Basic Type 1 Hinting

By David Lemon, Adobe Systems Inc.

Introduction

I'm David Lemon. I'm a type nerd. I'm also a manager in the Type Development group at Adobe Systems. People often ask me what I do, and I've decided the best answer is that my group takes typefaces and turns them into fonts. Although we use tools that are different from what most of you use, I'd like to talk about what we have in common, and help you make better fonts out of your designs.

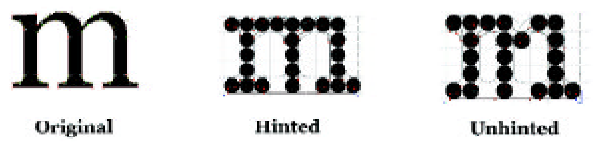
What I'll cover

Type 1 is the font file format used most widely by font creators. As you may know, Adobe is starting to work with a number of independent type companies. In looking at the fonts produced by a variety of vendors, we've noticed that virtually all of them could be improved significantly by better use of hints. I'd like to help that happen.

Complete Type 1 hinting can be done using one of a few programs proprietary to major manufacturers, or FontLab, or Fontographer. My talk assumes you're using an off-the-shelf, cross-platform tool. I'll discuss how to get the best results possible within the limits of the software, but the basic information applies to any application which can hint.

What does hinting do?

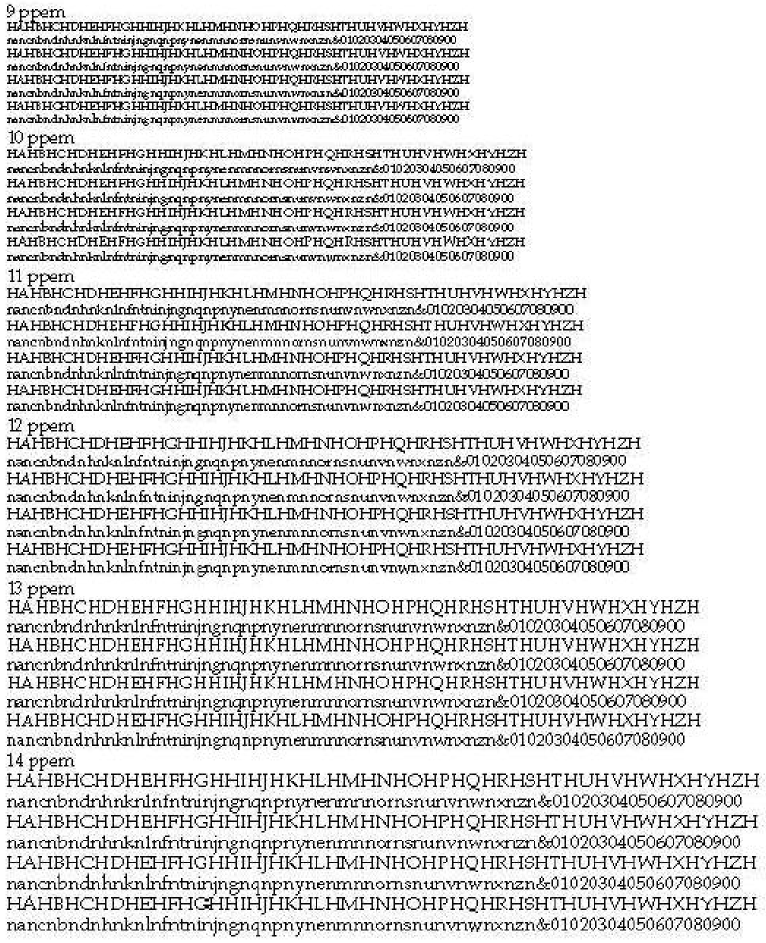
Hinting is information which goes to the rasterizer along with the glyph outlines, and is used to make better decisions about which pixels to turn on. Good hinting improves typographic quality by controlling alignment, weight and spacing. The main thing hinting does is to adjust the way the outline fits on the rectangular grid of pixels which will be used to turn it into a solid shape. Without hinting, even features which are identical in weight or width will vary, because they'll lie on the grid differently. The fewer pixels there are to work with, the more critical this is, because each pixel represents a larger portion of the glyph. So although Type 1 hints work at all resolutions, our primary concern is on-screen rasterization, and - for text designs - text sizes at 300 dpi. In the past, it may have been acceptable to say that you care only about how your design looks at high resolution, but today your customers have good reason to use your fonts on-screen as part of their final product, which makes hinting essential. Hinting is that final step which takes the all the effort you've put into your design, and makes the results fully available to your customers.



