About Face

Andrew Glassner

I love letters. From the squiggly g to the stately m, our modern alphabet offers a wealth of beautiful shapes.

One of the great benefits of the revolution in desktop publishing is that it has opened up the doors of typeface design to anyone with a computer and patience. Talent and a sense of aesthetics aren't required to create a valid typeface, but they don't hurt.

Dozens of Web sites offer visitors thousands of free typefaces (see the "Further Resources" sidebar for pointers to all the sites mentioned in this article). Most of these free fonts are the work of inspired amateurs who had an idea for a typeface; they created it using commercial design tools, and then released their work into the world. You can also buy collections of typefaces on CD compilations, saving you the trouble of downloading them one by one. Commercial foundries offer many professional typefaces as well, created by skilled and trained designers.

I love browsing new typefaces and collecting the ones I like. This collecting pays off when a graphic-design job requires me to pick just the right typefaces to go with the artwork, text, and overall feel of the piece.

I'm pretty discriminating in what I choose to save from public-domain sites, but when I find one that I like, I download it for my collection. The quality of the typographical information provided with a free font varies: few include kerning information, it's rare to see any kind of layout hinting, and most are missing at least a few characters like ampersands or semicolons. But if you're willing to live with these limitations, you can find some great designs.

Between the free typefaces I've saved from the Web, those that I've gotten with magazine CDs or packages I've purchased, and the commercial typefaces that I've bought for specific jobs, I've now got more than 9,000 typefaces on my computer.

Of course, they're not all installed. Every typeface installed in an operating system consumes system resources, and slows down everyday tasks. Happily, several programs exist that let you browse through all the font files on your computer.

Even with one of those programs, 9,000 typefaces is a lot to look through. In practical terms, it's way too many. I recently designed a logo for a company, so I had to look for a good typeface for the company name. Using a commercial font browser, I set the name of the company in the preview window, and then stepped through every typeface on my hard drive so I could see how the name looked in that font. Because I had to visually consider each one, my top speed was about 2 fonts per second. A little math shows that if I ground through without a single break, I could have theoretically looked them all over in about an hour and a quarter. In reality, my eyes glazed over after a few minutes, and it took about four hours to consider the whole collection. When I finally finished looking through everything, I knew I never wanted to go through that again.

Categories

Nobody wants to sort through 9,000 typefaces every time they get a new job. Commercial foundries often label their work with a wide variety of keywords to make them easier to find. For example, Apple uses 28 different categories such as Blackletter, Cyrillic, Glyphic, Monospaced, Opticals, Ornamentals, and Swash. Planet typography has a different set of categories, but just as useful.

Searching through categories is great when you have a general feeling for what you want, and all of your fonts have been given useful and accurate labels. Unfortunately, free fonts rarely come categorized. In my collection, I have only a few hundred commercial typefaces; the rest are nothing more than a family name and, if I'm lucky, an indication of whether the font is regular, bold, or italic.

I thought about sitting down and going through all my typefaces and assigning categories to them. This wouldn't be difficult, but it would be colossally boring and time consuming. I could get a reasonably good list of categories from any of the commercial catalogs, but assigning them one by one to each face would take forever. Even if I wrote a program to help me out, I figured it would take at least a whole day to go through everything. Even then, a huge number of the public-domain typefaces are so oddball that they would end up in the miscellaneous or novelty categories, which would defeat the whole point of the exercise.

My next thought was to write a program to do this categorization for me, but I threw that idea out almost immediately. The categories for many typefaces are subjective, and even the most common ones can be unclear. For example, consider the letters in Figure 1 and ask yourself which ones should be labeled as serif or sans-serif fonts (broadly speaking, the serif is the lit-

 Published by the IEEE Computer Society

Further Resources

To create the figures for this article, I used display fonts taken from *ClickArt Fonts 2* (published by Broderbund, 2003) and the *2000 Font Collection* (published by Greenstreet, 2003). Many of the fonts on these CDs are also freely available on the Web, but it's still worth buying the packages. It's more convenient than downloading them all one by one, and these companies seem to spend some time cleaning up the font files.

