Font Rendering Methods

Neal Burger

Abstract— This paper consists of a brief overview of the history of fonts an brief introduction to computer fonts and different displaying techniques.

Index Terms—bitmap fonts, outline fonts, stroke fonts, font hinting, anti-aliasing, subpixel rendering, Microsoft Cleartype

1 Introduction

With the dawn of the information age, more and more text is being produced and consumed with electronic devices. E-mail, blogs, twitters, short messages - all these different forms of electronic text are now becoming part of our daily lifes. Many developments have been made to make this experience as pleasant as possible. Most of these discoveries are very subtle but have an enormous subconscious effect.

2 History of Fonts

Throughout history, rendering of letters was always defined by the then current technology. For example, in ancient Rome stone plates and chisel were used to write. This led to the very well known serif font style (Times New Roman or similar fonts). Serifs were introduced to increase readability. The serifs only suggest a line, but this helps the eye significantly to read words. Centuries later, with the influence of parchment and quill from Egypt, stone tablets became obsolete and calligraphy gained importance. This switch lead eventually in the 11th century to the first gothic scripts known as "blackletter".

The "blackletter" also became the first font used by Johannes Gutenberg's printing press. Glyphs were cut into the printing blocks, assembled into text and then printed by the press. As time progressed the printing press manufacturing got enhanced and depending on the material used for the glyphs designers started to develop different font families. For example, the font "Bodoni" was originally sculpted from steel blocks. Its form was limited how steel could be cut at the time. Bodoni is one of the first abstract fonts, having extremly thick and extremely thin lines. The most significant change was, that now without manual writing, all glyphs were printed the same way, and a machine defined the appearance of the text. The concept of movable type was known in China before Gutenberg invented the printing press. But the Chinese could not apply it in the same way as the Western culture, because their texts contain vast amount of different letters.

Figure 1. Wim Crouwel's "new alphabet" created for optimal output on CRT-Monitors - but only a theoretical attempt for displaying text.

Centuries later in the beginning of video displays (CRT-Monitors) had only a very coarse display grid. Only dots could be displayed or printed. Traditional fonts could not be displayed with this new technology. The designer Wim Crouwel published 1967 the first typeface consisting only of lines *(see figure 1)*. This was only a theoretical attempt to avoid many problems of digitized fonts at the time, it was never meant to be used. In the 90's, however, its variations got used by the pop-culture[5]. As technology advanced higher resolutions could be printed. Traditional fonts were ported to computers with scanners or manual input - even though in the beginning they could not be displayed properly on screen [7].

- *Neal Burger is studying Media Informatics at the University of Munich, Germany, E-mail: N.Buerger@campus.lmu.de*
- *This research paper was written for the Media Informatics Proseminar on "Algorithms in Media Informatics", 2009*

3 Computer Font Types

3.1 Bitmap Fonts

Bitmap fonts, also known as raster font, was the first form of a computer font. Every glyph is stored as array of pixels, so in essence it stores a separate image for each glyph. The first bitmap fonts were manually entered into the computer. Later, font designers used photoscopic scanners (the first black and white scanners, based on spatial sampling of images) to input single glyphs. The font can be arranged as mono spaced font (every glyph has the same width) as well as proportional font (every glyph has its own width), there are no limitations concerning kerning.

However, one cannot scale bitmap images without side effects. All data stored in a bitmap image is pixel absolute, the data cannot be manipulated to produce proper indefinite scaling. Each individual font size needs a separate set of images. (The same applies for bold and italic variations of the letters). Unlike outline fonts, unscaled bitmap fonts produce always exactly identical output and do not have to be interpreted before output. These fonts are today mainly used on consoles of operating systems and environments with limited processing power, like embedded systems or older computers.

To render bitmap fonts the binary data is simply read out of the file, and is displayed according to the pixel information*(see figure 2)*. This is a fast and easy approach, no extra hardware or acceleration is needed for display. For example, the typeface Tahoma was specifically designed as bitmap font (but implemented as TrueType Font) and then was used as the system default font for Microsoft Windows 2000, XP [11].

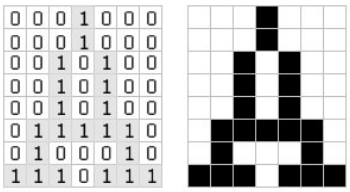

Figure 2. Binary data to pixel display

3.2 Outline Fonts

Outline fonts are defined by Bezier curves. These curves can be scaled indefinitely, only one set of characters is needed for all font sizes. More advanced outline font formats, like Adobe PostScript-Type-1 or Apple TrueType include outline curve grid fitting information to ensure readability at small font sizes.

