, the character, or rather the position of dots that form its outline, must be translated into the coordinate system of the output device. The positions of dots, expressed in units independent of the units of the output device, must be converted to absolute units of a particular output device, and when it comes to screen, the unit is pixel.

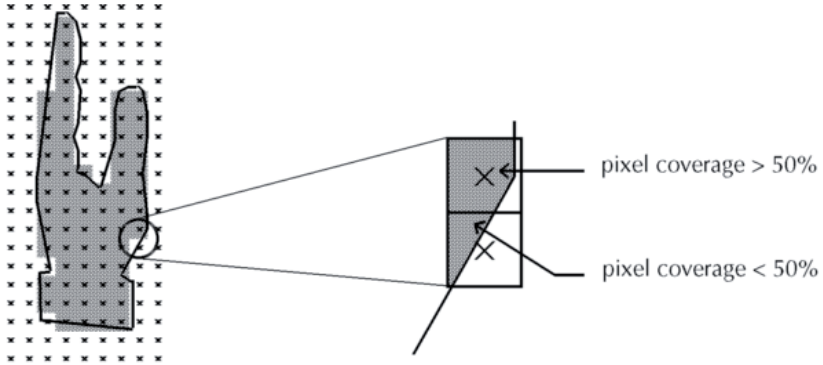

Figure 3. Scan-conversion of the outline. The coloured pixels are the ones that are covered by the outline in more than 50%, i.e. the ones whose center enters the space framed by the character letter outline

Convertion of the relative units of the Em square into absolute units of the output device (pixels) is performed by the following equation:

pointSize * resolution / (72 points per inch * units_ per_em) (1)

where pointSize represents the displayed size of the letter character on the output device, resolution means the resolution of the output device, 72 point per inch is the number of dots per inch.

For example, the height of a letter character is 500 FUnits on the 72 dpi screen, at the size of 12 pt. UPM value is 2048. The size of this letter character in pixels will be:

500*12*72/(72*2048)=2,93 (2)

The resolution of any output device, is expressed in the number of dots or pixels per inch (dpi). Resolution on the Windows platform is 96 dpi, the Macintosh platform 72 dpi, and in most laser printers 300 dpi.

The display of a letter character on a specific device is expressed in pixels per em (ppem). The formula for determining the resolution of the letter character on a particular output device is as follows:

Ppem=pointSize*dpi/72 (3)

Where the point size is character size in points, defined by a particular application of the output device, and dpi is the resolution of the output device. So for the size of 12pt on the Windows platform ppem value will be:

Ppem=12pt*96dpi/72dpi=16ppem (4)

At resolutions of 72 dpi, ppem value will always be the same as the value in points.

TrueType instructions (hints)

TrueType programming language, in addition to the description of the outline, also contains a collection of instructions intended to define the ways in which the outlines are to be adjusted to the pixel grid during rasterization on a specific output device. These instructions are called hints.
TreuType hints directly control the shifting of points and the transformation of their coordinates values from the relative coordinate system of the Em square to the absolute coordinate system of the output device. Hinting programs are written in a special programming language, which makes them very flexible, but also complex for direct programming. A set of higher programming languages is usually used for defining hints, and it is possible to get satisfactory results with minimal knowledge of the coding.

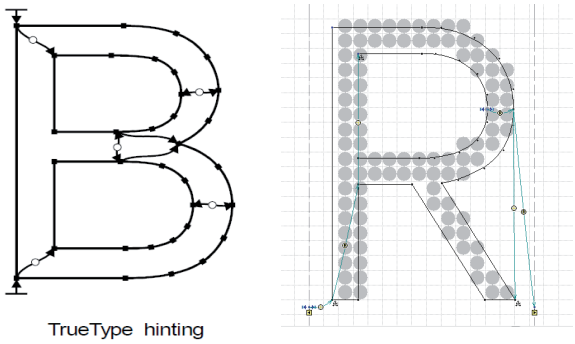

Figure 4. Schematic representation of the TrueType hints of the letters "B" (left) and the display of the TrueType hints in FontLab (right)

Hint mechanisms of the TrueType font file are defined by three types of programs (instruction).

The first is the so-called global program (Font program) - for all fonts and in all PPM sizes; the second is also a global program (PPM program) - for one glyph at all PPM values; and the third is local, or glyph (delta) program - which defines a glyph at a specific size of PPM.

