Boring Expansion
extensions to OpenType

TypeLab @ Typographics 2023

behdad

Boring Expansion: What’s in a name?

bor·ing

adjective

not interesting; tedious.

bore

verb

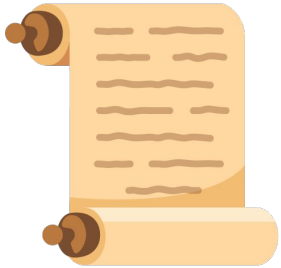
make (a hole) in something, especially with a revolving tool.

Boring Expansion: What is it?

Extend the font format:

- To address decades-long limitations.
- Add highly-desirable features.
- Maintain general design structure, such that:
- Fonts not using the new features are generally backward compatible with the existing format.
- It’s okay if adoption takes time. Many of these ideas were suggested back in 2015… If we had acted on them then, we could use them today. We have to act now, so we can rely on them in 5 / 10 years. That is just how format changes work.

Boring Expansion: So far…



- avar2
  https://github.com/harfbuzz/boring-expansion-spec/blob/main/avar2.md
- Cubic `glyf` outlines
  https://github.com/harfbuzz/boring-expansion-spec/blob/main/glyf1-cubicOutlines.md
- VarComposites
  https://github.com/harfbuzz/boring-expansion-spec/blob/main/glyf1-varComposites.md
- beyond-64k
  https://github.com/harfbuzz/boring-expansion-spec/blob/main/beyond-64k.md

avar2: What is it?

- New version of the `avar` table that supports arbitrary mapping across variation axes.

- Specification:
  github.com/harfbuzz/boring-expansion-spec/blob/main/avar2.md

- Support: FontTools, HarfBuzz, FreeType, macOS/iOS

avar2: Use cases

- Designspace warping
- Higher-order interpolation (HOI, aka Non Linear Interpolation)
- Parametric fonts
- Designspace fences

avar2: How it works

- In avar1, each axis can be *remapped* individually and piecewise linearly.

- In avar2, each axis can be *remapped* as a function of *all* input axes. This allows saving corner masters in many designs.

- “Mom, You Sure Can Re-hydrate a Pizza (or: avar table version 2.0)”
  docs.google.com/presentation/d/1jLiUvh4PoYz2sw0T0Fygj2HoMrBhmJQuGgAmoMc_Sns
  youtu.be/j7unMVZOcaw

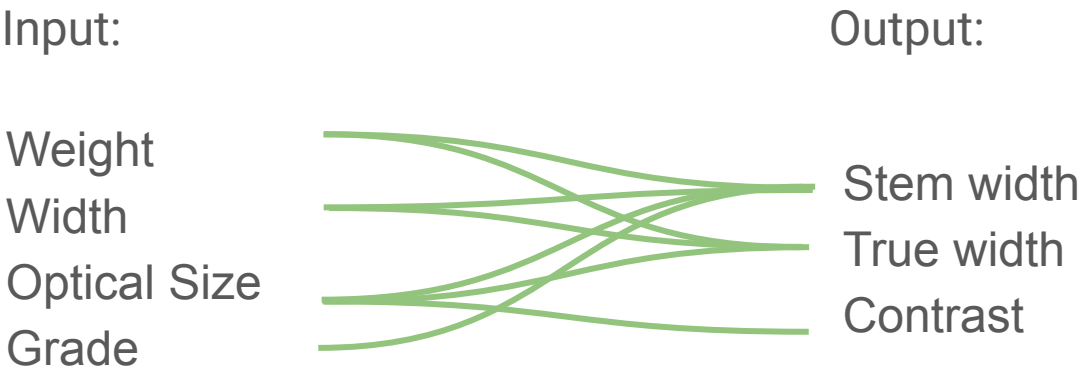
avar1: Old model

Adjustments limited to one axis at a time.

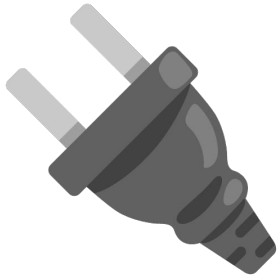
Input:		Output:
Weight	———	Weight
Width	———	Width
Optical Size	———	Optical Size
Grade	———	Grade

avar2: Designspace warping

Adjustments across axes.