Many free fonts available online have corrupted or badly formed descriptors in the file, which means that some programs can't use a given typeface. Of course, you only discover the failures when you're trying to finish a project only moments before the deadline.

I've had no problem reading any of the fonts on these compilation CDs, which makes them worth the purchase price to me. Be aware that the quality of the fonts in these collections tends to vary considerably.

You can read about the type metrics saved in Adobe fonts in the document "Adobe Font Metrics Format Specification File" by Adobe Systems. Version 4.1 is available online at http://partners.adobe.com/asn/developer/pdfs/tn/5004.AFM_Spec.pdf.

The Panose specifications are laid out in the "Panose 2.0 White Paper" by Michael S. De Laurentis. It's available at http://www.w3.org/Fonts/Panose/pan2.html. Another source for Panose data is the "Panose Classification Metrics Guide" available online from AGFA Monotype at http://www.panose.com/hardware/pan1.asp.

Many great tools are available for browsing fonts on your computer. On my PC I use Printer's Apprentice (http://www.loseyourmind.com). Other popular tools are Typograph for the PC (http://www.neuber.com/typograph), and Font Book for the Mac (http://www.apple.com/macosx/features/fontbook).

You can see the type classifications used by Apple at http://www.adobe.com/type/browser/classifications.html, and those used by Planet Typography at http://www.planet-typography.com/manual/families.html.

If you're looking for the name of a particular typeface, check out two great Web sites. Indentifont asks you a series of questions based on the qualities of the typeface you're looking at (http://www.identifont.com). What The Font?! lets you upload an image of a few characters and responds with the name of the typeface (http://www.myfonts.com/WhatTheFont).

If you enjoy the thrill of the hunt, and discovering new amateur typefaces as they're developed, lots of sites exist that collect these typefaces, often providing versions in both PC and Macintosh formats. DaFont (http://www.dafont.com/en), and Abstract Fonts (http://www.abstractfonts.com/fonts) both sort the typefaces by category. Other sites I periodically check are High Fonts (http://www.highfonts.com), and 1001 Fonts (http://www.1001fonts.com). Once you've browsed the collections at these sites, most let you sort by date, so you can quickly catch up on what's been added since the last time you stopped by. Many other smaller sites specialize in different varieties of typefaces.

Of course, don't forget professional, commercial fonts. You must pay for these, but your money buys you a high-quality typeface with sophisticated kerning and hinting information, ligatures, and other professional details that will make your typeset work look professional and polished. The Adobe Type Library provides a wide variety of quality typefaces (http://www.adobe.com/type/main.jhtml). A good consolidated source of fonts offered by many different foundries is MyFonts (http://www.myfonts.com).

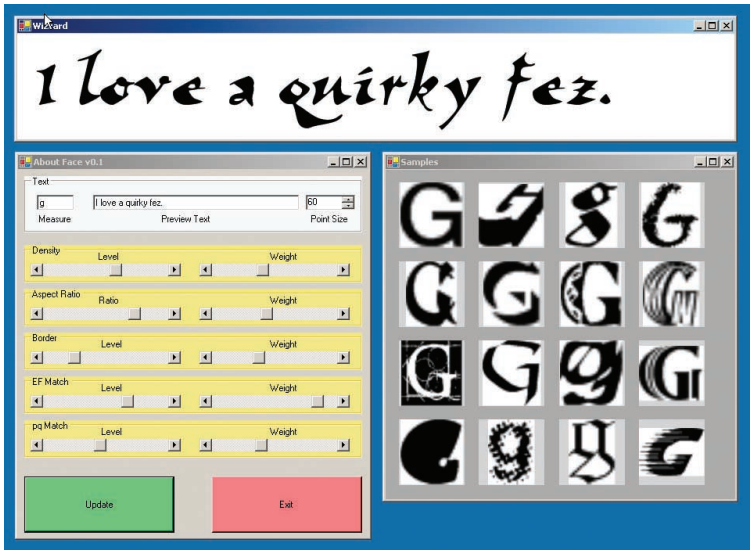
A nice general resource for typefaces is at http://jeff.cs.mcgill.ca/,luc/classify.html. You can find a nice collection of pangrams at http://rinkworks.com/words/pangrams.shtml.

