I guess there are two ways to think about distances with variable fonts. CVTs are typically used when pre-measuring distances, and when combined with the cvar, adapt those distances with the variations. The FN 86 model is essentially this way, but it was not adapted for variable fonts, hence it didn't use a CVT for the distance. That could be relatively easily updated to a new function that takes a CVT as an input rather than a distance. It might, though, be complicated to update the autohinter to handle this (and more than likely the cvar would have to be updated manually). The current autohinter, not being aware of variable fonts, calculates distances from the default outline. You said that "As of right now, it is based on a measured outline distance, that is part of the function 86, that the autohinter generates, which I assume you also think is not correct?" I believe that, and I may be wrong here as I've not recently studied FN 86, the outline distance is pre-calculated by the autohinter, and doesn't internally use instructions that are based on original outline distances. If the function needs to rely on pre-calculated values, the only way to handle that with variable fonts is through a CVT (and cvar)—therefore, what the autohinter generates is only correct for the default instance but most likely incorrect for all other variations (assuming there is an outline change in that dimension).

The second way is to rely on original outline distances and the TT instructions that work with original outlines. That technique should work fine with variable fonts as it is relative to the variation outline. The MIAP in your technique is rendering specific, but the MDRP is not—and, understandably, it forces a minimum distance. That distance will not account for different rendering methods.

The description Beat provided for FN 86 states:

- convert distance (fUnits) to pixels, round, and space child from parent by result this essentially implements the functionality of an MDRP[m>RWh] instruction for composites which lack a suitable implementation of "original" coordinates.
- roundMethod 0, 1, 2, and 3 round down to, to, up to, and to half virtual grid
- function assumes cvt #20 is reserved and can be used temporarily in here...
- storage 16 contains a threshold ppem size. if zero, ignore the threshold, otherwise if less than or equal to threshold, use 64 (26.6) #samples/pixel.

His main point in the comment is "implements the functionality of an MDRP[m>RWh] instruction for *composites which lack a suitable implementation of "original" coordinates*. [emphasis mine]". He is correct in that in the rasterizer, when we create a composite, we lose the original outline data—and we try to recreate it by unscaling the current outline. But that is just a guess on the part of the rasterizer. With FN 86, he is trying to handle this better.

My overall point is that if you want to use a solution that is like FN 86, it will have to be converted to a CVT based solution to work with variable fonts. If, on the other hand, your technique of using instructions that take measurements from the original outline, then that is fine, and it will also work with variable fonts. Ultimately it is what looks the best for a reasonable cost.

I hope this helps to explain my thinking.

GregH

From: MICHAEL DUGGAN <michaeldug@gmail.com>
Sent: Wednesday, July 7, 2021 5:26 AM
To: Greg Hitchcock <gregh@microsoft.com>
Subject: RE: VTT Autohinter and accents

Thanks Greg

1. I am not sure I grok why this needs to be based on a cvt. I understand the need for interpolated values, but in practical terms, using a dist that uses the outlines measurement, seems to work just fine, when I test in a browser, where I can choose any intermediate variable in the range. Perhaps you could if/when you have time, let me know the exact reasons this needs to be based on a cvt? As of right now, it is based on a measured outline distance, that is part of the function 86, that the Autohinter generates, which I assume you also think is not correct? as I showed the problem in my email. As I am using VTT for production purposes for Variable fonts, I am looking for practical, and correct solutions, and ideally any technically correct solution, that produces code that makes perfect composites, would be generated automatically by the VTT Autohinter, without the need for editing any values individually. If this is something you proposed new function would look like, please tell me more, I am not clear how this would work in practice?
2. For now, the practical effect works well, as tested in VTT and in the browser, this code keeps a minimum distance at small sizes, which is desirable across the whole variation design range, and also maintains the correct distance between the base glyph and the accent, which is much better than the current output from VTT.

