Measuring and Enhancing the Legibility of GPU-rendered Text

Christoph von Tycowicz and Jörn Loviscach

Fakultät Elektrotechnik und Informatik, Hochschule Bremen (University of Applied Sciences), Bremen, Germany

Abstract

Whereas the rendering of tiny typefaces in 2D applications has been perfected over decades, the legibility of text in 3D visualizations has rarely been addressed. This affects road signs, meters, screens, and books in virtual reality applications ranging from car driving simulators to digital lecture halls. To improve text rendering and display, we devised a lightweight psychovisual test to measure one prominent aspect of legibility. We subjected promising GPU-based methods for crisp minification to this test. It turned out that legibility may be improved without undue costs. Not all techniques whose results look appealing on first sight will, however, enhance legibility.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Display Algorithms

1. Introduction

Much research has addressed GPU-based methods that focus on a clean display of *large* fonts. This paper looks at *small* fonts, however, since they are vital for the legibility of, for instance, road signs in car driving simulators or presentation screens in virtual lecture halls. To the best of our knowledge, the merits of GPU-based font rendering techniques in terms of legibility have not been researched into before. We define a psychovisual test that quantifies legibility quickly enough to compare a substantial number of techniques. Then, we discuss a number of known as well as novel minimally invasive GPU-based techniques that may improve legibility, see Figure 1. This discussion includes a number of practical insights concerning efficiency. Finally, we quantify legibility for a selection of technically promising methods.

Figure 1: *Standard texturing leads to excessive blur (top, strongly magnified). Negative MIP level bias (mid) and RGB subpixel rendering can help (bottom; this grayscale print does not reveal the detrimental color fringes).*

After reviewing related work in Section 2, we describe our test method in Section 3. In Section 4 we discuss rendering techniques; Section 5 presents the outcome of the tests. Section 6 concludes the paper, pointing out directions for future research. To observe the strict limitation in page length, screenshots, speed benchmarks as well as the detailed test results are provided as electronic attachment to this paper.

2. Related work

Loop and Blinn [LB05], Kokjima et al. [KSST06], and Qin et al. [QMK06] create curved shapes on the graphics processor. Texture images can be employed to store data to be rendered with hard edges. This idea lies at the heart of methods proposed by Tumblin [TC04], Sen [Sen04], Ramanarayanan et al. [RBW04], Turini and Cignoni [TC05], and Loviscach [Lov06]. However, all of these GPU-based methods for vector graphics and/or hard-edged textures mostly address magnification: rendering large fonts without jaggies. If they address minification, which is vital for legibility, it is either through standard MIP-mapping or through measures that only take the nearest contour into account. A different option [HL07] is to borrow the high quality of standard 2D font renderers, with costs in speed and geometric precision.

To quantify legibility is a complex task. A debated issue is in how far the overall shape of a word (the so-called bouma) is important. As Larson [Lar03] argues, experiments overwhelmingly support that the legibility of the individual characters is paramount, in contrast to the opinion of many typeface designers. Gugerty et al. [GTAE04] observe that standard antialiasing may be less legible then pixelated fonts, with Microsoft's ClearType method [BBD*00] offering the optimum in terms of both smoothness and legibility.

3. Measuring legibility

Mathematically exact antialiasing may not be the most legible way to render typefaces [GTAE04]. This hints at a limited value of simple measurements such as signal-to-noise ratio. The applicability of more complex models of human vision for legibility studies has not yet been examined. Thus, one has to resort to tests with human participants. A standard test concerns reading speed, see for instance [BMPS01]. However, reading long texts in a 3D visualization is not a common task, and possibly for good reasons. A search task [MHC98] fits better to the current usage of text in VR environments. We elected to display text on billboards that seem to approach the user in 3D. This mimics road signs in a virtual reality application and allows sampling font size as a variable, thus reducing the number of test runs.

Based on legibility research [Lar03], we focus on the discernibility of single characters, which is strongly influenced by the rendering technique. Of course, there are other factors: Spotting the number of words and the ascenders and descenders (that is: knowing the bouma) may be enough to guess the road name on a distant sign. To mostly eliminate the effect of the bouma, we introduced six types of spelling errors that keep the contour of the word intact: a $\rightarrow$ n, e $\rightarrow$ o, e $\rightarrow$ s, h $\rightarrow$ k, i $\rightarrow$ l, m $\rightarrow$ rn. We prepared a list of names of well-known cities consisting of five to ten letters and introduced one of the six spelling errors into each, such as turning "Paris" into "Parls." These names were assigned to the rendering techniques so that every technique was tested for every participant with every kind of spelling error. For every participant, a individual random sequence of the test items was chosen to prevent a training effect.

