Raph Levien

Raph Levien is a programmer with a special interest in graphics applied to type. He is also a typeface designer. He holds several patents and is a past maintainer of Ghostscript.

[Interview completed 16 May 2009.]

[Background of this interview: TUG, through the TEX development fund (`http://tug.org/tc/devfund`), supported Raph Levien's font work to a small extent. When it came time to document his work, Raph suggested an interview instead of a formal paper, which we were glad to oblige. Dave Crossland (`http://understandinglimited.com`), another type designer and colleague of Raph, shared in the conversation.]

Karl Berry, interviewer: Please tell us a bit about your personal history independent of typesetting and fonts.

Raph Levien, interviewee: I was born in Enkhuizen, the Netherlands, and moved to Virginia when I was three, so I don't really speak Dutch or anything but I do find myself with a liking for herring. I was something of a gifted child and, after a few somewhat disastrous years in public school, did a combination of homeschooling and taking classes at nearby colleges, mostly Virginia Tech and mostly math and physics. After a few years of trying to sell software on my own (with my dad helping on the business side), I decided that I really wanted to go to grad school, so I moved to Berkeley for that. I'm just now finishing my PhD (hopefully by the time this interview is published), and the topic is interactive curve design—motivated by my desire to have better tools to design fonts.

Between working at Google on spam prevention, finishing my thesis, and parenting two boys, I don't have a whole lot of time for hobbies but I do like photography.

KB: Did you get a bachelor's degree along the way? It would be quite unusual if you received a Ph.D. with no undergraduate degree

RL: Indeed, I took a bunch of undergraduate classes when I was younger but never actually got a degree. Berkeley has a great tradition of people with nontraditional backgrounds, and of taking some risks.

KB: I know you from the world of typography and fonts. How and when did you first get introduced to typesetting and font design?

RL: Among other things, my father made a number of books (he's especially well known for his miniature editions) and was very interested in fonts and typography. One of my earliest memories of books is an old Phil's Fonts catalog (an old New York phototypesetting vendor who I think mostly served the advertising market), which had a big letter on each page. So I sometimes say I learned the alphabet that way — A is for Aachen, B is for Baskerville, C is for Caslon.

In the mid-'80s, I wrote a batch typesetting system called Byso Print. It was in some ways fairly ambitious — among other things, it had autokerning based on the shapes of the individual glyphs. In other ways (especially compared to TeX) it was fairly crude. It did scaling of fonts by starting from a 256-pixel per em bitmap and scaling that. So, you can imagine, it looked pretty rough at large sizes. It also piggybacked on top of WordPerfect to do the hyphenation and justification.

My first serious attempt at a scalable font was a techno design called Zaltbommel (for a sample of this, and a bit more of my early history, see `http://typophile.com/node/54857`).

Ever since really diving into TeX, I've been fascinated and impressed with how advanced it was for its day. It's a shame its usability problems kept it from ever going mainstream, but it's good to see that it's carved out a solid niche for itself, and that there's still interesting development. Metafont in particular was way ahead of its time. I've only come to appreciate how good its interpolating spline algorithm is after going deep into the theory behind them for my own thesis. A good way of understanding it is as an approximation to the Euler spiral spline, basically as close as you can get to that using a cubic Bezier as the primitive segment. I think if it had been easier to use, it might have become popular. As it is, you have to be a hardcore computer scientist to understand Metafont programs — I'm even a little intimidated by them.

Metafont's use of cubic Beziers is very efficient, but I figured we have orders of magnitude more CPU power than then. What would the absolute *best* curve look like, if you had unlimited computing power? It turns out to be a trickier problem than I first thought, but I've come a long way to answering it in my thesis. Some of the optimization-based approaches, like Henry Moreton's Minimum Energy Curve and Minimum Variation Curve, took many seconds to solve a curve, but I've figured out some numerical techniques (strongly influenced by what John Hobby did for the curves in Metafont and MetaPost) that get you to interactive speeds without any problem. And, of course, the fact that you're drawing letters on the screen instead of writing a program makes the design process easier too.

