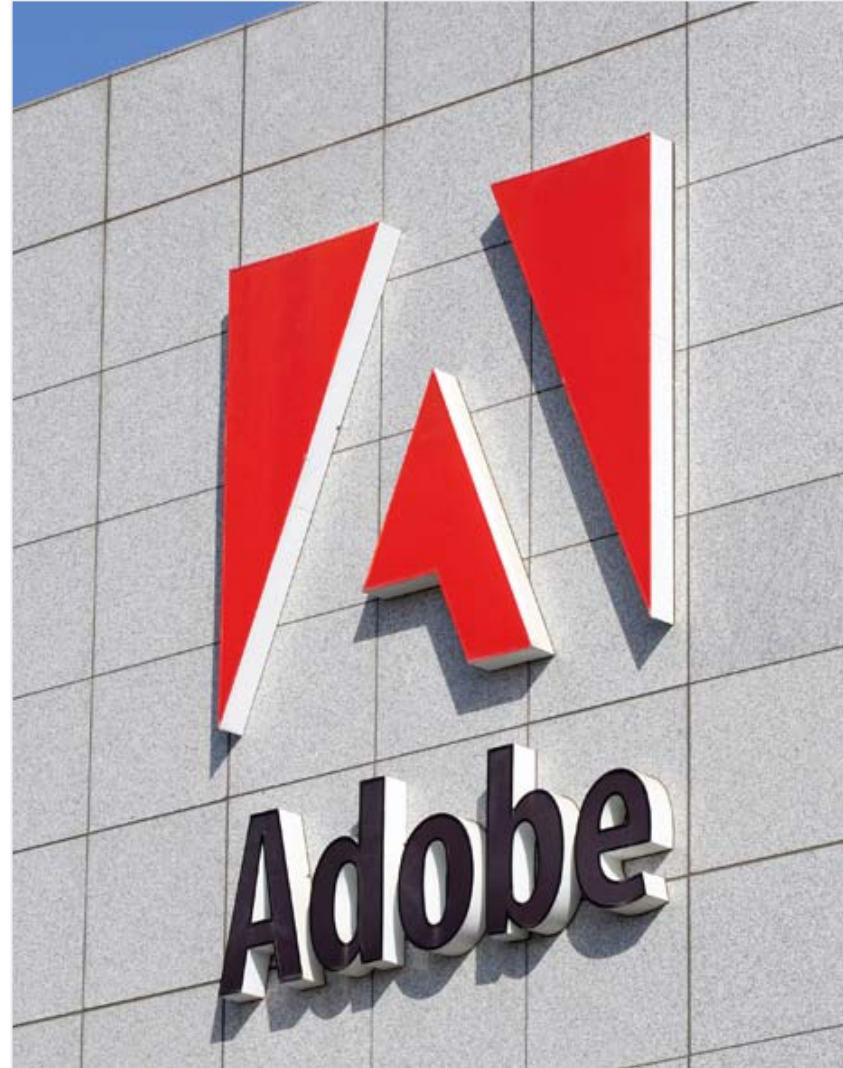


# Windows, CSS font families & OpenType 1.5

Thomas Phinney  
Program Manager  
Fonts & SING technologies

28 September 2006



# Agenda

- Why should we care?
- How & why Windows Presentation Foundation (WPF) processes fonts for font menus

# Why care about this?

- What is Windows Presentation Foundation?
  - WPF (“Avalon”) is new with Windows Vista
  - But also available for XP at the same time Vista ships
  - Apps have to go out of their way to use it
- WPF may change your font’s menu name family and style to meet its own needs
- Affected: TrueType and OpenType fonts
  - WPF does not support Type 1 (“PostScript”) fonts at all
  - Type 1 fonts still supported in GDI applications under Vista
- You might care about how your font shows up in WPF application menus
- OpenType 1.5 gives you some additional controls over WPF treatment
  - **Red:** New in OpenType 1.5 and supported in WPF 1.0
  - **Orange:** New in OpenType 1.5 and not yet supported in WPF

# WPF & CSS

- WPF processing goal: Fit fonts into CSS style families
- What is CSS?
  - W3C “Cascading Style Sheets”
- Fonts within a CSS family are differentiated by:
  - “WWS” family
  - Weight
  - Width (“Stretch”)
  - Slope (“Style”): regular, italic, or oblique

# WPF's Procrustean bed

- WPF looks at both names and table values to determine WWS
  - When they differ, complex processing to determine which value is “correct”
- In WPF, Arial, Arial Narrow & Arial Black can all be one happy family
- But fonts that differ on other grounds cannot. For example:
  - Adobe families with four optical sizes
  - Family with “regular” and “inline” variants
- NOTE: rules are very complicated!  
**Strongly recommend** reading upcoming MS white paper on the subject.