Global program (Font program)

Font Program includes two operations over all the letters at all values of PPM.

The first operation is to determine the **aligment zones**. They define the area (zone) in which are all the characters which have overshoots, through some of the main vertical font metrics (cap-height, x height, baseline, ascender and descender). This is the case with rounded characters such as "o" which due to the visual compensation exceeds the x-height in order to appear roughly the same height as the letter "x". However, in smaller sizes and at lower resolutions, the visual compensation of only few pixels can reach 50% or 100% of the letter character height, so under these circumstances it is necessary to suppress the compensation and bring these letter characters to the height of the other letter characters.

The second procedure is to define the **standard stem widths**. This is important because while adaptation of the outline to the pixel grid (especially at small sizes and resolutions), there may be inconsistencies in the width of the strokes. Thus, two strokes of the same width can have different values at some sizes . This is especially conspicuous with the characters “n” and “m”, where it is expected that all vertical strokes, on small sizes also, remain the same.

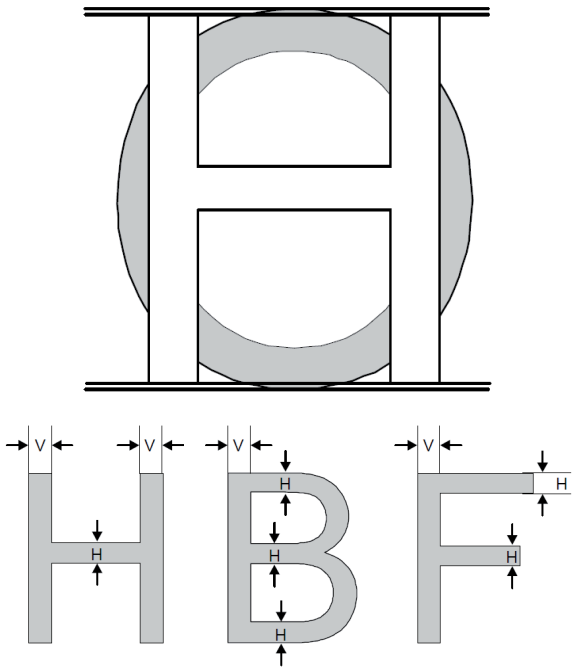

Figure 5. Alignment Zones (left) and standard stems widths (right)

Global program (PPM program)

This program contains several commands, and when applying these, it has an effect on the appearance of a letter character at all PPM values. These commands include:

Align - Aligns (moves) the position of the outline point to the designated position on the grid or to the edge of the alignment zone.

Single Link - Sets the position of the point relative to the position of another point. Distance can be linked with one of the stem widths. Distances also may be rounded or not.

Double Link - Sets the distance between two points to an integer value that may be linked with a stem width Interpolate Interpolates the position of a point between two other points

Interpolation – define the position, between two other points, and sets the relations among them.

Local, glif program (delta program)

These commands are specific for a particular glyph and at the specific PPM size. These commands are called *delta instructions* and are used for fine shifting of the positions of pixels, so the result is turning on/off only certain pixels. These instructions include Middle Delta and Delta Final commands.

In FontLab Studio or in Microsoft Visual TrueType (VTT) tool we use a small set of high-level hinting instructions that are automatically compiled to TrueType instructions during font export. Because these instructions can be set and edited visually we call them visual TrueType hints or just visual hints. Visual hints are enough to define TrueType hints even in very complex situations and they are compiled in very compact and effective TrueType instruction code *(FontLab, 2006)*.

This way, the typo designer is able to define the TrueType instructions and at the end produce a well designed font, without detailed knowledge of the TrueType programming language.

Approaches to Automatic Hinting

The process of writing TrueType instructions is a very time consuming process, especially when its about family of 10 fonts for example. That will be about 2500 characters. Each has its own hinting program.

Mitigating factor in the manual mode of entering the program code, are the tools for visual defining of the hint, called Visual TrueType. Using these tools, TrueType code is generated, which can afterwards be edited. This code is compiled into a TrueType font file.

Given the amount of work behind the manual hinting of fonts, automatic generation of the hints is a good starting point for further fine, manual adjustment of certain characters.

In the field of automatic hinting of fonts there have been some research with the aim to develop own tools for hinting or presentation of certain models and algorithms of solving the problem of font rasterization.