1 Serifs for the letter F. The top row goes from a letter with clear serifs on the left to a sans-serif letter on the right. On the bottom row, you could argue about whether these have serifs at all. Top row fonts: Amery, Freeman Condensed, Antique, Loose Cruse, Castle, Architect, and Adams. Bottom row: Cowboy, Epic, Cocoa, Adorable, Freedom 9, Amaze, and Banner Light.

tle flare at the ends of lines, like the foot at the bottom of the letter F in Figure 1a). Reasonable people can hold different opinions on which of these letters have serifs and which don't, and there's no way a computer can reliably settle these disputes. Writing a program to reliably detect serifs in even the easiest cases still seems pretty hard to me.

Metrics

If deciding whether a character has serifs is hard, imagine how much tougher it would be to categorize a typeface as handwritten or brushstroke. Automating these types of quality distinctions is nearly impossible. Yet I wanted a procedural way to sort through my font collection. So rather than automatically assigning qual-



2 Screen shot of About Face in action. The control panel is in the lower left, the font browser is in the lower right, and the preview window is on top.

itative descriptions and labels, I decided to see how far I could get with simple numerical measures.

Of course, other people have looked into developing numerical measures, or *metrics*, for typefaces. The Panose system specifies about 65 measurements to help describe and distinguish a font.

Panose was designed for Latin typefaces only, so it's not applicable to typefaces for languages like Hebrew and Kanji. A Panose description of a Latin Text face is made up of 10 numbers, one each for the type of family, serif style, weight, proportion, contrast, stroke variation, arm style, letterform, midline, and X-height. To each of these categories the system assigns a number, drawn from a list of standardized values. For example, family type is set to 2 for Latin Text, and serif style is set to 5 if the serifs are of the form the Panose documentation calls "obtuse square cove". If you're looking for a particular typeface, you can figure out what its Panose code would be, and then use that code in a directory of fonts to see a sample and identify its name.

Another approach to identify a font is to use a tool that tries to work backward from one or more characters. The Identifont Web site leads you through a series of questions about the shape and style of different characters. Using a process of elimination, the site eventually identifies the typeface by name and foundry. If you're just browsing for something interesting, you can answer the questions according to your intuition or how you think your desired font would look, and see what it gives you.

If you have a sample in-hand, you can submit it to the What The Font?! site. You simply upload an image of a type sample you have, and it will look through its database to name the font for you.

Neither of these approaches is quite as nice as hunting through a wide variety of typefaces all at once and choosing among them, like flipping through the pages of a catalog. I wanted to try creating a system that would let me characterize my intentions in some general, high-level way and browse through the typefaces that match those goals, yet not require me to hand annotate every typeface in my collection with one or more descriptive labels.

I decided to cook up a bunch of different metrics that seemed easy to measure and had a chance of being meaningful, and then write a tool that let me browse through my collection according to how well they met **the** weighted collection of metrics.

About face

Figure 2 shows a screen shot of my program, About Face, in action. The interface all happens in three windows. I call the top window the *preview window*, the lower left the *control panel*, and the lower right the *font browser*.

Let's start with the control panel. The gray section at the top lets us pick characters to display in the font browser, and the text and point size of the sample shown in the preview window.

Below the gray section are six pairs of sliders. Each pair lets you specify the value for a particular metric, and how much that value should matter. For example, the top metric is density, which can take on a value from 0 to 1. To measure density, I typeset the character entered in the measure box (in the top left of the control panel), find its bounding box, and then compute the average density of the character. White is 0, black is 1, and gray values are intermediate. I add up the density of all the pixels and divide by the total number of pixels in the bounding box to get a measure for the density of that character.