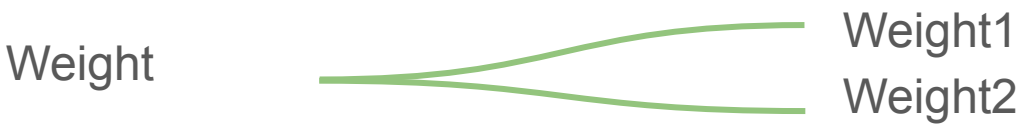


avar2: Higher-order interpolation (aka HOI)



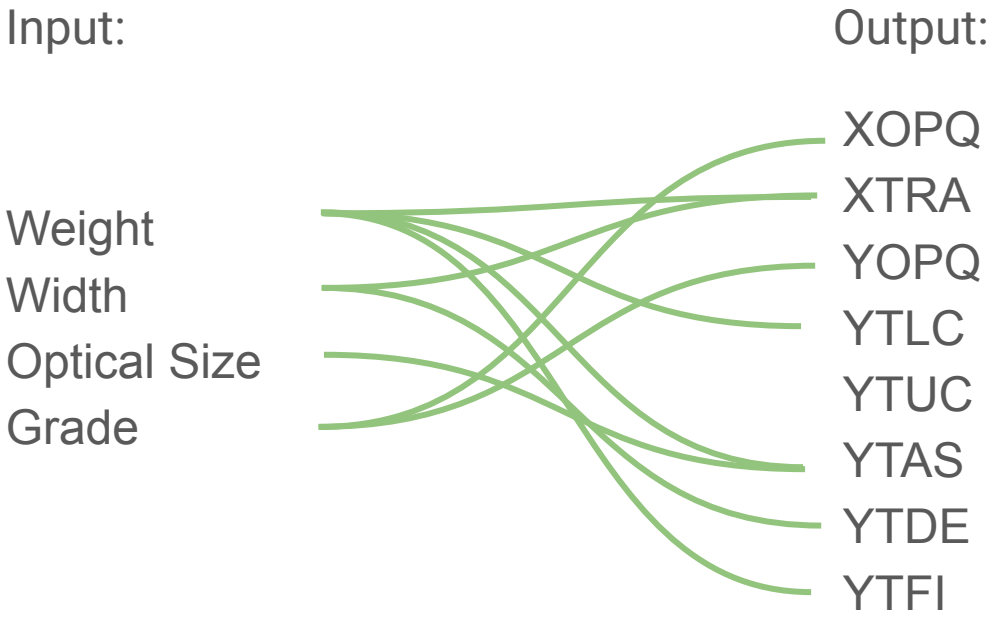
No user-visible hacks needed for higher-order interpolation.

Input: Output:



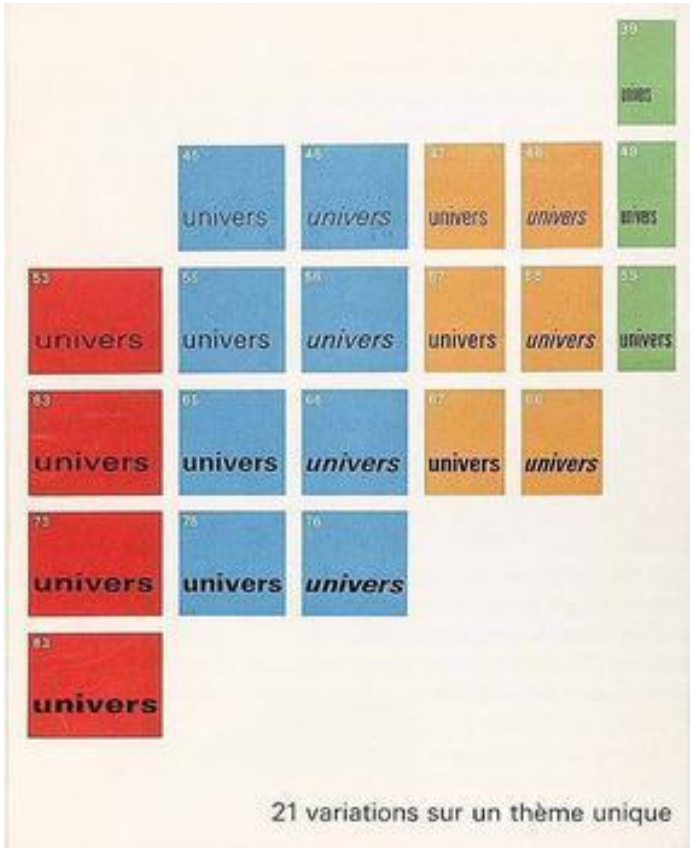
avar2: Parametric fonts

Vastly fewer corner masters needed in parametrically-designed fonts.



avar2: Designspace fences

Make undesirable regions inaccessible.