I did a lot of tracing existing metal fonts using my curves. That was a good way to put my curves through their paces, but before long I started itching to do my own original design. I saw there was a niche to do a better monospaced font than what was already out there in the free world, and went for it.

Dave Crossland, interviewer: Metafont was one of the first graphic-program languages, and the way it is used has not changed much since the 1970s when it was invented. The success of Logo with children suggests that such languages need not be so intimidating, and perhaps different approaches to the way Metafont programs are written could help. Recently this set of languages has become an active area, with Processing, DrawBot, NodeBox, ShoeBot and others; perhaps the most advanced is Field, with its tight integration of a text editor to the graphic output of the code in that editor. There's a good video of this at `http://vimeo.com/3034647`. Do you have any suggestions for the way that Metafont could be made less intimidating?

RL: I'm not sure Metafont itself has much more life in it, sadly. While it was amazing technology for the time, its output can't be easily converted to standard font formats. That's pretty much a dealbreaker. (I know of various efforts, including rasterizing at high resolution and converting that to vectors, but that's not really acceptable for professional font design).

I think you could easily do Metafont-like things though, and the tools you listed are extremely promising. I think one of the most important things is to make it interactive rather than batch mode. In the '70s, batch mode made sense, but these days, if there's a coordinate, you really need to be able to click on the point and drag it.

I think one of the most exciting developments today is JavaScript in the browser, using either SVG or Canvas to do the drawing. Just in the past year or so, you have really good, fast JavaScript implementations, and a solid 2D rendering engine beneath. A couple years ago, it would be nearly impossible to get interactive performance. Today, by developing for the Open Web platform, you can make tools accessible to a lot of people, and having JavaScript right there means you could combine programmability with interactive graphics in interesting ways. I think John Resig's `processing.js` is a huge step in that direction.

KB: I'm intrigued by your passing mention of autokerning. Is autokerning implemented in any program today? I have never come across it. I've tried to write programs along those lines from time to time, for kerning and interletter spacing in general, but never succeeded.

RL: Yes, there are a few interesting autokerning systems out there. The first one that was really any good was the Kernus system by the Ikarus people. I think a version of that shipped with PageMaker 5 in the early '90s. Adobe then shipped a more refined version in InDesign, which is used by lots of people.

One of the more interesting things going on now in that space is iKern, by Igino Marino. I don't know too much about how the algorithm works, but I have tried it out on my Century Catalogue. The results are quite good, even better than InDesign in my opinion. Last I checked, he's not interested in releasing the tool itself as open source, but there are free font designers who are interested in using it. The quality of spacing is one of the huge determiners of quality between an amateurish font and a truly excellent one.

DC: Marino is apparently not interested in releasing it at all — he operates it as 'software as a postal service' where you email him a font and he emails it back with better spacing.

RL: Yeah. I hope that works out for him, but it pretty much limits the scope for his ideas to have broad impact.

KB: I'd like to hear a little more about these new curves. In reading your blog (`http://www.advogato.org/person/raph/diary.html`), I see that they're called Euler spirals or Cornu spirals. Can you say a bit about the difference a designer would see working with these spirals instead of Beziers? And would it make a visible difference to the ultimate reader?

RL: Well, Euler got there a hundred and thirty years before Cornu, so I think he deserves the credit for it. Actually, James Bernoulli first wrote down the equation for the curve in 1694, so it goes a long way back, but he didn't actually plot it and it's not clear he had a clear picture what it looked like. Since then it's been rediscovered any number of times in a number of different applications and, each time, given a new name. Fresnel found that their equations were useful for solving diffraction problems, so they're also known as the Fresnel integrals. Cornu figured out you could plot those and read the answer to the diffraction problem off the plot, so the curve got his name. It was also rediscovered a few times by railway engineers looking to make smooth transitions between straight sections of rail and curves, so the train doesn't suddenly lurch from side to side. In that domain, it's now most commonly called the clothoid.

And, in fact, I don't just use the Euler spirals, I use a mixture of curves (my package is called spiro, which is kind of an abbreviation for polynomial spirals). Most of Inconsolata (the monospaced font mentioned above) is drawn using G4-continuous splines, which are a very close approximation to the Minimum Variation Curve of Henry Moreton. I now think that's overkill, and G2-continuous splines (the Euler spiral ones) are plenty, and could be done with fewer points.