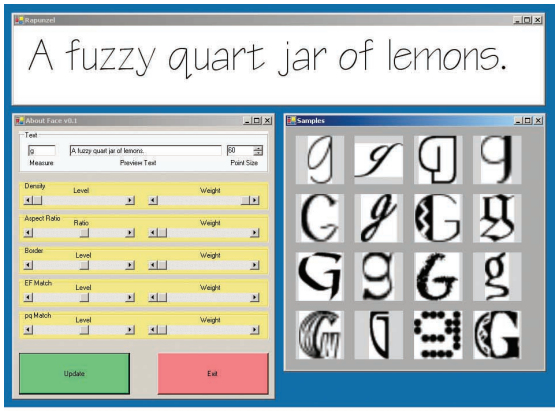
You can move the level slider from 0 at the left to 1 at the right, describing the density you'd like. Then you adjust the weight slider, again from 0 at the left to 1 at the right. This describes how much the density value should factor into the system's choice of font.

There's a pair of sliders for each of the other four metrics, which I'll describe later.

When you press the update button, the system goes through all the fonts in the program's directory, and scores the measure character of each font with regard to the metrics. The sorted results are shown in the font browser window in the bottom right of Figure 2. The best result is in the upper-left, and descending scores run left to right, top down.

You can click on any font in the font browser window, and the preview window will show you the preview text using that typeface. The name of the window changes to identify the face that's currently displayed.

It's fun to think about what constitutes good preview text. Obviously if you're setting something in particular, like a company name, you'll want to see that text. If you're going to use the typeface in a more general way, you want to get a feeling for its general appearance. One approach is to use a short fragment of text from the copy you're going to set. Another approach is to use a *pangram*, which is a sentence that contains all the letters of the alphabet. The best-known pangram is "The quick brown fox jumps over a lazy dog." Some people enjoy constructing ever-shorter pangrams. Perhaps the short-

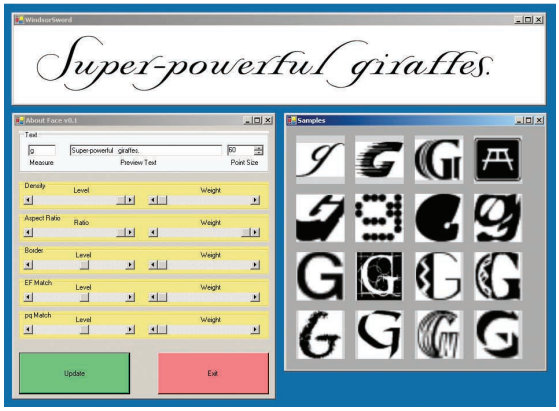


(a)

(b)

3 Looking at the effect of the density metric. The weights for all other metrics are set to 0. (a) Density = 0. (b) Density = 1.

(a)



(b)

4 Looking at the effect of the aspect ratio metric. The weights for all other metrics are set to 0. (a) Aspect ratio = 0. (b) Aspect ratio = 3.

est one that reads like a real sentence, and doesn't use odd abbreviations, is "Sphinx of black quartz, judge my vow." I tend to change the preview text several times as I'm considering a typeface, just to look at different combinations of letters and see how the text feels.

In Figure 3a I've set the desired density to 0, and cranked the weight for density all the way up and set all the other weights to zero. Thus the fonts are sorted so that the lightest (or whitest) is first, with increasingly darker faces following. In Figure 3b I moved the desired density to 1, and the darkest letters bubble to the top. If I want to sort on some other letter, I just enter it into the measure box and press the Update button.

Metrics

Finding good metrics is the key to making this approach work. I wanted metrics that I could easily compute, and wouldn't require any kind of shape analysis. I thought about many of the standard morphographic measures, but they didn't seem like they'd be of much value in finding fonts.

I eventually settled on a set of five measures. We've already covered density, so let's look at the others.