For each font width, still a single set of characters is needed. There are possibilities of Faux Bold (the outline of the font gets an extra

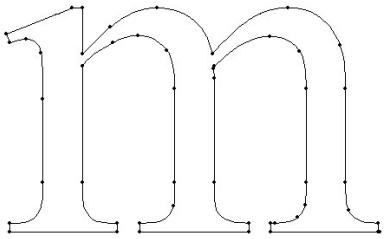

Figure 3. Character "m" in Times New Roman as Outline. The dots represent the Bezier control points

stroke) and Faux Italic (the font gets slanted) but in most cases the computer calculated glyphs are not appealing and a separate set of glyphs is needed.

Three steps are involved rendering outline fonts: outline-grid fitting, outline-scan conversion and outline-filling. This leads to the side effect that fonts do not render equally on every type of device. Dependent on the display type, pixel density and render engine subtle differences can be seen [4]. The most noticable difference is between monitors and print output. Laser printers in most cases have an much higher resolution than monitors. The fonts need to be re-rendered for print, in most cases a different rendering engine is used than for monitors to optimize the print output.

3.2.1 Font Hinting

Even though one could indefinitely downscale outline fonts, due to physical limitations of display devices one only can fill the outlines with full pixels. When scaling to a size so small every detail of the font cannot be displayed as separate pixels. The outline produces irregular shapes, because details start disappearing and the type gets unreadable. In these cases font hinting is used. Font hinting is in essence additional

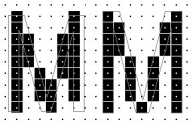

Figure 4. 9px high Arial "M" Comparison without and with font hinting

mathematical information to enable the font to be still readable without some of its details. This information is only used to deform the original outline when the font is displayed at a low resolution. This process of snapping the information to the pixel grid is called grid fitting [10]. The font hinting information must be added when the font is created. This hinting information are additional characteristics of a letter, like the distance between parts and phase information. (The phase determines the discrete width/height of the bar.)

3.2.2 Outline-Scan Conversion and Fill

In essence the Outline Font is a collection of Bezier curve control points. The outline needs to be calculated and scaled to the proper size. The outline-scan conversion lays the outline over the pixel grid. All pixels that make up the outline of the font get flagged and stored in the image buffer *(see figure 6)*. The fill of a character is defined as inner and outer spans. The inner spans get filled with color and the outer spans stay white/transparent. Once the complete outline has been flagged, the flag-fill algorithm scans horizontally the image and colors each pixel. For each read flag the color gets toggled between transparent/white and the fill color. Since the same rules apply to every character complete words and lines can be rendered at the same time[4].

3.3 Stroke Fonts

For Japanese, Korean and Chinese, the combined character sets have around 37,000 characters. Even if they would be stored as outline fonts

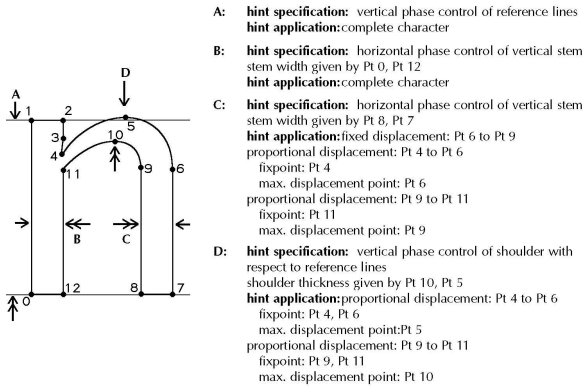

Figure 5. Font hinting information for grid fitting

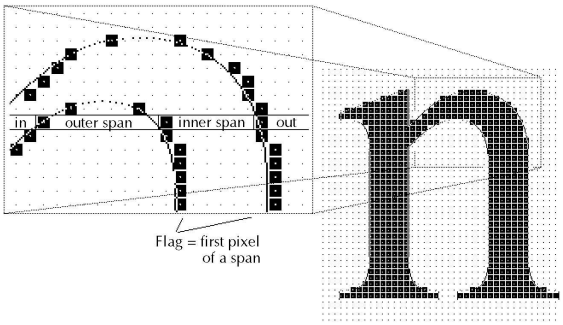

Figure 6. Inner and outer spans

the file size would be from 6-12MB. To preserve memory and performance on low-performance devices, stroke fonts were introduced. The same character set now only needs 250-650KB. Stroke fonts are defined like outline fonts by Bezier curves, but only the center line gets defined. This way much fewer points are needed to define a shape, less data needs to be stored. Besides the stroke, the width of the font needs to be declared. By changing this width, different font weights (bold, light) can be created. The drawback of stroke fonts are that it takes longer to design fonts and only simplistic designs make sense to implement. In addition, complex extensions to letters, like serifs, cannot be supported.