3. If this could be modified at some point to not output x-code, great!

Thanks
Mike

From: Greg Hitchcock
Sent: Tuesday 6 July 2021 18:28
To: michaeldug@gmail.com
Subject: RE: VTT Autohinter and accents

Hi Mike,

Our current focus on VTT is extracting the VTTTalk compiler and TrueType assembler from the code and separately providing them as open-sourced components. At some point I expect we'll also do the same for the autohinter. So, in the near term, we don't have any resources to fix the autohinter.

Here are a few thoughts.

1. In variable fonts, outline distances need to be handled by CVTs. The CVTs can then be interpolated to handle the variations. Would it make sense to provide a new function that emulates FN 86's behavior, but uses a CVT value for the distance? That would require making sure that the CVT is setup correctly, but

this shouldn't be too difficult working with the outline producer who presumably understands all the axes.
2. I'm not sure the impact of the MDRP when y-direction AA is being used.
3. This makes sense. I'll talk to Paul about this, but as I noted, not real high priority.

GregH

From: MICHAEL DUGGAN <michaeldug@gmail.com>
Sent: Tuesday, July 6, 2021 6:56 AM
To: Greg Hitchcock <gregh@microsoft.com>
Subject: VTT Autohinter and accents

Hi Greg

I have a few other issues with the VTT Autohinter output, and I wanted to ask your help/opinion on, see the details below. The email has quite a few details and a question for you in number 2 below.

There are a couple of major problems, in how the VTT Authinter Light deals with accent positioning. In some projects I have done, I have had to go through all of the accents to fix individually. This is very time consuming, and if VTT handled accents better, it would cut down the production time considerably.

The problem

1. VTT Light Latin autohinter adds code to position accents in the y-direction, ensuring there is always a minimum distance of at least one pixel between the base glyph and the accent. The method used to position the accent uses function 86, and a measured outline distance from base glyph to accent.

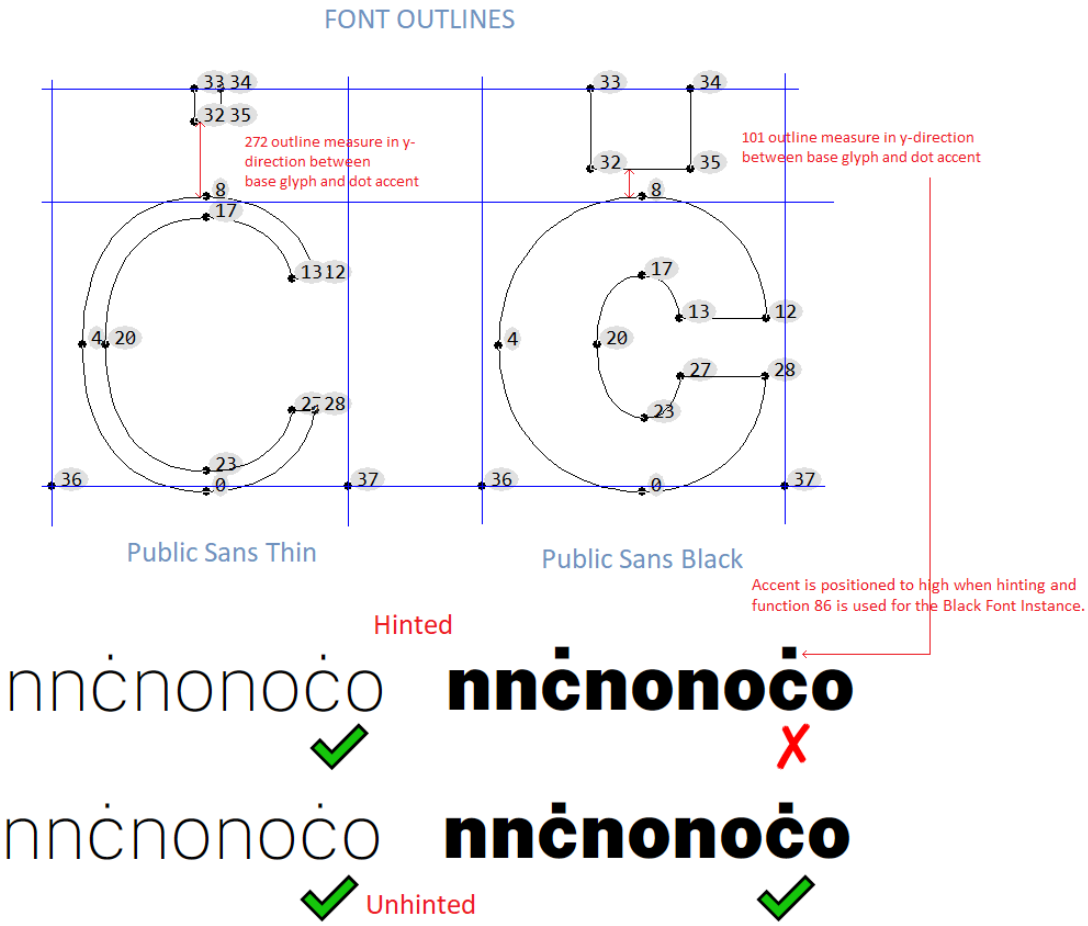
   CALL[], <child>, <parent>, <roundMethod>, <minDist?>, <distance>, 86