Deberny & Peignot, 1964

avar2: Results

- Prototypes: Parametric fonts:
  - 80% file-size savings in Roboto Flex
  - 70% file-size savings in Amstelvar

Cubic `glyf` outlines: What is it?

- Extension to the `glyf` table to support cubic Bezier curves.

- Specification:
  github.com/harfbuzz/boring-expansion-spec/blob/main/glyf1-cubicOutlines.md

- Experimental support: FontTools, HarfBuzz
  Patch: FreeType

Cubic `glyf` outlines: Use cases

- Higher-fidelity representation of outlines, especially important for:
  - lower units-per-EM fonts,
  - very light weights,
  - variable fonts with many interacting masters, as conversion errors accumulate.

- Slightly smaller fonts.

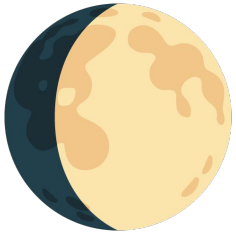
Cubic `glyf` outlines: How it works

- New flag in the SimpleGlyph outline specification to imply cubic Bezier off-curve points.

- Implied on-curve points between two cubic segments supported.

- Hinting capability unaffected.

Cubic `glyf` outlines: Results



- Roughly 3% to 10% savings by just rebuilding fonts from existing sources, depending on how curve-heavy the design is.

VarComposites: What is it?

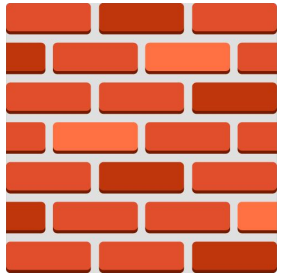
- Extension to `glyf` table to support *variable components*, aka “smart components”.

- Specification:
  github.com/harfbuzz/boring-expansion-spec/blob/main/glyf1-varComposites.md

- Experimental support: FontTools, HarfBuzz

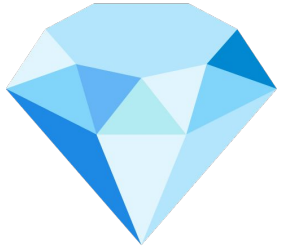
- Based on proposal by Black Foundry / Just van Rossum.

Classic components



- References another glyph *at the font's current variations settings*
- Translation offset: *variable*
- 2x2 transformation matrix for scale, rotate and shear: *not variable*!

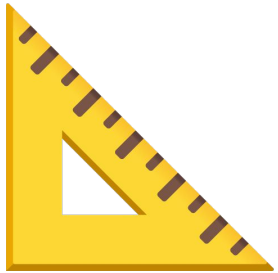
Variable components



- References another glyph *an arbitrary variable location* in the variation space

- Translation offset: *variable*
- Rotation: *variable*
- Scale: *variable*
- Shear: *variable*
- Center of transformation: *variable*