Another feature of stroke fonts is that it can use a library of "radicals". Radicals are common features in many characters *(see figure 8)*. This further reduces the space requirements to store the font. The radicals and strokes get combined into the final character which is then rendered.

To render stroke fonts first a stroke gets calculated by combining the radicals and the strokes to a single stroke. Then outlines are drawn along each center line of every stroke in correspondence to the defined

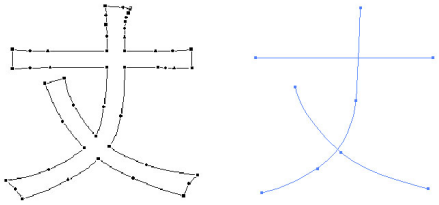

Figure 7. Same symbol defined by outline and defined by stroke/center line

weight[9]. The remaing process is equivalent to the outline rendering process.

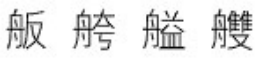

Figure 8. Same radical in different symbols

4 Font Smoothing

Font Smoothing are technologies that enhance the visual appeal of fonts on a display. The techniques presented here are optimized to work with outline fonts.

4.1 Anti-Aliasing

With anti-aliasing, the character is created with squares of color that are shaded darker or lighter depending on how much of the curve would overlap that pixel. The most common approch to this is overscaling the font outline and then downsampling it again. The underlying downsampled outline gets filled with the foreground color. The overlapping parts of the overscaled image are then shaded in a lighter tone of the foreground color. This can be further enhanced with extended font hinting[12].

The human eye perceives this shape as a more rounded shape making the text more natural and easier to read. In popular consumer devices like the Amazon Kindle, which has a 16 grayscale E-Ink display, uses anti-aliasing to display its text.

Figure 9. Comparison of anti-aliased text vs. aliased text.

4.2 Sub-Pixel Smoothing

Sub-Pixel smoothing, as the name indicates, needs a display device capable of addressing Sub-Pixels. Such a device would be, for example, a Liquid Crystal Display (LCD). In Cathode-Ray-Tube Monitors (CRT) usually the beamspot overlaps the other beamspots. So different individual RGB-Sub-Pixels cannot be addressed properly. However, in an LCD-Monitor all pixels consist of 3 seperate subpixels R-G-B; these are arranged horizontally in a matrix [2].

Fonts have now been redesigned to use the three horizontal subpixels, enabling to rasterize the font at a three times higher resolution. Sub-Pixel smoothing "borrows" from the neighboring pixel one or two Sub-Pixels. This increases the contrast and smoothness of the letter with its surroundings. It is important to note that no color fringing occurs. When all three subpixels are turned on, the human eye combines the colors and the complete pixel appears white. The human eye combines the adjunct subpixel into the pixel before and after so no difference is noticable. It does not matter in which sequence RGB gets combined, RGB, GBR or BRG, at full intensity the three subpixels always get combined into white[8].

While implementing Sub-Pixel rendering,it is important to include information about the monitor actively being used and the properties of its matrix. Not all monitors have RGB arranged pixels, some may have different arrangements.

4.3 Visual Search and Reading Tasks Using ClearType and Regular Displays: Two Experiments

In 1998, Microsoft Corp. unveiled Microsoft ClearType - A Sub-Pixel Font Rendering Technology optimized for matrix LCD Panels.

In April 2006, a paper[1] was published that tries to proof Jakob Nielson's theory, "ClearType could save $2000 per year because reading efficiency increased by 10-15%." The experiment was divided into two parts: scanning and reading text. This experiment was using common work tasks. The findings as published in the paper support Nielson's theory, an average reading efficiency increase of 7,2% by using ClearType. On the other hand they discovered that in 19 out of 51 users experienced difficulties with ClearType[1].

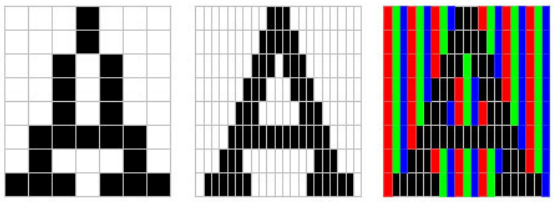

Figure 10. Left to Right: 1.Bitmap render, 2.Sub-Pixel render 3.Sub-Pixel render hardware monitor output - you can clearly see the much smoother edge in the subpixel render towards the bitmap render.

Unfortuneatly, the paper neglects to mention that the typeface Arial used in the experiments was optimized for IBM's laserxerographic printer and not for ClearType.[3] It is not clear if the result would have been different if they would have used a ClearType optimized font like Calibri. Nevertheless in November 2006, with the release of Windows Vista, Microsoft enables ClearType per default in their operating system and includes several ClearType optimized font families. No further studies have yet been made with these enhancements.