The participants are told that every sign contains a typo, which they have to spot. For every item they see a billboard that faces and slowly approaches them. On finding the spelling error, they have to press the space bar on the computer's keyboard. Our evaluation software immediately blanks the screen, saves the current size of the typeface, and asks for the misspelled letter. If the user picks a wrong letter, this counts as invalid. The threshold type size thus determined is a quantitative measure for the difficulty of the task. This difficulty varies for different types of misspellings and varies from user to user due to different acuities of vision or due to different levels of hesitation in determining the wrong character. On top of these influences, the reaction time of each user enters the measurement, even though the billboard is magnified only slowly. All of these effects can be countered by only looking at the *ratios* of threshold type sizes determined for the same user and the same type of typo but with different rendering techniques.

4. Techniques and speed benchmarks

We tried to cover many plausible routes to crisp font minification. Speed benchmarks and a first visual inspection were applied to select a subset of viable methods to be subjected to the ensuing legibility test. All techniques rely on vertex and pixel shaders that operate on standard textures or on textures provided by a corresponding preprocessing method. Focusing on font minification and aiming at simple integration with existing VR applications or games, we could not make use of the techniques mentioned in Section 2, with the exception of [HL07], which we cover by using standard 2D font rendering in comparison. To allow for moving type, one also cannot snap the font's outlines to the pixel grid through the "hint" instructions of a font file or other techniques [Art05], as this would lead to jerky motion.

Individual generation of MIP levels. One may create each MIP map level from scratch as opposed to averaging a finer level. However, using Microsoft Windows' standard 2D font antialiasing, we found no visible difference.

Gamma-correct MIP-mapping. MIP interpolation in the linear domain [Gd07] causes brighter, seemingly *less* clear characters. On closer inspection, however, the stems are less inflated; typefaces with highly varying stroke widths do not decompose into too dark thick strokes and too bright thin strokes. The most efficient way to correct gamma was to use non-gamma textures and switch on hardware-based gamma correction for the pixel shader's linear output.

Negative MIP bias. A negative bias will force a sharper MIP level to be used. A bias of -1 incurs a performance penalty of less than one percent and is virtually not invasive, since one can control the bias with the texture sampler's settings.

Unsharp masking. Sharpening may be applied in texel space or in screen space. For the former, we applied standard unsharp masking to all MIP levels [Bjo06], noting that the blackening of the strokes lost its uniformity due to increased aliasing. For sharpening in screen space, we formed combinations of texture samples taken with different MIP bias settings, as described in the OpenGL documentation [Nvi01]. The visual results were hardly distinguishable from a standard texture with negative MIP bias. The latter requires only one texture request per pixel and thus is to be favored.

Contrast enhancement. Averaging lets the coarse MIP levels lose contrast, which may reduce legibility. This can be countered by scaling the contrast by a factor of two around the map's average gray. This can be elegantly combined with the method [Lov06], where a similar computation serves tex-

ture *magni*fication. That the enlarged contrast is not photorealistic was not found detrimental by the test users.

MIP powers of $\sqrt{2}$. Standard MIP levels are sized at powers of two. For a finer stepping, we created two regular MIP chains—one with sizes 1, 2, 4, ..., the other one with $\sqrt{2}$, $2\sqrt{2}$, ...—using the texel-to-pixel size ratio for blending. To have a common grid for both chains, we also tried rotating one chain by 45°. No result, however, was noticeably sharper or less aliased than regular MIP-mapping. An additional MIP bias of -1 caused uneven blackening.

Supersampling. Full-screen antialiasing (FSAA) expensively affects all objects. To adapt the computational effort to the local need, we looked at supersampling on a regular grid and on random points in the pixel shader. We computed the footprints through the pixel shader's `ddx` and `ddy` functions. Allegedly, these are slow, but the—highly invasive—precomputation of the footprint in the vertex shader was only a few percent faster, even in the best case. For supersampling with a regular grid, using a `for` loop with a parameter-controlled limit turned out to be not viable, since in this case the sampling footprint has to be specified explicitly through the slow `tex2Dgrad`. Hence, we fixed the number of samples. We also tried to abandon trilinear MIP-mapping altogether because its effect is similar to that of supersampling. This, however, was too slow since it required too many texture samples to be visually competitive. We looked into two options to incorporate randomness: First, we stopped the sampling when the color variance of the samples collected so far fell below a given threshold; second, we applied spatio-temporal randomness by changing the sampling pattern not only from pixel to pixel but also from frame to frame, so that the samples can be averaged over time in the viewer's eye. The first method turned out to require up to 20 samples to limit visible noise at the typeface character's edges. The second method did not behave much better, presumably due to electronic measures in the display unit to boost temporal color changes to lower the response time.