VarComposites: Use cases

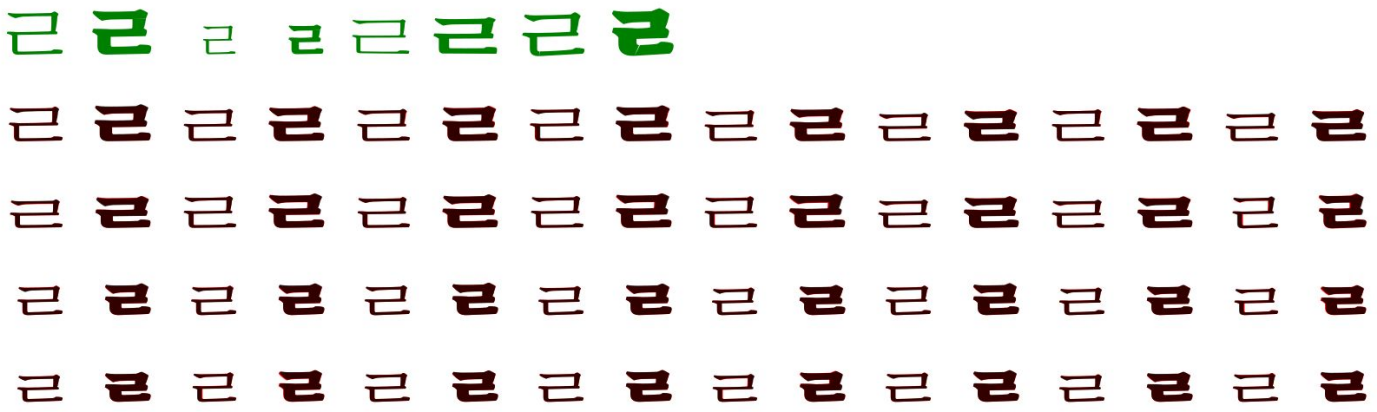


- Han fonts
- Hangul fonts
- Tibetan fonts
- Latin stencil / modular fonts
- Icon fonts
- …

VarComposites: How it works

- Each VarComposite glyph refers to components at certain location in the component glyphs' designspace.

- Same component can be used to provide a variety of shapes.



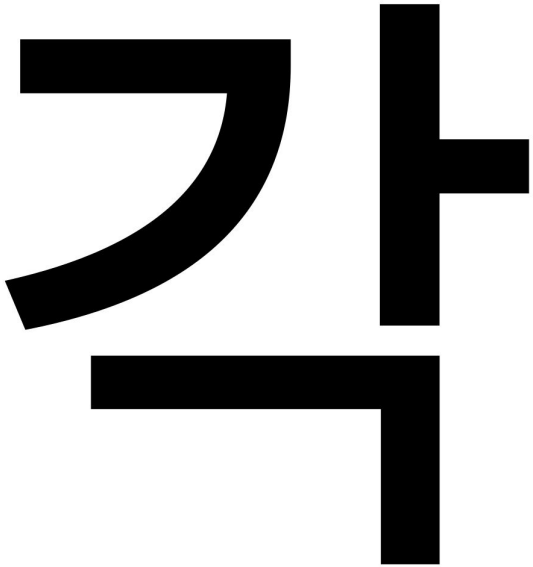
VarComposites: Results

- Han varfont with 2 weights & ~44k ideographs:

  63% file-size savings

- Hangul varfont with 2 weights & ~11k syllables:

  72% file-size savings



beyond-64k: What is it?

- Extension to the font format to support more than the current glyph limit: 65,535!

- Specification:
  github.com/harfbuzz/boring-expansion-spec/blob/main/beyond-64k.md

- Experimental support: HarfBuzz
  Proof-of-concept: FontTools, FreeType

beyond-64k: Use cases

- Massive CJK fonts

  Currently Noto CJK / Source Han CJK fonts at maximum glyph count, and missing many CJK characters.

- Pan-Unicode fonts

  Currently pan-Unicode fonts use 100+ separate files that are hard to organize / order, with distinct family names. This feature allows eg. having one `NotoSans.ttf` as One Font to Rule Them All.

  Simplifies document fallback where file-size is not constrained, or incremental transfer is available

beyond-64k: How it works

- 24-bit glyph indices instead of 16-bit, allowing 16m glyphs.
  Ought to be enough for everyone, right?

  For context, Unicode allows up to ~1m codepoints,
  of which currently ~150k are used.
  - Noto fonts use less than 200k glyphs total (due to many scripts' conjuncts)

- Many tables updated to allow more than 64k glyphs.

- Most intensive changes applied to the GDEF / GSUB / GPOS tables, also alleviating the long-standing *offset-overflow* issues. Designed for code-reuse via templates.

beyond-64k: Results

- 15MB merged `NotoSans.ttf`, containing all scripts except for CJK.
- Over 100,000 glyphs.
- `GPOS` table over 1.5MB, compiled successfully without overflows.

- Drawback: HarfBuzz binary size growth +100kb (+%10)

Boring Expansion: Future work

- Next-gen Variations:

  Native support for higher-order interpolation? More compact storage?