DC: Øyvind Kolås (`http://pippin.gimp.org`) mentioned to me that he came to the same conclusion — using G2 points as default and G4 points where extra smoothness is desired. And for me, FontForge allows one to switch quickly between Spiros and Beziers (although the conversion is not yet optimised); I find myself using Beziers by default, and then using Spiro splines when drawing curves that are tricky to keep smooth with Beziers, like S-bend curves — ones that double back on themselves — the vertical stroke of a '7', or the tail of a two-story 'g'.

RL: Interesting. I haven't actually tried working that way myself, and one of the things I worry about is making the interface too complex, but having more choices certainly fits into the free software aesthetic.

I've lately redone the optimized conversion to Beziers to be quite a bit faster (it used to take over a minute for most glyphs, now 15 seconds or so, and that's in Python). Maybe if that were recoded in C, it would be fast enough you could flip back and forth inside the drawing program and it would be fast enough not to interrupt the flow.

You can draw any curve you like using either Beziers or splines like I'm using. But I'm absolutely convinced that the tool you use has a profound effect on the shapes you draw. Almost all "contemporary" looking fonts you see produced these days have shapes that are very characteristic of Bezier curves. And, if you look at their outlines, you'll see that most often one quadrant of, say, an 'o' is drawn with a single cubic Bezier. So that gives you a palette. A reasonably rich one, but ultimately there are only two parameters describing a cubic Bezier that traverses exactly one quadrant of arc. You can go past it, but it takes more work, and it's easier to get lumpy results.

The curves you get using Spiro are a different palette. I think they're more classical, more like a French curve (and, in fact, I believe the Euler spiral is at least used as inspiration for French curves). So I think ultimately Inconsolata is a different font than it would have been had I drawn it using Beziers.

DC: Your Spiro package includes GTK and GTK2 prototype GUI programs for drawing with these splines, and since the release in 2007 they have been integrated into FontForge and Inkscape. Do you have any suggestions about what the free software community can do to make Spiro more popular?

RL: Part of the problem is that the Spiro integration has been on development branches, and it's fairly painful for most people to build from scratch, getting all the libraries, and so on. When you can just 'apt-get install' your apps that have Spiro turned on by default, I think lots more people will use those curves.

The other thing that I think might make a big difference is to build a good vector drawing editor for the Open Web platform. I have a prototype of Spiro for that too, but unfortunately it's too unfinished to actually draw in, and I don't have much time to work on it now. It would make a great project for someone who wanted to do something awesome in that platform, and get lots of users.

KB: Let's turn to that monospaced font you've designed that we mentioned above, named Inconsolata. Please tell us where the name came from, and about your design ideas for the font.

RL: My original idea for the name was "Unconsoled", which was intended to be both descriptive and self-deprecating as is common for free software projects (keep in mind that one of the projects I worked on was The Gimp). One idea for the font is that I was going to optimize it entirely for very high resolution rendering, and make no concessions to adapting it to a low-res pixel grid, which is of course the space that most monospaced fonts need to work in. I was very inspired by Consolas, a beautifully executed design, but I felt I could do some things better for the print domain, free to do some subtly angled strokes that just don't work when you're fitting to a grid. Ironically, most people use it as a terminal font anyway. On Mac OS X and (to a lesser extent) Linux, the rendering is pretty good, but on Windows it's fuzzy and users aren't as happy.

Hrant Papazian came up with the name "Inconsolata", which is more or less the Italian translation. I thought it sounded classier, so I went with it.

I wanted it to be a classical sans, clean like the Franklin Gothic series. I wanted to make the spacing as smooth as possible, so even though it is of course monospaced, it doesn't necessarily *look* like it is. That's one of the highest bits of praise I got — somebody saying that it had the color on the page of a proportionally spaced font. There are also quite a few glyphs with subtle curves in them, like the lowercase 't v w'. Some people don't like those, but I think they make text look warmer.