5 Font Formats

5.1 PostScript-Type-1-Font Format (Type1)

Developed in 1984 Adobe introduced one of the first Outline Font Formats. PostScript is a device independent page description language. The accompaning Font Format Type-1 uses third degree Bezier curves to describe outlines. A proprierary hinting technology was used for the display output. Type-1 ensures complete compability with all PostScript language devices. The PostScript 3 has been extended to support OpenType and TrueType Fonts, so all PostScript devices support all font standards.

5.2 TrueType Format (TTF)

Apple developed the TrueType Format to avoid paying money for licencing proprietary font technologies. To describe the outlines, true type uses second degree Bezier Curves and a newer type of font hinting got added to this format. Microsoft later licenced this technology but the TrueType Fonts are not interchangeable between operating systems. Two seperate versions of the same font are needed so it works on both operating systems.

5.3 OpenType Format (OTF)

In a joint effort by Microsoft and Adobe the OpenType Format got created. The "OpenType Format" is open to everyone and has as little limitations as possible. This format supports the PostScript Type 1 outlines as well as the TrueType outlines. To further enhance font sets one now has the possibility of true small caps, different styles of figures and alternates (for every single glyph). Advanced typografic features like ligatures can also be stored in the format. The maximum number of characters in an OpenType Font is limited to 64.000 glyphs. OpenType has even cross-platform support. It has no support for stroke fonts, because stroke fonts use proprietary software[6].

6 Common Fonts - Helvetica, Arial, Calibri

Helvetica, the "white" of typography, is the most widely used font. It was designed 1957 by Swiss typeface designer Max Miedinger. It has a very harmonic lettering and gererally is conceived as an extremely pleasant font[5].

Arial, developed for the IBM laserxerographic printer, is a derivative of Helvetica. It is actually built up in pixel steps, and was (in the beginning) not defined by curves - due to the hardware design of the

printer. Several years later Arial was shipped with the early Version 3.0 of Microsoft Windows. Since then it has undergone several iterations and now with the newest Version 5.0 of Arial it is optimized for several different use cases, like ClearType or small screens and is defined by outline curves.

Calibri, developed by Lucas de Groot for Microsoft - belongs like Helvetica to the human-sans-serif families. But is in fact not directly based on Helvetica. It was entirely developed for the ClearType-rasterization from Microsoft. Microsoft decided to use Calibri as their default typeface for their operating system as well as for their Office Suite - this resulted in that Calibri is currently one of the most widespread used fonts.

7 Conclusion

Fonts have been a human development for centuries. It is interesting to see how technology infuences design and design influences technology. Good design and good technolgy improves information aquisition.

References

[1] A. Dillon, L. Kleinman, G. O. Choi, and R. Bias. Visual search and reading tasks using cleartype and regular displays: two experiments. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 503–511, New York, NY, USA, 2006. ACM.

[2] S. Gibson. How sub-pixel font rendering works. `http://www.grc.com/ct/ctwhat.htm`, 2005. visited 08.07.2009.

[3] A. Haley. Is arial dead yet? `http://www.useit.com/alertbox/20030825.html`, 2007. visited 2.5.2009.

[4] R. D. Hersch. Font rasterization: the state of the art. In *in Visual and Technical Aspects of Type*, pages 78–109. University Press, 1993.

[5] G. Hustwit. Helvetica the movie. `http://www.helveticafilm.com`, 2007. visited 08.07.2009.

[6] A. S. Incorporated. Font formats. `http://www.adobe.com/type/topics/info9.html`, 2009. visited 08.07.2009.

[7] E. Lupton. thinking with type. In *thinking with type*, pages 13–37, New York, NY, USA, 2004. Princeton Architecutal Press.

[8] D. S. Messing and S. Daly. Improved display resolution of subsampled colour images using subpixel addressing. In *ICIP (1)*, pages 625–628, 2002.

[9] B. Thomas. Stroke-based fonts. pages 2–7. Bitstream Inc., 2004.

[10] M. Typography. Truetype hinting. `http://www.microsoft.com/typography/TrueTypeHintingWhat.mspx`, 1997. visited 08.07.2009.

[11] M. Typography. Windows xp/2000 default system font. `http://www.microsoft.com/typography/fonts/family.aspx?FID=19` `http://support.microsoft.com/kb/823033`, 2008. visited 08.07.2009.

[12] M. Typography. Engineering changes to cleartype in windows 7. `http://blogs.msdn.com/e7/archive/2009/06/23/engineering-changes-to-cleartype-in-windows-7.aspx`, 2009. visited 08.07.2009.