How do we approach hinting?

While automatic hinting - performed by any application - is better than nothing, it leaves a lot to be desired. I recommend the same approach we use at Adobe: Auto-hint the font, then test it to see how well it rasterizes. At anything other than high resolutions, most visible problems can be attributed to poor hinting, and many of them can be eliminated or at least minimized using the manual hinting features in FontLab or Fontographer. The most common problems involve uneven alignments or gross differences in stem weights. A normal font can be well-hinted in an hour or two if you know what you're doing, so it's not a huge task. Of course it helps a lot to understand what hinting can do, and thus what changes to make, but I'll say this repeatedly: The essential step is to identify poor rasterization and make changes until the problems are resolved.

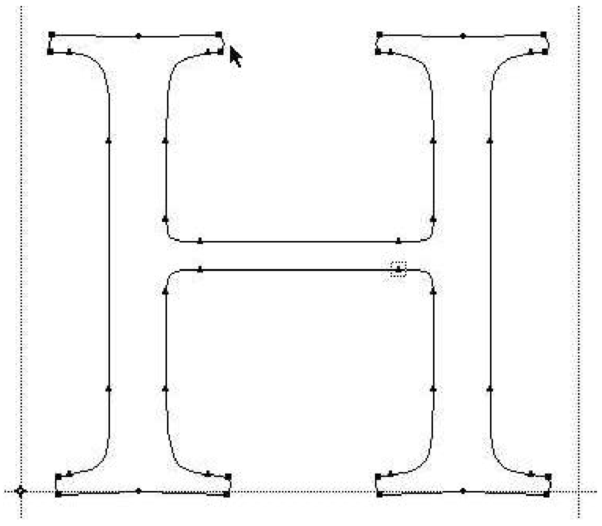
Build a new version with different hints and a different name, and compare the results side-by-side. Attention to testing is the heart of good font software; fortunately, FontLab/Fontographer and ATM allow you to do this almost instantly. Here's one way to compare two versions of a font:


9 ppem
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
10 ppem
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
11 ppem
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
12 ppem
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
13 ppem
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
14 ppem
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900
HAHBHCHDHEHFHGHHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZH
nancnbndnhnknlnfntninjngnqnpnynenmnnornsnunvnwnxnzn&01020304050607080900

Preparing the outlines

Hinting works best with outlines which are consistent in weight and alignment. Features should not include unintentional irregularities. Of course, if features are supposed to be subtly different, don't compromise them; as Einstein said, make things as simple as possible, but no simpler.

This includes using no more lines or curves than necessary to describe the intended shapes. The major exception to this general rule is that it's important to place endpoints at horizontal and vertical extrema of significant features, since the hints control the outer edges, and work better with endpoints to grab. In Fontographer, the Clean Up Paths feature does a pretty good job of placing points and minimizing elements. In FontLab use the font audit feature. If you find that it changes the intended shapes, you can reset it to use less simplification. And of course you can always edit some more.



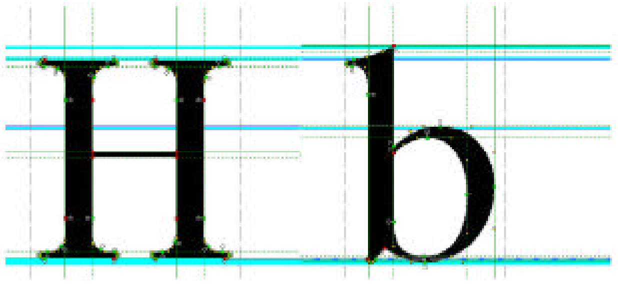
Here's the element structure of an H in Adobe Garamond. Note that, while there are minor extrema at the tips of the serifs, we didn't worry about putting points on them.