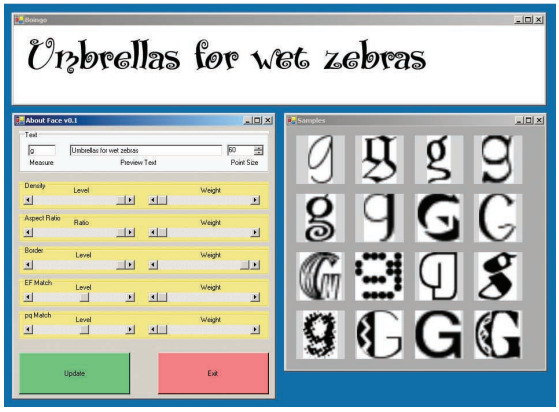
The aspect ratio is simply the value of dividing the width of the character's bounding box by its height. Figure 4 shows this value set to its minimum and maximum values, with the weights for the other metrics set to 0.

To compute **the border metric** I run through all the pixels in the character's bounding box and try to determine which are on the character's edge. I do this by thresholding the bitmap of the character to 0 and 1 (I put the breakpoint at 0.5), and look for black pixels that have at least one 8-connected white neighbor. The number of such pixels, divided by the total number of pixels in the bounding box, gives me an estimate for the length of that character's border. Figure 5 (next page) shows this value set to its minimum and maximum values, with the weights for the other metrics set to 0.

The last two metrics ignore the letter in the measure box and instead check two specific characters in the typeface. The first is the EF match. My thinking is that this tells me something about the regularity of a typeface. Traditional typefaces, plus many others that have a traditional feeling, tend to have a capital F that looks a lot like their capital E. Figure 6a shows a few examples of this. On the other hand, more irregular and free-form typefaces, and those drawn by hand, will have a much weaker match between these letters, as Figure 6b shows. To measure the similarity between these two letters, I typeset them, align their bounding boxes, and then count the number of pixels that are different in the two bitmaps. I divide this count by the number of pixels in the bounding box, and subtract this ratio from 1, resulting in a value from 0 to 1 telling me how much the E and F match one another.

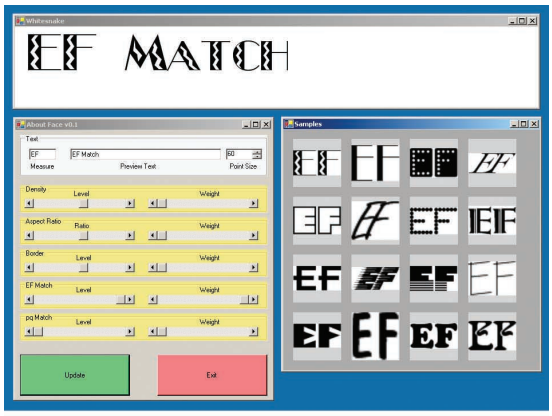
We've all heard the phrase "mind your p's and q's,"

(a)

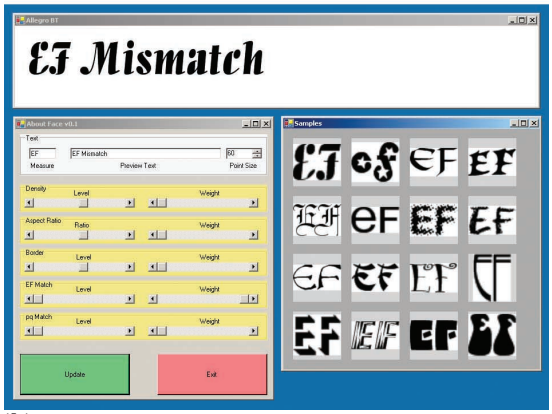


(b)

5 **Looking at the effect of the border metric. The weights for all other metrics are set to 0. (a) Border = 0. (b) Border = 1.**



(a)



(b)