One of the presented approaches describes the automatic recognition and describing the characteristics of letter characters of a font, comparing it with the predefined model *(Hersch and Claude, 1991)*.

In this paper, the authors explain the approach where they define the base consisting of all the alphanumeric characters of Latin alphabet. These characters are hand drawn and constructed with the most optimal number of points necessary for making the outline. Hints are also defined in these models.

The point is that the points that define the character of the input font are compared to points of the outline of the same character of the model font. The points being compared must have an approximately the same position, or they are eliminated during the calculation. When all the points are compared, the result is description of the outline of the input character in the form of metadata.

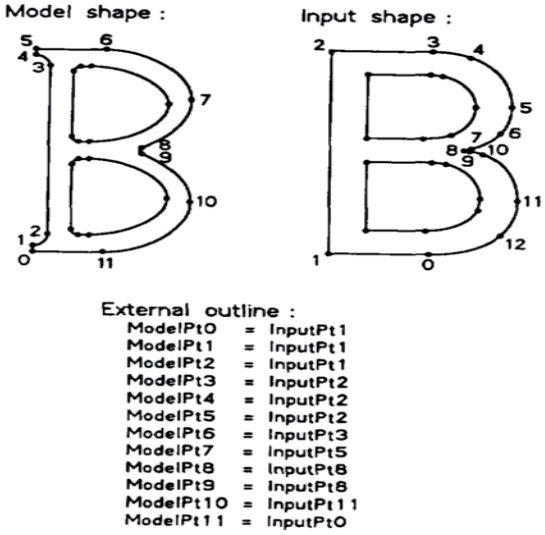

Figure 6. Comparison of the outline points on a predefined model and the outline points of the font being described

This description of the input font can later be used for hinting, when the hints of a predefined model are transferred to the input font, after they have been adapted to the compared and described outline. This approach is applicable only to the serif and sans-serif fonts, with simpler appearance and no needless ornaments. Comparing 70 different fonts to their model, the success rate of outline recognition was 99%. In the 1% the program reported that there were no characters that match the model. The error occurred with fonts which have slightly rounded vertical, horizontal and diagonal strokes (Optima, Palatino, Zapf Book).

Slightly more advanced approach is described in u *(Zongker et al., 2000)*. This approach is also based on comparison of models with the font that needs hinting. Only here, unlike in *(Hersch and Claude, 1991)*, instead of the hand-drawn model, professionally hinted TrueType font is used. First are compared the points on the outline of the model font and the target font in order to find similarities between the two outlines, i.e. to find matching pairs between them.

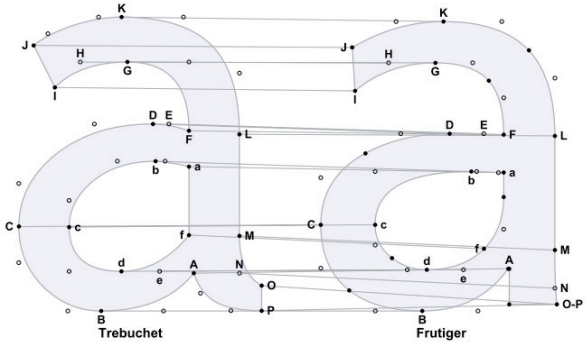

Figure 7. Comparison and matching the points of the font outline in a template and the font for which the description and automatic hinting needs to be set

Then the CVT table of the model font is used (which contains descriptions of characteristic points), it is copied and adjusted to the target font, and the changed names and the value of positions of points are entered.