It's also important to make sure the outline "paths" are written in the correct direction; this is the order in which the individual elements are stored and drawn. In Type 1 fonts the correct order is counter-clockwise for black-filled paths, and clockwise for counters; you can visualize walking along the path with the black area always on your left. If you're not sure the direction is correct, you can use the Reverse Contour feature (FontLab) or the Correct Path Direction feature in FOG.

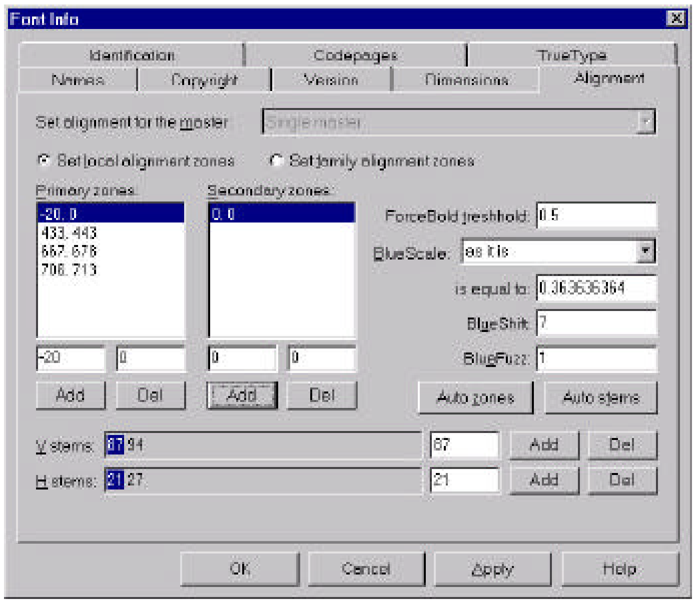
Type 1 hints are designed to be simple. There's a lot of intelligence built into the Type 1 rasterizer, so it needs only minimal hinting information to do its job. There are three kinds of Type 1 hinting information. These are font-wide behaviors, hints to control common features in individual glyphs, and some special cases.

The most obvious font-wide behavior is the top and bottom alignment zones which help preserve the horizontal alignment crucial to Latin text faces. Generally you have a zone for each set of related features (the baseline, x-height, figure height, cap height, ascender height, descender depth, superior baseline and so on), but any number of feature types can share a single zone if they're logically the same height.

Here, an H and b use the same zones at the top and bottom.



To be compatible with some older models of PostScript-language printers, it's best to use no more than five zones for the various feature tops, and no more than six for the bottoms. <Yes, I know this sounds backwards.> You want to set the flat side of the zone (for example, the bottom of a top zone) at the exact height of the flat features, and the far side of the zone (for example, the top of a top zone) to the exact height of the overshooting features. Thus, a cap height zone would generally extend from the top of an H to the top of an O. Of course, if the heights of glyphs for other characters vary, you'll want to adjust the zone to include them. Much of the hinting adjustment we do involves fine-tuning these zone values. There are several special behaviors associated with these zones (known as BlueShift, BlueFuzz and BlueScale in Type 1 parlance), but Fontographer doesn't provide a way to adjust them, so I won't discuss them here, except to say that under the default behavior no zone should be more than 25 units deep, and any two zones should be separated by at least three units. Fortunately, the default behavior is fine most of the time. In FontLab you can set values for these behaviors on the alignment page of the Font Info dialog box:



Set alignment for the master: Single master

(●) Set local alignment zones ( ) Set family alignment zones

Primary zones:
-20, 0
433, 443
667, 678
706, 713

-20 0

Secondary zones:
0, 0

0 0

ForceBold treshhold: 0.5
BlueScale: as it is
is equal to: 0.363636364
BlueShift: 7
BlueFuzz: 1