6 **Looking at the effect of the EF-match metric. The weights for all other metrics are set to 0. (a) The EF match = 1 (this gives higher scores to fonts where the E and F are as dissimilar as possible). (b) The EF match = 0.**

which urges us to pay attention to details. This phrase started out as advice to typesetters, who set metal type in frames to create words. These metal letters were shaped backward so that the text would appear correctly when the letters were inked and pressed against paper. Because the lowercase letters p and q usually looked similar, it was easy to confuse them. (The same thing applied to the lowercase b and d, but I don't remember any specific advice regarding those.) I felt this aphorism was a good metric, like the EF-match metric, so that's what the last sliders control. I measure this just like the EF match, but I mirror reverse the q before computing the measure. Figure 7a shows some pq pairs that are good matches, and Figure 7b shows some poor matches.

Figure 8 shows the program in action. I played with the sliders for a bit, adjusting the values and the weights, and I followed my instincts to find the kind of letter forms I was after.

Efficiency

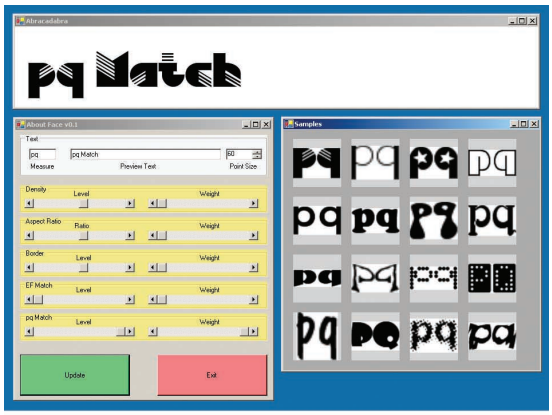
As I said earlier, I have thousands of typefaces on my hard disk. If I had to measure all of these characteristics each time I changed a value, the program would be too slow to use.

Instead, I precompute all of these metrics for each character in each typeface and save the results in a file (the EF- and pq-match metrics are saved only once per face, of course). When I enter a new character into the measure box, I read through the file and pull into memory the values for that character from all the faces. Then each time I move the sliders, it's a simple matter to compute the score.

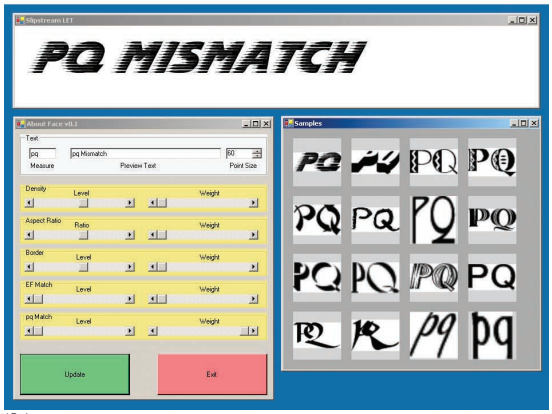
To compute a score, let's say that the value for slider i is v_i, and its weight is w_i. In my current system, that means density is v_0, aspect ratio is v_1, and so on. All of the values, and all of the weights, are between 0 and 1 (the only exception is the value of the desired aspect ratio, which ranges from 0 to 3). I score each font one at a time. I start by looking up the metrics for the character in the measure box. Let's call them p, so the density of that character is p_0, its aspect ratio is p_1, and so on. The highest-scoring character will be the one whose values exactly match the desired values on the sliders; that is, each $p_i - v_i = 0$. Of course, most of the time these values will be different, so we weight the absolute value of the difference by the corresponding weight for that metric, and then add everything up. In symbols, we compute the score S by

$$S = \sum_{i=0}^{5} w_i \, | \, v_i - p_i \, |$$

I could normalize by the sum of the weights, but there's no need to since they're the same for all the typefaces, and we only care about the relative scores, not their actual values.



(a)



(b)

7 Looking at the effect of the pq match metric. The weights for all other metrics are set to 0. (a) The pq match = 1 (this gives higher scores to fonts where the p and q are as dissimilar as possible). (b) The pq match = 0.

