News from the HINT Project

Martin Ruckert

Abstract

The HINT file format[5] was presented at TEX Users Group 2019[4] and at TEX Users Group 2020[6], the first usable viewer for HINT files was presented. The HiTEX engine became part of TEX Live in 2022. This presentation will explore the changes that have taken place since then and what to expect in the future. The talk will focus on

- demonstrating the more recent versions of the HINT file viewer and the improvements in glyph rendering;
- demonstrating the use of links, labels, and outlines;
- explaining the capabilities of the HINT file format to convert pages to plain text for searching or text-to-speech processing; and
- presenting hints on how to design TEX macros for variable page sizes.

1 Displaying Glyphs

Initially, the HINT viewer did support only `.pk` fonts. These font files contain METAFONT fonts at a fixed resolution, usually at 600 dpi. Rendering such a font on a computer screen with a typically much lower resolution, was done in three steps:

1. Decoding the font file header and caching it for later use.
2. Decoding a glyph into a black and white bitmap and caching it for later use.
3. For each pixel on screen intersecting the glyph's bounding box
   - map the pixel center to a source point in the glyph's bitmap and
   - compute the pixel's gray value by linear interpolating the black and white values of the four pixels surrounding the source point in the bitmap.

Since high dpi values, often above 300 dpi, are common on small mobile devices, the results were more than acceptable on these devices. On ordinary computer screens typically with dpi values below 100, the results were insufficient. Especially the rendering of thin lines would distribute the available amount of black ink over a two pixel wide area and the line would fade away into a blurry light-gray.

Things changed with the use of the FreeType font rendering library[7]. This library can render PostScript Type 1 outline fonts at any resolution desired. After replacing the `.pk` fonts by `.pfb` fonts, the viewer could render the glyphs as gray-value bitmaps for the actual screen resolution[3]. To produce good looking glyphs from an outline font, first the positions of key points of the outline, for example the points where the outline has a horizontal or vertical tangent, will be rounded to the pixel grid. After that, pixels that are only partly covered by the outline will be assigned gray values depending on the amount of coverage. This will result in less blur and consistent stroke widths. Improving the readability especially for small font sizes.

The quality of the font rendering in the HINT viewer was, however, still inferior to a rendering of the same font by other programs. The reason was, that the viewer would not map the glyph bitmap one to one to the screen but instead would map the bitmap to TEX's exact glyph position – usually not aligned to the pixel grid – using step 3 as given above.

To improve the readability at small font sizes, the current viewer will round the glyph position to the pixel grid before rendering the glyph. And it replaces the linear interpolation of pixel values by using the gray value of the nearest source pixel. The rounding will occur only if the font size is below a given threshold. In principle the rounding can be split into rounding horizontal and rounding vertical position. While the first affects character distances, the latter moves entire lines and is less distracting. For a demonstration see [3].

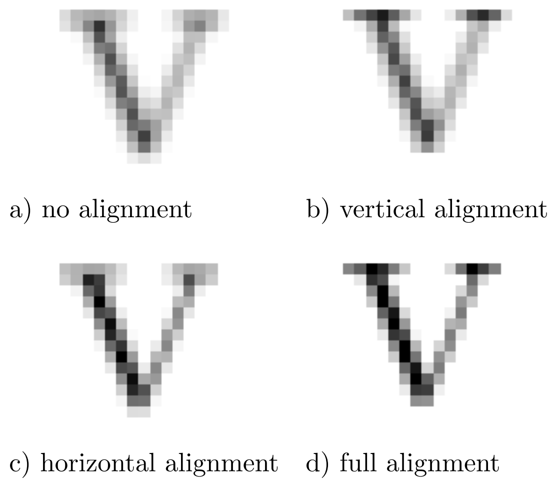

Figure 1: A cmr 10pt V with different alignment to the pixel grid.

Further improvements are possible, but not yet implemented. One method is oversampling, where a glyph is rendered at, for example, four different horizontal positions on the pixel grid. Choosing one of these four renderings, the horizontal glyph position must be rounded to 1/4 of the pixel size which is far less distracting. Another method is sub-pixel rendering. This method uses the fact that one white pixel on screen actually consists of three colored dots: red, green, and blue. So by considering them a independent light sources, the horizontal resolution can be tripled. This improves the positioning but leads to colored borders which some people find distracting.

2 Links, Labels, and Outlines

People my age have learned navigating through thick books already in primary school, if not in kindergarten. These skills are more or less obsolete when it comes to navigating through "thick" electronic documents. So good replacements are necessary. The most obvious point to start exploring a book is its table of content where for each section the corresponding page number is listed. The HINT file format supports the concept of a home page: a position in the document identified be the author that can be reached in the viewer with a single key stroke, touch, or click. The HINT document, however, has no fixed page numbers. The pages grow and shrink with the window size (and with the magnification factor). So instead a table of content must use a click-able link that brings you immediately to the section in question. Similar links are used for the table of figures, the index, and for all kinds of cross-references, be it to individual parts of the text, a figure, a table, a citation, or a displayed formula.