V stems: 87 94 — 87
H stems: 21 27 — 21

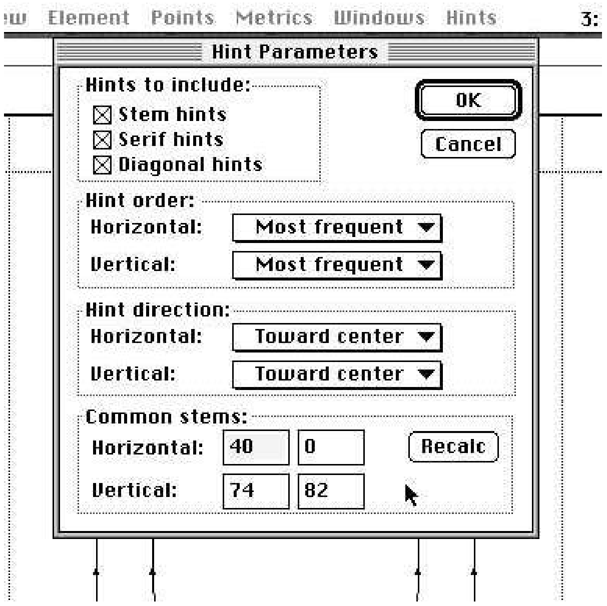
Stem weights

Another font-wide behavior is the stem weights provided for the font. It's generally best to enter two values for vertical stems and one or two for horizontal, listing the values in ascending order. In a standard Latin font, the values should represent the optimal weight of the normal lowercase and capital stems (hence the two values). These values - especially the first vertical value - are used for a number of rasterization purposes, so it's important to make them as representative as possible. Two of the most obvious uses include letting ATM know when to turn off synthetic emboldening in a bold design, and serving as target values, to ensure consistent stem weights at low resolutions. In a design where the stems vary slightly in weight, fine-tuning these target values can make a significant difference.

The values selected automatically are a good starting point, but generally not optimal. FOG specifies stem weights in the Hint Parameters dialog under the Hints menu. Here's the dialog; I'll come back to some of its other features in a minute.

Stem hints

Two or more stems hinted at the same weight will always rasterize at the same number of pixels, and stems with a value similar to the target stem weights will be held to the same number of pixels as those weights until they reach a size where their difference from those weights is big enough to represent legitimately. Horizontal stem hints which have one edge in an alignment zone will have their vertical position controlled by that zone, preserving the alignment inherent in the design. If there's a problem in alignment, it can be useful to verify that the features are hinted in the appropriate zone.



Hint Parameters

Hints to include:
- ☒ Stem hints
- ☒ Serif hints
- ☒ Diagonal hints

Hint order:
- Horizontal: Most frequent
- Vertical: Most frequent

Hint direction:
- Horizontal: Toward center
- Vertical: Toward center

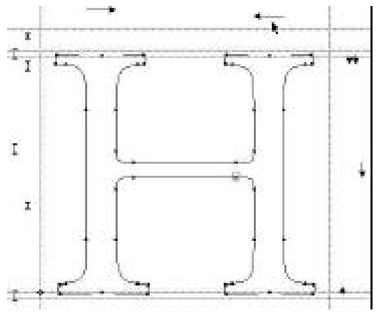
Common stems:
- Horizontal: 40 0
- Vertical: 74 82

Note that a horizontal stem hint can have either edge in an alignment zone, but should not have both edges in zones, because the hinted stem value can conflict with the behavior of the two different zones it's tied to. I'll talk about how to resolve this in a minute.

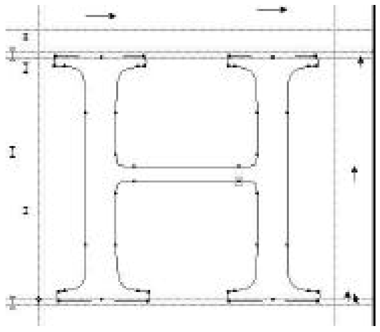
Adding and adjusting stem hints

The default values in FOG apply stem hints from the outside toward the center of the glyph. Since the Type 1 format stores a hint as a horizontal value paired with a horizontal distance, or a vertical value paired with a vertical distance, hints that are represented as going from right to left, or from top to bottom will be written to the font program as negative widths. This is non-ideal for Type 1 fonts. I recommend you reset the hint-direction behavior to have vertical hints written from the bottoms to the tops of horizontal stems, and from the left to the right of vertical stems - that is, increasing in value as measured by the coordinate space. This behavior is specified in the Hint Parameters dialog under the Hints menu. FontLab corrects negative hints automatically on export, but if you want to change the direction of a hint yourself you just use the Reverse command on the hint popup menu.