RGB-subpixel rendering. As known from Microsoft ClearType, Adobe CoolType and Apple Quartz, the RGB subpixels of standard displays may be used to boost the horizontal resolution [BBD*00]. On a GPU, one can realize this through three MIP-map texture requests per pixel that are shifted in *uv* texture space by plus or minus one third of the pixel's horizontal footprint, determined through `ddx`. Due to the blurriness of standard texture mapping, the effect of subpixel rendering was not noticeable, however. Thus, we applied a negative bias of -1. Stronger settings lead to offensive color fringes around the letters.

Adaptive sampling. To fetch multiple texture samples per pixel impacts rendering speed. Since texts contain large uniformly colored areas, adaptive sampling may lower the computational cost. To control the sampling, we tried two options: first, we augmented the grayscale texture by a second eight-bit channel in which all texels close to contours were marked; second, we checked whether the grayscale value retrieved from a standard MIP map is darker than 0.98. When applied to RGB subpixel rendering as discussed before, both methods *reduced* the frame rate on our test system. Obviously, the price of the `if` instruction in a pixel shader is still too high, even when the branching occurs spatially coherent.

5. Legibility results

Our legibility test focused on techniques found to be viable according to Section 4. For each of the ten participants (eight male, two female, age 22 to 28, all with normal or corrected eye sight) we tested a randomized series of all combinations of six types of misspellings with five rendering techniques: standard (not gamma-correct) MIP-mapping; three gamma-correct methods with a MIP bias of -1, namely, MIP-mapping, contrast enhancement by a factor of two, and subpixel rendering; and standard Windows 2D GDI antialiasing as a reference. The typeface used was a black-on-white TrueType font implementing "DIN-Schrift," the German standard DIN 1451, which is used for signs along highways, so that the participants were accustomed to reading this typeface in similar situations. They faced the display, an Acer AL1916 with a dot pitch of 0.294 mm, perpendicularly at an eye-to-screen distance of 50 cm.

We discarded the 14 of all 300 test runs in which the users failed to specify the typo correctly and in time. To remove the differences between the participants and between the types of spelling errors, we used the GDI test data as a reference: For every participant and every type of spelling error, we divided the recorded height by the height recorded for the GDI presentation for the same participant and the same type of spelling error. We took the logarithm of the ratios to arrive at data that satisfied standard tests for normal distribution. Then we conducted an analysis of variance (ANOVA) on these data, which revealed a high statistical significance ($p = 0.0003$). According to the criteria of both Scheffe and Bonferroni, there is a significant difference between subpixel rendering on the one hand and standard MIP-mapping as well as contrast enhancement on the other hand.

The averaged results—see Figure 2—show that only subpixel rendering is on par with the GDI. Biasing the MIP level comes close (4 percent larger font size). Contrast enhancement (16 percent larger) performs even worse than standard gamma-incorrect MIP mapping (13 percent larger). The scatter of the data is most pronounced for the contrast enhancement technique. A closer look at the data reveals that blackening is a two-edged sword: In some cases, it improves the readability appreciably; in other cases it ruins it, which may be attributed to the strong aliasing this method causes.

6. Conclusion and outlook

We presented a measurement method and a large collection of ideas on how to improve font rendering in 3D scenes for

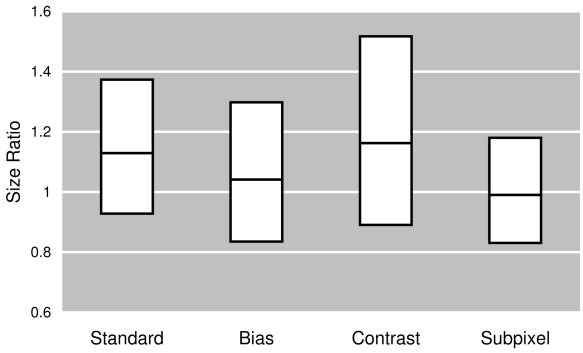

Figure 2: *Mapping the averages (indicated by the middle lines) and the standard deviations (indicated by the boxes) back to the non-logarithmic domain allows to read off the type size required for error detection in relation to the size required with standard GDI font rendering.*

better legibility. As the results show, it is possible to enhance legibility at little cost, even though we found that it is difficult to go beyond the results of biasing the MIP level. This technique is not only highly efficient but also virtually non-invasive. RGB subpixel rendering provided a further improvement in terms of quality at a slightly higher price. It provides reliably better results that standard MIP-mapping as it is better by almost one standard deviation.