   This is fine for static fonts, as there is only one measurement, but fall apart when hinting variable fonts, where the distance from base glyph to accent can vary significantly. See the example below. I am using an Open Source font Public sans, (VTT Light Latin autohinted version is enclosed in the ZIP file).

   In this glyph 'cdotaccent' 0x10b (GID 228) you can see the problem in the graphic below. The distance used for the Thin Variant, is correct. In the Black Variant of the outlines, the measured outline distance is much smaller = 101 f units. This all makes sense from an outline point of view, where the accent in the light is highter, keeping the typographic colour of openness where as in the Black design the accent needs to be much closer to keep the right typographic colour. This is a good example of a common problem with Variable fonts and VTT accent positioning, showing the current method used by VTT Autohinter to be unworkable**. (see enclosed font PublicSans[wght].ttf** 0x10b (GID 228)

   USEMYMETRICS[]
   OFFSET[R], 223, 0, 0
   OFFSET[R], 602, 507, 0
   SVTCA[Y]
   CALL[], 32, 8, 1, 1, 272, 86

SHC[2], 1
SVTCA[X]
CALL[], 32, 35, 4, 34, 12, 87
SHC[2], 1



2. The method I am using to fix this issue is the following. I remove the Function 86 that positions the accent in the y-direction. Because the function requires a distance**, it cannot be used successfully for Variable fonts, where the distance between base glyph and accent can vary significantly.**

I use this simple code, point 8 is the top of the ‘c’ positioned on height 7, the lowercase round overshoot cvt, then MDRP to point 32 on the accent, to keep a minimum distance, but also to keep the correct distance in the other variant weights.

- Could you have a look at this **(see enclosed font MikeSans[wght].ttf** 0x10b (GID 228), and let me know if this looks correct as a solution to you? It appears to work very well, the accent is kept clear by at least one pixel, which is desirable at all sizes and weight variants in the Variable font, as well as keeping the right hinted distance, for al weights also.

SVTCA[Y]