If you need to change the direction of an existing hint, Fontographer allows you to do so by switching to the Hints layer and clicking the cursor on the hint's arrowhead, or by selecting the hint and using the Flip Hint Direction command under the Hints menu. The vertical stem hint here is going in the wrong direction:



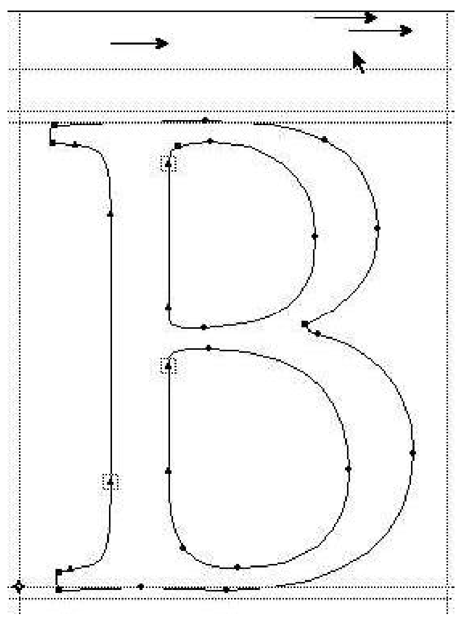
here it's corrected:



When the application finds hints that aren't exactly what you want, you'll need to add new ones or tweak existing ones. To add a new hint in FOG, select the stem in question and then select Make Horizontal Stem or Make Vertical Stem under the Hints menu. To adjust an existing hint, select it and open the Selection Info dialog under the Elements menu.
In FontLab you add a new hint by holding down the control key while clicking and dragging a guideline from the top or side rulers of the Glyph Edit window. You adjust them by clicking and dragging them or by right-clicking on them and using the commands in the popup menu.

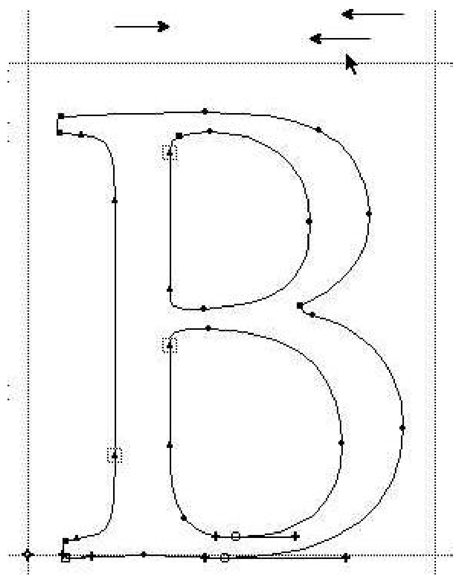
Overlapping hints

As I mentioned earlier, Type 1 stem hints are a value and a delta. They aren't tied to particular elements, but control all elements that match them. For this reason, the Type 1 rasterizer won't accept two overlapping stem hints at the same time. Well-hinted fonts supply several sets of hints for each outline, along with instructions for when to use which set, to ensure that the optimal set is used for each part of the path as it's drawn. Since FOG doesn't support this feature, it's important to choose hints carefully in cases where there'd normally be an overlap. In general, the choice is obvious: Hint the most significant features. FontLab supports hint "replacement" (instructions which let the rasterizer know when to use each hint set), which allows you to create overlapping hint sets that can be processed correctly by a Type 1 rasterizer.



When two stems of equal significance overlap, it's usually best to hint the stem farthest from the center of the glyph. The overlapping edge of the unhinted stem will be controlled by the hinted one, minimizing the problem.
The vertical stems on the bowls of a B or 8 often present such a case. When the values of one stem lie entirely inside the values for another, and the two are similar in significance, it's generally better to hint the larger stem, thus including the smaller one. This case is pretty rare.

FontLab autohinting creates overlapping hints and uses hint replacement to make sure the rasterizer doesn't ignore them. But since auto-hinting in FOG will often create overlapping hints (but not hint replacement), it's important to know which ones will be built into the Type 1 font program. The hints represented by arrows closest to the glyph outline are the ones that will be used. They can be reordered by selecting one and dragging it with the cursor moused down. Occasionally this doesn't seem to work; in that case, you can use the Forward and Back commands in the Selection Info dialog under the Elements menu.