As an alternative to the table of content, the HINT file format also supports "outlines": A clickable table of content, hierarchically organized and displayed in a separate window. To allow optimal use of the available space, sub levels of the hierarchy can be hidden or expanded as needed[3].

In the mean time, the LaTeX hyperref package offers support for most of the above features.

In one respect HINT files are radically different from books or pdf files: There are no predefined pages. So following a link is not as simple as displaying a page with a given page number, but it requires finding two good page breaks so that the target is on the page between them. The algorithm used in the current HINT viewer is still under development and there are cases where the choice of page breaks could be better.

3 Designing Macros for Variable Pages

The traditional implementation of centering text is the `\centerline` macro. It expands to `\hbox` to `\hsize` { `\hfill` *text* `\hfill` } which will look nice as long as the *text* is shorter than `\hsize`. If the text is longer, it will produce an overfull box, stick out into the margin, and even goes over the edge of the window. A better solution uses TeX's line breaking procedure[3], which requires a vertical box.

```
\vbox{\rightskip 0pt plus2em
      \leftskip=\rightskip
      \parfillskip=0pt\parindent 0pt
      \spaceskip.3333em
  \xspaceskip.5em\relax
This is Text Centered on the Page
}
```

Letting `\rightskip` and `\leftskip` stretch enough, but not too much, so that the line breaking routine will try to keep the lines filled but still has enough room to produce decent lines. The inter-word-glue, on the other hand, is prevented from stretching. (It could be made to allow for some shrinking to gain additional flexibility.)

The only new feature introduced in HiTeX since 2019 is the support for `\vtop`. This is important because writing for variable page sizes often requires replacing a horizontal box by a vertical box to enable the breaking of paragraphs into lines. `\vtop` is required if multiple vertical boxes need to be aligned on the top baseline[3].

4 Searching

The user input into a search field is just a plain sequence of characters coded in UTF8 or some local encoding like ISO 8859-1. The text, as represented in a TeX document is far more complex and searching requires finding a match between both representations. Even if the input consist only of ASCII characters the HINT viewer must handle some special cases.

If the word the user wants to find uses a ligature, the match is made using the replacement characters, that are retained in TeX's ligature node. If the word on the page is hyphenated and split across two lines, the match must ignore extra characters inserted by the pre and post hyphenation lists as well as the space that is usually separating the word at the end of one line from the word that starts the new line. Indeed the HINT backend provides a function, that converts entire pages into sequences of characters moving from top left to bottom right, eliminating the effects of ligatures and hyphenations and condensing various combinations of glue – inter-word glue, baseline skips, left skips, right skips, and indentations to name just a few – to a single space. Kerns, on the other hand, are completely ignored. An infelicity here is the definition of the LaTeX macro, which uses a glue instead of a kern between 'A´ and 'T´. So you have to search for "`LA TEX`".

It is planed to use the page to string function also to feed a Text-to-Speech converter.

Currently searching does not work well with non ASCII characters, but it is planed to implement UTF8 as the default encoding used for HiTeX and HINT files.

5 New Viewers for Linux, MacOS, and iOS

Together with the viewers for Windows and Android, the applications for Linux, MacOS, and iOS complete the set of Viewers. The Windows application, being the oldest and my work-horse for conducting experiments, is the most complex. The application for MacOS is the most recent and was presented at Jonathan Fines's TeX hour[1, 3]. The application for Linux is the most simple. It consists beside the backend and the OpenGL renderer (shared between all applications) only of a 600 line main program[2]. This is a good starting point for writing your own viewer.

References

[1] Jonatan Fine, Martin Ruckert, et al. Rethinking TeX in STEM. `https://texhour.github.io/2022/09/29/rethink-tex-in-stem/`, 9 2022.

[2] Martin Ruckert. Hint source repository. `https://github.com/ruckertm/HINT`.

[3] Martin Ruckert. The HiTeX video collection. `http://hint.userweb.mwn.de/hint/video/`.

[4] Martin Ruckert. The design of the HINT file format. *TUGboat*, 40(2):143–146, 2019.

[5] Martin Ruckert. *HINT: The File Format.* 2019. ISBN 1-079-48159-1.

[6] Martin Ruckert and Gudrun Socher. The HINT project: Status and open questions. *TUGboat*, 41(2):208–211, 2020.

[7] David Turner, Werner Lemberg, et al. Freetype. `http://www.freetype.org/`.

◇ Martin Ruckert
Hochschule München
Lothstrasse 64
80336 München
Germany
`martin.ruckert@hm.edu`