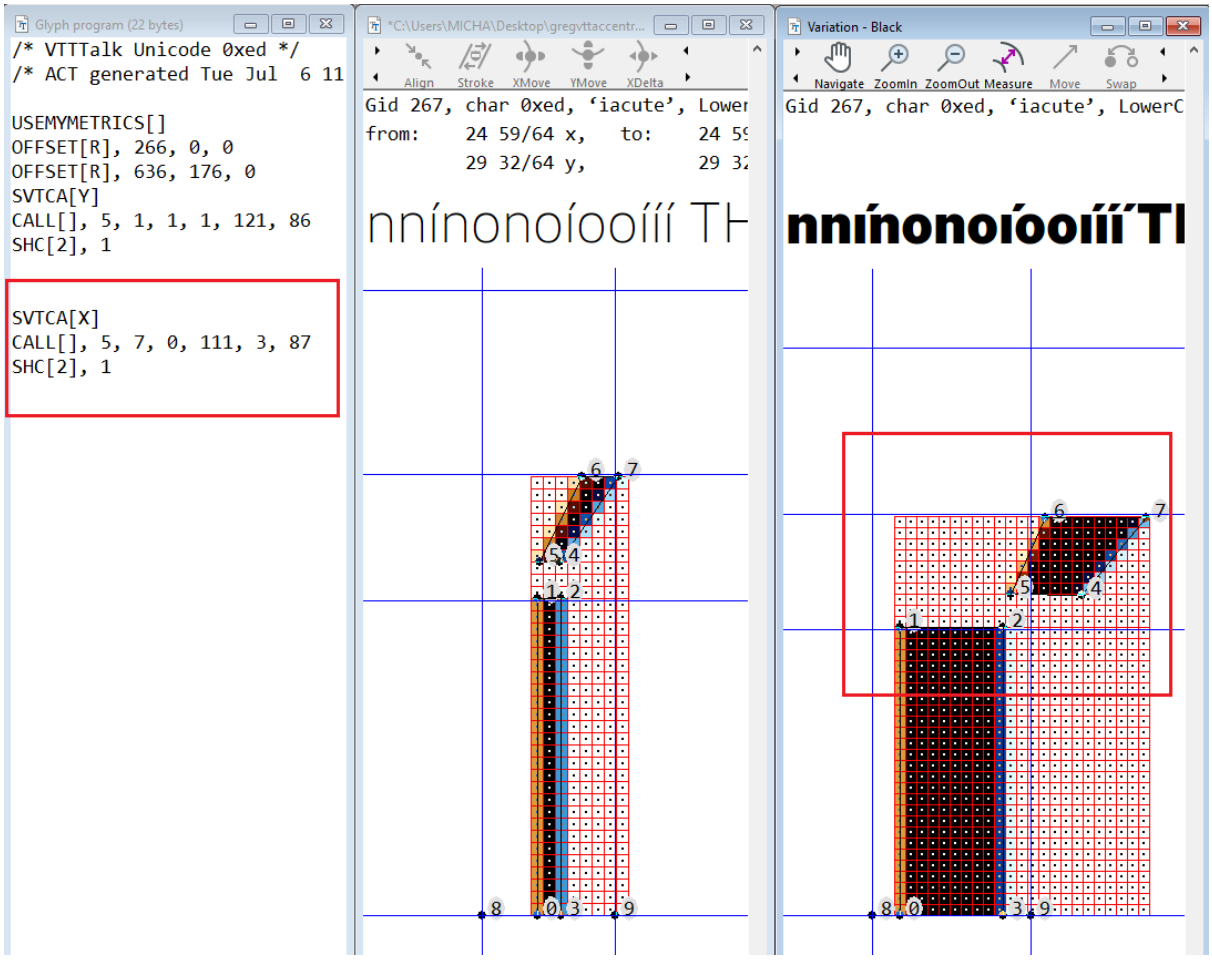
CALL[], 8, 7, 114
MDRP[m>RWh], 32

IUP[Y]
IUP[X]

3. VTT Light Latin autohinter also outputs x-positioning code for accents, also unusable, see below, the Black accent is positioned in the grid-fitted outline way to the right.

   (see enclosed font PublicSans[wght].ttf 0xed ‘i acute’ (GID 267)

   Again this is an example of a common problem. As a side note, if x-positioing code is not needed, as I have found, perhaps the Autohnter could be instructed not to output x-positioning code, for composite glyphs. Actually, x-code has been disabled on output for all other glyphs already, in the Autohinter, as an option. However if you choose to disable x-code in the Autohinter, it still outputs x-code for composites. I don’t know what Function 87 is doing here, using something called ‘partial factor’, but it is not a good solution for Variable fonts, and I always have to remove all x-code for accented glyphs.



Thanks Greg, if you could take a look I would appreciate any comments and or if I can help with any solutions. Cheers Mike