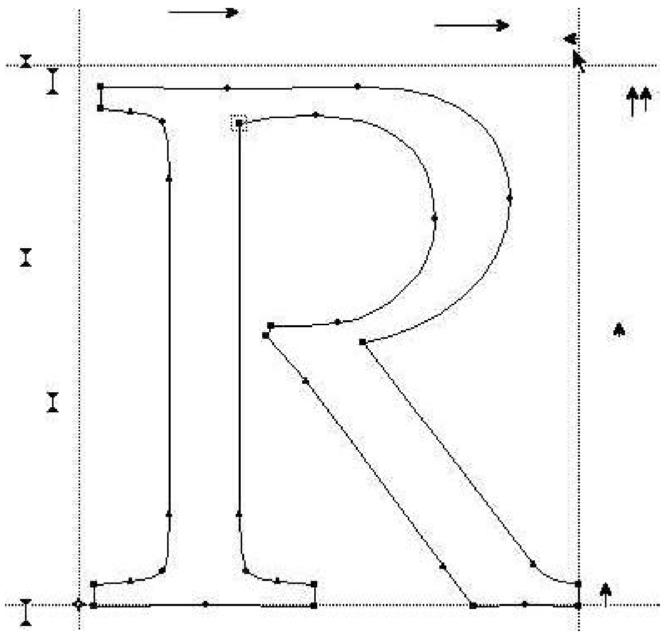


Here the hints have been reordered:

Most designs include vertical stems whose top or bottom alignment we want to control with alignment zones, but which don't have horizontal stems (like serifs, for example) at their top or bottom to be hinted in the zones. This situation is widespread in sans-serif designs, but also occurs in cases like the apex of A, v and w. I mentioned a related case earlier, where the top and bottom of a single "stem" lie in different alignment zones - for example, a sans-serif capital I.

In these cases, Type 1 fonts use a virtual horizontal stem to hint the aligning feature. These virtual stems are called ghost hints, and are written just like any other horizontal stems, with two important differences. First, they have a negative value (thus represented in Fontographer as going from the top down instead of from the bottom up, or from right to left instead of left to right), and second, they use a value of exactly 20 units (at the tops or rights of features) or 21 units (at the bottoms or lefts). This is the main reason to pay attention to hint direction; with the wrong direction, stems that aren't ghost hints will be treated as if they were (if they happen to have the expected weight), and intended ghost hints may be treated like regular stems. This wasn't much of an issue in the past, but ATM 4 has improved code for dealing with ghost hints. FontLab creates ghost hints during autohinting, but you can create them manually as well by using the Links tool.

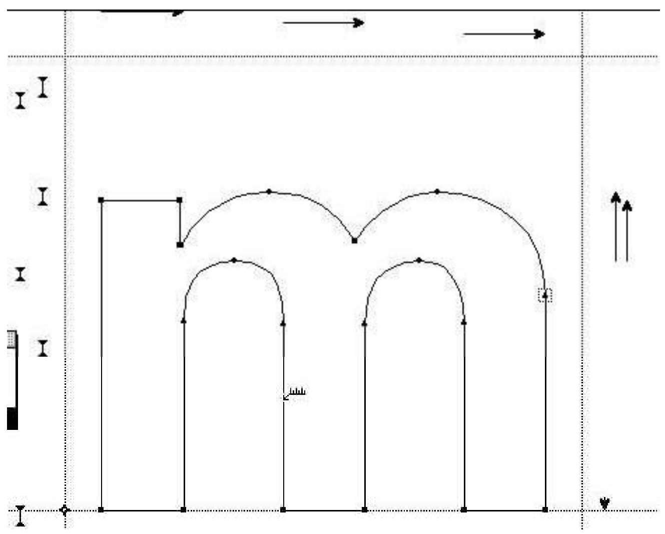
Since Fontographer users can't use sets of hints, vertical ghost hints provide a workaround for a few cases. For example, an R may have two serifs which extend equally far to the left, and you want to ensure that consistency. Obviously the vertical stem is more important to hint, but if the serifs are more than 21 units from the stem, they can use a horizontal ghost hint. Again, this is identified by its negative value, so FOG would show it going from right to left. I've put one on the tip of the leg serif as well; at the right, it needs to be negative 20 units wide.



You can add a ghost hint in FOG by selecting a stem hint of the same direction and using the New Hint command in the Selection Info dialog, or by selecting top and bottom points and using the Make Horizontal Stem or Make Vertical Stem command in the Hints menu, then adjusting the results in the Selection Info dialog.