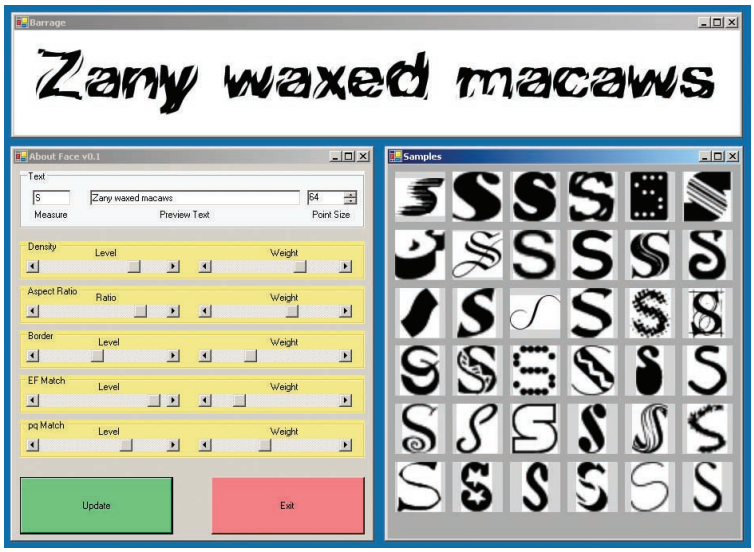
I wrote this program in C# using Microsoft's .NET development environment. This made it easy to create little bitmaps with typeset characters in them. I didn't have to learn how to read or parse the font files themselves, or figure out how to use them to render text. Instead, I just gave the system the font, the text to be set, a bitmap, and it did the rest.

Each time I press the Update button and create a new sorted list of typefaces, I read them into memory one by one, typeset the given character, scale it down if necessary, and then copy it into the browser window. I do this until the browser's filled. Then any time I get a mouse click in the browser, I find which box it's in, look up that typeface, and update the preview window accordingly.

It can take a few seconds to measure all the metrics for all the characters in a given font, including all of its variant faces (for example, bold, italic, condensed, and so on). Once that information is saved, it's fast to read in and use for updating. Working with several thousand fonts, I can get new sorted lists almost instantly for databases of a few hundred fonts.

Wrapping up

As I mentioned, the big trick here is figuring out the right set of metrics that make it possible to search through a huge number of fonts efficiently and pleasantly. I approached this system as a testbed: it's pretty easy to add new metrics, try them out, and toss them overboard if they don't measure up.



8 Snapshot of About Face in use.

I probably went through a few dozen metrics before settling on these. They seem pretty reliable at measuring what they're supposed to, and I find that often when I'm searching for some kind of typeface, these let me get something in the ball park. Unfortunately, with so many fonts on my computer, the ball park is huge. I show the top 36 candidates for each search in the browser window, but I often find myself wishing for more (I wanted to make sure that the characters in the browser window would be legible in this column, so I temporarily set the browser window to show a 4×4 grid of fonts rather than the 6×6 grid I use in practice).

I'd like to add a few things to this program. One is a Next button—so that, for example, I can see the second-best set of 36 candidates, or the third best, and so on. I'd also like to add a More-like-this button, so I can select a typeface and quickly get others that closely match. This can be on the basis of just the selected character in the measure box, or an overall score for the whole typeface.

On the whole, I'd say that the program is a qualified success and a good start. I can sometimes find good typefaces quickly, and I'm sometimes pleasantly surprised by what I discover. On the other hand, when I have something specific in mind, I've found that these metrics sometimes don't let me hone in on my preconceived ideas very closely.

More work on the metrics, and the user interface (which is admittedly crude and just for proof of concept) would definitely pay off in a more pleasant and robust tool for finding typefaces. ■

Acknowledgments

Thanks to Matt Conway, Greg Hitchcock, and Geraldine Wade for discussions and references while I worked on this article.

Contact Andrew Glassner at andrew@glassner.com.