# Comments on CSS family model

- Yes, it's broken
- But it's a standard
- Many other programs are interested in using it
- Probably other platforms too

# Apps & Font Access in WPF

- App refers to font by WPF family name, plus values for weight, width & slope
- Windows determines closest available match and returns it
- Similar to today's Windows (GDI); can return synthetic bold or oblique
- WPF also shows synthetic bold and italic in default app font menus ☹️
  - But they are segregated/labeled so you can tell they're synthetic 😊
- If specific characters are not available in selected font, WPF performs **font fallback**
  - Outside the scope of this discussion

# WPF Processing: Overview

- Analyze each single font for WWS values & names
  - Assign face names and appropriate WWS values
- Analyze the family to ensure all members differ on WWS basis
  - **Anything that WPF didn't recognize as relevant to WWS earlier, gets left on the family name, not the style name!**
  - Essentially splits families into two or more as needed

# WPF single-face processing: Data sources

- Examine OS/2 table
  - usWeightClass (or head table macStyle bit 0)
  - usWidthClass (or head table macStyle bits 5 and 6)
  - fsSelection bit 9, and bit 1 or head table macStyle bit 1
- Examine name table
  - nameIDs 1, 2, 16, 17 (and 21, 22)
- Font family with faces that differ only in WWS “opts out” of name analysis
  - Set fsSelection bit 8 in all fonts within the family
- If results of above differ, WPF must resolve those differences! (see later)

# Menu name rules: First round, initial names

- Decide which names to add together for the initial font name

	Family nameID	Subfamily nameID	
First choice	<b>21</b>	<b>22</b>	New WWS name IDs
Second choice	16	17	“Preferred” names
Third choice	1	2	Old Windows-style names

# WPF single-face processing: Process

1. Determine the canonical name table language for the font.  
All name analysis is based on this language.
2. Determine the canonical legacy and preferred names (previous chart).  
The preferred names are the starting point for analysis.
3. Combine the preferred family and face names  
(Regular, roman, book, normal and upright are all treated as equivalent styles)
4. Extract any style (slope) name
5. Extract any stretch (width) name
6. Extract any weight name  
**(what's left is the family name)**
7. Determine FontWeight based on extracted weight & weight value from font os/2 or head table.
8. Determine FontStretch based on extracted stretch & stretch value from font os/2 or head table.
9. Determine FontStyle based on extracted style & style value from font os/2 or head table.
10. Extract numbers that describe font style, weight & stretch from face names.
11. Determine localized names

# Menu name rules: Conflict resolution?

- Will bypass extraction in determining family name if all members of family either
  - Have fsSelection bit 8
  - Have name table IDs 21 and 22

# FontWeight processing rules

- If name processing yields nothing, use the value derived from bits
  - OS/2 usWeightClass, else head macStyle
  - WPF supports usWeightClass values 1-999
- If both the bit weight and the name weight are lighter than Normal (400), use the font specified weight.
- Else, if both the bit weight and the name weight are heavier than Medium (500) and the bit weight is not Bold, use the bit weight.
- Else, if the bit weight and the name weight differ by no more than 150, and the font specified weight is neither normal, nor medium, nor bold, then use the font specified weight.

# Width (FontStretch) processing rules

- If name processing yields nothing, use the value derived from bits
  - OS/2 usWidthClass
  - Else use macStyle bits 5 (Condensed) and 6 (Expanded)
- Else, compare the two values.
  - If both are narrower than regular, use the value derived from bits.
  - If both are wider than regular, use the value derived from bits.
  - Otherwise, use the value extracted from the name

# Slope (FontStyle) processing rules

- If name processing yields nothing, use the value derived from bits
  - OS/2 fsSelection else head macStyle)
- If styles derived from **both** name and bit settings, the name is preferred

# Special case processing

- “Frutiger” style two-numbered faces
  - Frutiger, Helvetica Neue, etc.
- LT Univers style three-numbered faces
- CJK-style weight abbreviations
  - Kozuka Mincho H (Heavy), EL (ExtraLight), etc.

# Collisions?

- In case of two faces seeming to be identical to WPF, one is discarded.
- Generally, only likely with different versions of “same” font
  - Or fonts that would have similar collisions in other systems
- Could also happen from erroneously setting fsSelection bit 8 or name IDs 21 and 22.

# Simulated fonts

- If font doesn't have style linked bolds for all combinations of style and stretch, synthesize them
- If font doesn't contain matching obliques for all non-sloped faces, synthesize
- Any family that has italics but not obliques gets synthesized obliques added!
- Setback for good typography
- But WPF is very clear about which faces are synthetic

# New OpenType options for WWS families

- **OS/2 fsSelection bit 9** to denote an oblique font (set italic bits as well)
- **OS/2 fsSelection bit 8** to denote a WWS-only font (use “preferred family/subfamily,” nameIDs 16 & 17)
- **name table IDs 21 and 22** for WWS family and subfamily names (instead of using bit 8 and and nameIDs 16 & 17)

**Better by Adobe.™**