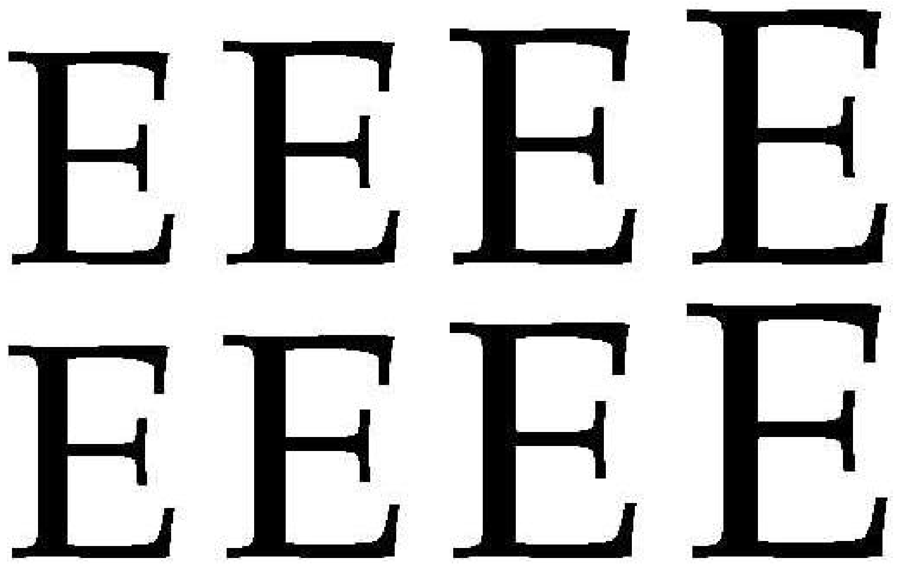
Hints for special cases

One special case occurs when you have a glyph with two counters which are the same width or depth. At smaller sizes and lower resolutions, a one-pixel difference in these counters is really ugly. The most obvious example is the lowercase m in many Roman designs, but this condition can also be found in glyphs for other characters, including an ellipsis, a serif T, or vertically, in the division symbol.



Many designs, like Italics, will have hardly any cases, while others - like ITC Bauhaus - may have lots. Hints for these features are called vstem3 or hstem3 in the Type 1 format, although we usually just call them counter hints. FOG is limited in its ability to recognize these glyphs, but usually manages to find at least the m, which is usually the one that needs it most. The only thing you can do about this in FOG is to make certain the outer stems are hinted at the same widths, and the counters themselves are the same widths.

Shallow curves which are oriented horizontally or vertically, and are thus aligned with the pixel grid, are one of the trickier features to rasterize well. Type 1 fonts allow the creator to add "flex" hints to these features to control them better. Flex replaces the curve with a straight line until its depth is large enough to represent fairly, at which point flex controls the pixel placement on the curve to keep it looking reasonable.



Here's a series of cupped serifs at different sizes, with and without flex. Note how flex gives them consistent behavior. Cupped serifs are the most common case for flex. We place the center of cupped serifs at the flat side of alignment zones when possible. FOG's implementation is rather buggy; it often fails to apply flex where it would be helpful, and sometimes causes characters to not print. Flex also adds to the size of a font, so it's best not to use it unless there are significant features in the design which could benefit. You'll want to do some experimentation, and skip flex if it's not helping. FontLab 3 doesn't do Type 1 flex hints, but the new T2/CFF font format flex hints will be supported in FontLab 4.

In Fontographer, Flex is enabled by a checkbox in the Generate Font Files dialog (with the Advanced setting) under the File menu.

An in-depth explanation of the hints is in the Type 1 spec, also known as the Black Book. An on-line version, and related documents, are available in the Developer section of Adobe's web site. Here's the specific URL: www.adobe.com/supportservice/devrelations/technotes.html.

That covers the general picture for hinting Type 1 fonts. You see it really is pretty simple, and a little attention will go a long way. I believe good hinting is only fair to your customers, and a sign of pride in your work. In the end, the name of the game is the same as in the type-design process: Look, tweak, and test again until you're satisfied. I look forward to seeing better Type 1 fonts out there next year.

Thanks!