- Next-gen Layout:

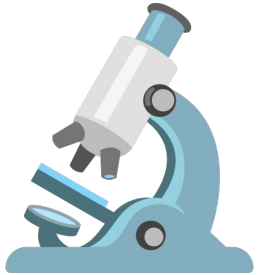
  Address long-standing issues in OpenType Layout: a move lookup, arbitrary glyph filtering, a more compact / powerful contextual lookup…

- Streamline / relax the spec where possible, deprecate old cruft.
- Specification of variable justification

Variable justification: What is it?

- Use the `jstf` variable axis tag for text justification.

Variable justification: Use cases



- Arabic kashida and other stretching
- PdfTeX-like “micro-typography”

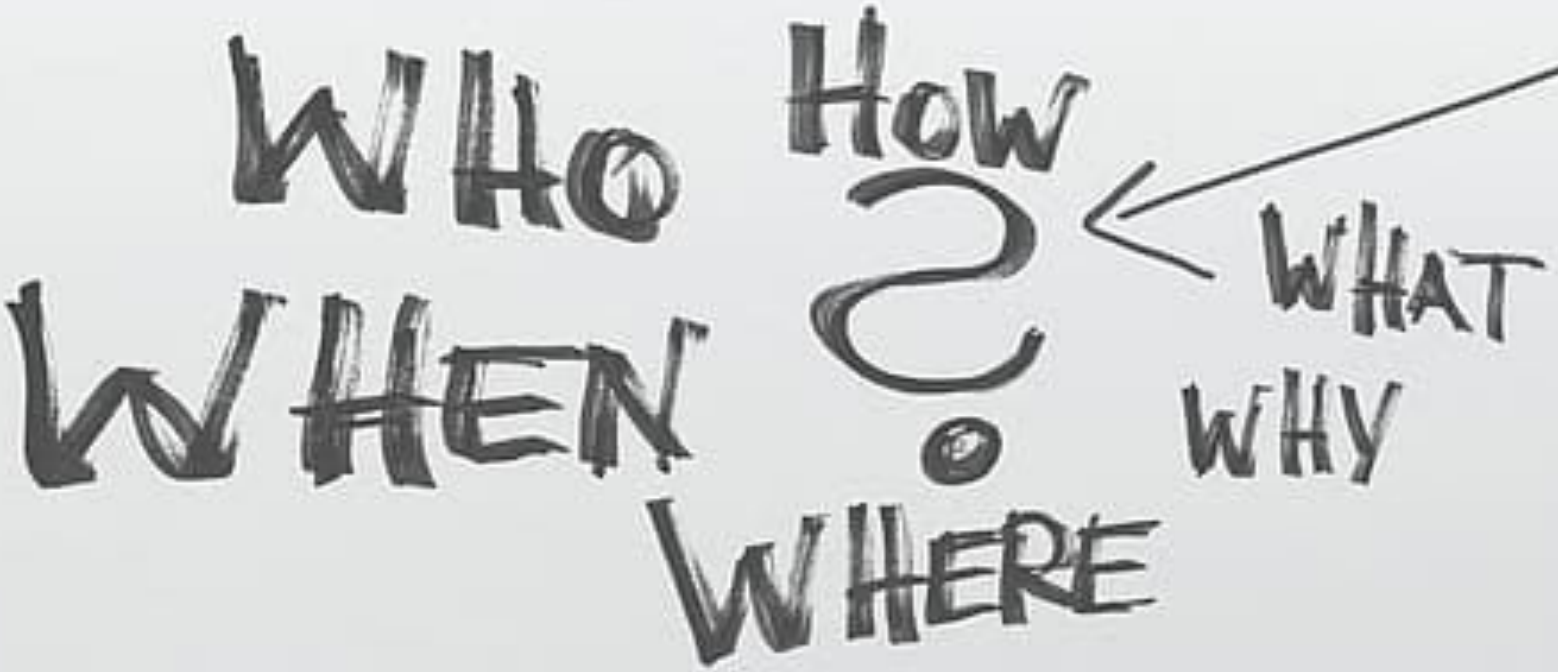
Variable justification: How it works

- Font defines the `jstf` variable axis tag.
- Various justification priorities all baked into the one axis.
- Typesetting engine finds the right variable axis setting that fits the line.

TODO:

- Figure out solution for glyph substitutions, as current FeatureVariations-based solution is inadequate.

Variable justification: Results


WHO
HOW
WHEN
?
WHAT
WHY
WHERE

Forum 1: BE Github

Forum 2: Type events

Forum 3: MPEG & OpenType