# Table-top Interfaces You Can Rotate: Rendering Issues

Billy Biggs

Faculty of Computer Science
Dalhousie University
biggs@cs.dal.ca

*Abstract*

Users often arrange items on a table-top with total disregard for the natural axis of the table itself. One often finds it more comfortable to tilt personal writing or working space to a comfortable viewing angle. We present examples of rendering issues faced by interfaces which allow rotation, and explore potential flaws in their implementation.

*Key words: Rotatable interfaces, gridfitting, computer supported cooperative work, tabletop displays.*

## 1 Introduction

Table-top displays are suitable for allowing users to rotate objects in the work area to fit their personal preference [4]. In a collaborative setting, user interface components are often oriented to better accommodate the group arrangement, tilting the work area such that all members of the group can easily read and interact with the items, and objects are tilted throughout the session.

Modern user interfaces often make use of axis-aligned detail for improving clarity and sharpness on low resolution displays. Cursors and interface widgets are often drawn by hand at the pixel level. The process of rendering high quality text at small pixel sizes is called font hinting, where a font or font system specifies how to ensure legibility of the individual characters, or glyphs, retain clarity. This process is usually done manually by the typographer, and is created specifically for axis-aligned text.

For rotated interfaces, these techniques become either useless, or in the case of font hinting, some automatic techniques may still apply while many are orientation specific. Current table-top displays are often constructed with a larger pixel size than a desktop display to provide a larger working area, but also increasing the need for rendering low-resolution details as clearly as possible.

We explore the issue of rendering interfaces that can rotate in an attempt to produce a set of guidelines for interface designers regarding the size and shape of graphics and text suitable for rotation without visible artifacts. We focus primarily on text rendering, as we are given a situation where a vectorized representation of the source is available, as well as a modified gridfitted version. We present requirements for rotated interfaces, discuss font hinting techniques and identify areas we believe cause problems, and discuss our method for exploring these issues.

## 2 Requirements for rotated interfaces

Graphical user interfaces often exploit the alignment of the pixels on a display to add sharp details. Window borders are discontinuities which occur on exact pixel boundaries, and single-pixel thin lines adorn many interface widgets. Cursors, window adornments, and icons are often hand edited at the pixel level. Upon rotation, we argue that many of these components will exhibit artifacts relating to undersampling.

### 2.1 Interactivity constraints

For a seamless user experience on a table, it is natural to allow objects to be rotated interactively. For example, a user may be provided with a rotation handle on each window or work area such that it may be angled dynamically. This places an additional constraint on our rendering engine: the engine should avoid having certain angles suddenly “pop” into clarity, as this effect is distracting.

## 3 Font hinting

Font hinting is a technique used to improve readability of text at small sizes relative to the pixel grid [6]. The two main font hinting systems in use today are the TrueType font hinting system [1], and Adobe’s Type 1 font hinting system [3]. In the TrueType system, font hinting is performed using a bytecode interpreter, where the typographer provides a program that performs modifications of glyph given the target output size, giving the ability to specify the output on a pixel level if necessary.

The process of font hinting generally involves perturbing the vertices of a vector representation of the font’s glyphs such that the following properties are met:

1. Each of the glyphs of a font at a certain size are not individually too large as to exhibit a difference in shade between characters;

2. Characters are spaced such that they do not touch;
3. Characters remain at a consistent alignment;
4. Symmetry properties of letters are preserved; and
5. Aesthetic features are preserved.

We may be able to preserve some properties on rotation, but other decisions by the font hinter may be inapplicable or even detrimental, or may cause discontinuities upon interactive rotation.

## 4 Method and rationale

Our intention is to better understand the following questions:

1. What elements of font hinting are applicable to rotated text?
2. Does it make sense to apply hinting at all before rotation?
3. What artifacts occur if we apply hinting rules during interactive rotation?
4. Is there a reasonable bound that we can place on line or detail size such that their geometric properties are preserved under rotation?

We believe that focusing on text rendering will give insight for rendering of other user interface components where both vector and pixel representations may be available.

We evaluate examples rendered using the open-source text renderer FreeType [5], our own 2D renderer for small pixel details, and rotated screenshots of common applications. Our evaluation of font hinting techniques is based on the bytecode interpreter methods available in the TrueType font system [1].

For evaluation of detail size and appearance, we provide some signal processing results. Also, we investigate size and shape geometrically by assuming that pixels are exact squares. We feel that table-top displays are likely to be driven either by LCD projectors or LCD panels, since the physical constraints of a CRT make it less suited for mounting under or on top of a table. Therefore, a geometric analysis of the output is warranted when evaluating detail size. For this evaluation we use the work of John Hobby [2].

## 5 Conclusions

We provide some reasonable bounds and recommendations for rendering user interfaces allowing interactive rotations and use at angles unaligned with the pixel grid. We do this by empirical evaluation of font hinting and rotation of small details, and some theoretical evaluation of the geometric properties of the results, assuming an LCD panel display with a box reconstruction filter.

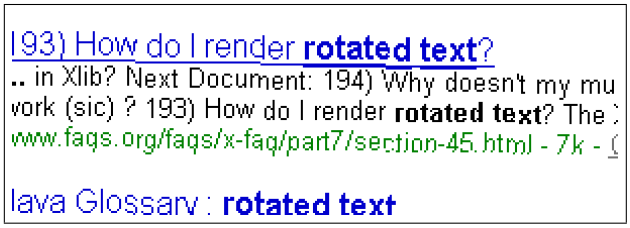


*Figure 1: Bitmap font rotated 3 degrees using nearest-neighbour scaling*

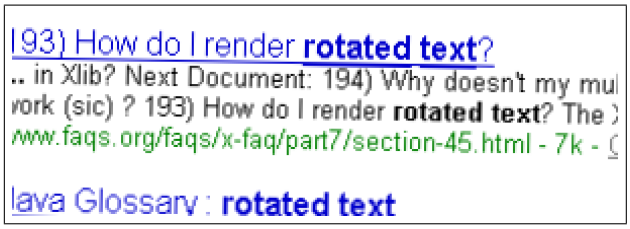


*Figure 2: Bitmap font rotated 3 degrees using bilinear interpolation*

## References

[1] Microsoft Corporation. The truetype font system. *http://www.microsoft.com/typography/default.asp*, 2003.

[2] John D. Hobby. Rasterizing curves of constant width. *Journal of the ACM*, 36(2):209–229, 1989.

[3] Adobe Systems Inc. Adobe type 1 font format. 1990.

[4] K. O'Hara and A. Sellen. A comparison of reading paper and on-line documents, 1997.

[5] FreeType Project. The freetype font renderer. *http://www.freetype.org*, 2003.

[6] Douglas E. Zongker, Geraldine Wade, and David H. Salesin. Example-based hinting of truetype fonts. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 411–416. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.