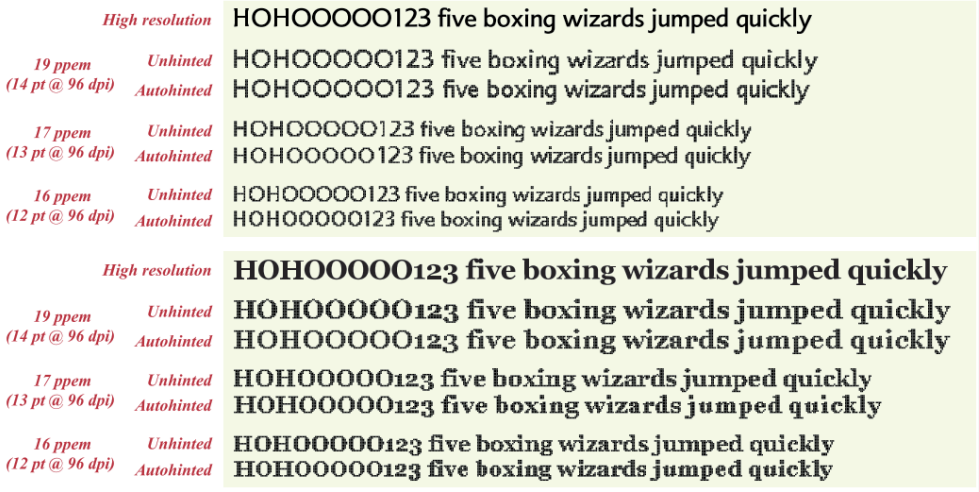

Figure 8. Scan-conversion of the outline. The coloured pixels are the ones that are covered by the outline in more than 50%, i.e. the ones whose center enters the space framed by the character letter outline

This being done, the appropriate hints can be copied from the model font to the target font, with the adjusted values of the points, of course.

Unlike the previous one, this approach is more automated and requires less time, because we do not need to make models, but use ready-made professionaly hinted fonts as templates. This approach was tested between the fonts of the same family (Sylfaen Sans Bold from Sylfaen Sans, Georgia from Georgia Bold, and Bold Italic Georgia Italic from Georgia), but also between different families of fonts (Bodoni, Calisto, Perpetua and Revival - from Georgia). Better performance was achieved when comparing fonts from the same family.

Based on the experience derived from previous studies of methods for automatic hinting and the perceived shortcomings of these methods, another approach for automatic font hinting is presented. This approach, in contrast to previous ones, is completely automated and it is based on identifying the parts of letter characters that need hints to be defined. This involves collecting global information on the font which is further associated with characters, defining a set of constraints, sorting according to relevance and conversion into a known hint programming language (PostScript, TrueType, VTT Talk, and others) *(Shamir, 2000)*.

Namely, each hinting technique can be divided into the following segments *(Shamir, 2000):*

Identification of hinting situations - recognizing the characteristic features that need to be preserved in every of letter character when setting the outlines on the raster grid (during gridfitting)

Handling of global sizes - determining the value of global sizes, such as character height, the height of vertical and horizontal strokes. This can be defined by the designer or by measuring and studying every character and collecting the information obtained. The global size serve to allow linking with the local hinting situation.
Prioritetizing - while defining hints, it can occur that an instruction can not be executed until another one has been defined, or that the existence of one instruction negates the existence of another, so it is necessary to set the logical sequence of instructions.

Translating - if some of the previous operations have been defined by a different programming language, they need to be translated into a low-level language of the specific technology (TrueType, PostScript, etc.).

Unlike the previous approaches *(Zongker et al., 2000)* and *(Hersch and Claude, 1991)*, which were semi-automatic, and applicable only to the Roman fonts (and if the topology, i.e. the shape, of the model letter character coincided with compared font), this model is fully automated and is applicable to all forms of letters (Asian, Hindu, ...).

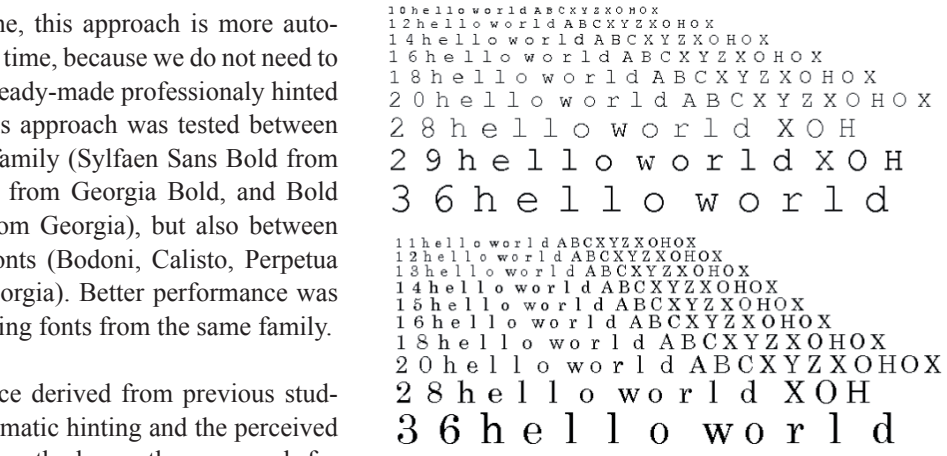

Figure 9. The result of automatic hinting using Constraint Based Approach for the fonts Courier (above) and Times-Roman (below). What is noticeable is the transition between 28 and 29pt in the example above and between 13 and 14pt in the example below. This is a much better solution than the one where these differences are apparent between the characters of the same size

In short, this approach involves the extraction of certain characteristic parts of the letter character (strokes, serifs, junctions, curves, white,etc.), their analysis and definition of the characteristic points and their limits, then, the comparison and collection of these data for each character, and eventually, linking this data with the global sizes, defining hints and their conversion into a known hint language.