To test the impact of pixelization as such, we run a preliminary test series with the display unit placed 150 cm from the user. This shrinks the apparent size of the pixels without any change in brightness or the testing procedure. Our results indicate that a high resolution may reduce the threshold typeface size by as much as 30 percent. This needs to be validated with actual high-resolution displays.

Knowing the correlation of the search speed data, as given in this paper, to the reading speed would be very helpful, even though the latter is difficult and/or laborious to assess, as discussed in Section 3. Future work may also address the legibility of laterally moving type and type under oblique view. In addition, one may research into the design of typefaces specifically suited for 3D worlds.

References

[Art05] ARTIFEX INC.: FontFocus whitepaper. http://www.artifex.com/FontFocus.pdf, 2005.

[BBD*00] BETRISEY C., BLINN J. F., DRESEVIC B., HILL B., HITCHCOCK G., KEELY B., MITCHELL D. P., PLATT J. C., WHITTED T.: Displaced filtering for patterned displays. In *SID Digest*. 2000, pp. 296–299.

[Bjo06] BJORKE K.: When shaders and textures collide. GDC 2006 presentation, 2006.

[BMPS01] BERNARD M., MILLS M., PETERSON M., STORRER K.: A comparison of popular online fonts: Which is best and when? *SURL Usability News 3*, 2 (2001).

[Gd07] GRITZ L., D'EON E.: The importance of being linear. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley, 2007, pp. 529–542.

[GTAE04] GUGERTY L., TYRRELL R. A., ATEN T. R., EDMONDS K. A.: The effects of subpixel addressing on users' performance and preferences during reading-related tasks. *ACM Transactions on Applied Perception 1*, 2 (2004), 81–101.

[HL07] HEISE S., LOVISCACH J.: Real-time lettering on 3D signs with a 2D font engine. In *Eurographics 2007 Short Papers* (2007), pp. 89–92.

[KSST06] KOKOJIMA Y., SUGITA K., SAITO T., TAKEMOTO T.: Resolution independent rendering of deformable vector objects using graphics hardware. SIGGRAPH 2006 Sketches, 2006.

[Lar03] LARSON K.: The science of word recognition or how I learned to stop worrying and love the bouma. http://www.microsoft.com/typography/ctfonts/WordRecognition.aspx, 2003.

[LB05] LOOP C., BLINN J.: Resolution independent curve rendering using programmable graphics hardware. *ACM TOG 4*, 3 (2005), 1000–1009.

[Lov06] LOVISCACH J.: Rendering road signs sharply. In *Game Programming Gems 6*, Dickheiser M., (Ed.). Charles River Media, Boston, 2006, pp. 501–516.

[MHC98] MORRIS R. A., HERSCH R. D., COIMBRA A.: Legibility of condensed perceptually-tuned grayscale fonts. In *Electronic Publishing, Artistic Imaging and Digital Typography*, Hersch R. D., André J., Brown H., (Eds.). Springer-Verlag LNCS 1375, 1998, pp. 281–293.

[Nvi01] NVIDIA: Ext_texture_lod_bias. OpenGL Extension, 2001.

[QMK06] QIN Z., MCCOOL M. D., KAPLAN C. S.: Real-time texture-mapped vector glyphs. In *Proc. of I3D 2006* (2006), pp. 125–132.

[RBW04] RAMANARAYANAN G., BALA K., WALTER B.: Feature-based textures. In *Eurographics Symposium on Rendering 2004* (2004), pp. 265–274.

[Sen04] SEN P.: Silhouette maps for improved texture magnification. In *SIGGRAPH/Eurographics Conference on Graphics Hardware 2004* (2004), pp. 65–73.

[TC04] TUMBLIN J., CHOUDHURY P.: Bixels: Picture samples with sharp embedded boundaries. In *Eurographics Symposium on Rendering 2004* (2004), pp. 255–264.

[TC05] TURINI M., CIGNONI P.: Pinchmaps: Textures with customizable discontinuties. *Computer Graphics Forum 24*, 3 (2005), 557–568.