Conclusion

The advantage of the automatic hinting is significantly reduced time of hinting. In the manual process, a professional typographer needs about 20-40 hours to define hints for the font of 256 characters *(Zongker et al.,2000)*. However, if we want high-quality results in the display on output device screen, in addition to automatic, it is necessary to also use the manual method of hinting. In order to achieve this, it is of course necessary to be familiar with the processes described above, but also with the opportunities that modern software tools offer. So as not to make the typo designers go further into programming in the low-level programming language (TrueType), there is the option to visually define the hint through some high-level languages, such as VisualTrueType (VTT), by relating and limiting the outline.
In a program such as FontLab Studio, these high-level commands are available and it is allowed to graphically

define hint using the appropriate tools. In this way, it is generated a code that is compiled in the font file and that can later be modified through the programming interface. TrueType language allows us to manually make correction of automatic hinting, using the option such as *dropout control*, which prevents the discontinuation of the outline when viewed at small sizes. In addition to this option, the TrueType also allows us to manipulate specific pixels of the letter character, but at specific size , which gives us greater control over the rasterization. This is possible through the use of *Delta hints*.

All these techniques (automated and manual) merged with the knowledge of anatomy of the letter and the basis of digital typography, are requirement for a well-designed font, which will be suitable for display on screens of different devices such as portable Tablet PCs, mobile phones, electronic book readers etc.

In this paper is shown some of the fundamental researhes in the field of automatic hinting. Understanding of these fundamentals is a starting point in advance research of how automatic hinting can be perfected so the end results will be close up, or equal to professional manual-hinted font.

Further research can go in the direction of getting acquainted with the ways in which various portable devices perform rasterization of fonts, which technology is used, what are their options and what hinting techniques work best on them.

References

1. Anon. TrueType Typography. TrueType Hinting. [Online] Available from: http://www.truetype-typography.com/tthints.htm [Accessed 29th June 2012].
2. FontLab Ltd. (2006) FontLab Studio 5 User Manual for Windows. [Online] Available from: http://www.fontlab.com/font-editor/fontlab-studio/download-fontlab-studio/ [Accessed 30th March 2012].
3. Hembert van J.. Practical TrueType hinting. [Online] Available from: http://luc.devroye.org/tt_hinting_tutorial.pdf [Accessed 28th June 2012].
4. Hersch R.D., Claude B. (1991) (1991) Model-based Matching and Hinting of Fonts. ACM SIGGRAPH Computer Graphics. [Online] 25 (4), 71-80 Available from: http://dl.acm.org/citation.cfm?id=122726&CFID=93842326&CFTOKEN=66875597 [Accessed 1st April 2012].
5. Hersch R.D. (1993) Font Rasterization: the State of the Art. In: Roger D. Hersch (eds.) Visual and Technical Aspects of Type. England, Cambridge University Press, 78-109
6. Microsoft Corporation. (1997) TrueType fundamentals. [Online] Available from: http://www.microsoft.com/typography/otspec/TTCH01.htm [Accessed 28th March 2012].
7. Shamir, A. (2003) Constraint Based Approach for Automatic Hinting of Digital Typefaces. ACM Transactions on Graphics (TOG). [Online] 22 (2), 131-151. Available from: http://dl.acm.org/citation.cfm?id=636887 [Accessed 4th April 2012].
8. Zonker D.E., Wade G., Salesin D.H. (2000) Example-Based Hinting of TrueType Fonts. SIGGRAPH '00 Proceedings of the 27th annual conference on Computer graphics and interactive techniques. [Online], 411-416 Available from: http://dl.acm.org/citation.cfm?id=344779.344969&coll=DL&dl=GUIDE&CFID=93842326&CFTOKEN=66875597&preflayout=tabs [Accessed 22nd of April 2012].