Since I was designing for very high resolution, I also wanted to play with an idea I got while looking at gothic fonts in Japan — the microserif, or very small spur. My idea was that it would make the font look a little sharper and crisper, and visually more interesting. In very small sizes, it's hardly visible at all (and of course not at all on the screen), but it's still cool knowing they're there.

KB: What's the current status of Inconsolata (I see it's available from `http://www.levien.com/type/myfonts/inconsolata.html`), and the Century Catalogue project you also mentioned, and any others you might have in the offing (post-thesis presumably)?

RL: Inconsolata is pretty close to what I'd consider done. It's shipping in a bunch of Linux distros, and there are quite a few people using it. There are a few more tweaks I want to do before calling it completely done, mostly responding to feedback I've gotten from users. In fact, a lot of that feedback has been in the form of people releasing their own versions with changes, which really feels to me like the free software spirit at work.

I did a lot of work tracing existing metal fonts, including Century Catalogue, to wrap my mind around the way the old masters worked (scans of these, mostly from the amazing American Type Founders catalogs, are available at my web page, and the raw high-res scans are on `tug.org`). Now, I think I've gained more confidence in doing my own designs. My latest work-in-progress is Cecco, for which I now have a complete lowercase and uppercase alphabet. It's intended to be a good working text font, and I'm pleased with the way it's turning out. With the amount of free time I have now, it'll take a while to finish, but that is one of the nice things about fonts — working slowly and steadily eventually gets the job done.

KB: I'm glad you consider Inconsolata at least pretty close to be being done, since we're using it for the typewriter material in this book.

Fonts in general have had a torturous legal history, going back to Goudy at least (as

I've heard it). Please say a few words about why you chose to release Inconsolata under the SIL Open Font License.

RL: It's very important to me that Inconsolata (and other fonts in the same vein) be useful for use in free software. I looked at the available licenses, and found problems with a lot of them. A lot of the licenses people use are for software, and aren't really appropriate for fonts. In particular, it's important to grant the right to embed the font into documents, and software licenses tend not to be clear about that — for GPL-licensed fonts, there's usually an explicit embedding exception. I also considered Creative Commons licenses (which are fairly popular among visual artists), but those may not be compatible with purist free software distributions.

To me, the exact legal terms are only part of the reason to choose a license. They're also important to signal intention and, to some extent, membership in a community. The OFL community tends to produce high quality fonts, like Victor Gaultney's Gentium and the excellent work of the Greek Font Society. I felt that choosing the OFL communicated my intent to do fonts better than most of the stuff that's out there for zero cost, but at the same time actually be useful for free software.

KB: Going along with font legalities, I noticed that you hold several patents, while licensing them for use in GPL'd software but not otherwise. Have you found that holding the patents is worth the bother of getting them, and what are your general thoughts on patenting of software and algorithms?

RL: Well, I'm very fortunate in this regard, because I have actually gotten some good revenues from licensing my patents over the years, both some early work I did on security (I was in fact eleven when I filed for my first patent) and more recently on halftone screening work. The screening work is used in the free Gutenprint inkjet drivers, and also commercially by several well-known companies. I think my strategy of using the free software release to bring more publicity to the work was a good one, and it would be great if the same thing happened with the curve work.

But the business of patent licensing is very hit or miss. Some of my better ideas never got licensed at all, and the ones that did were largely out of luck. The way the game is rigged, the best strategy for making money from patents is to be a patent troll, but that's never what I wanted to do — I just wanted to be able to create new technology and find a way to make some money.

Generally, I think the world would be better off if software and algorithms weren't patentable. The actual incentive to individual inventors — the main motivation you see cited — isn't very strong, and the potential for abuse is huge. It's also expensive and time consuming to get patents, so mostly you see big companies doing it. In general, I wouldn't recommend bothering.

KB: I also noticed you are or were one of the maintainers of Ghostscript, a crucial package in the free software world. How did that come about?

RL: I was one of the maintainers until a couple years ago, when I went over to Google — I unfortunately don't work on it any more. At the time I started on Ghostscript, around 2000, I was doing various 2d graphics projects in free software, including libart and the Gnome canvas, and Artifex was looking for people to work on projects for Ghostscript, both for release as free software and for their commercial licensing business, which continues to grow at a healthy clip. I started out with implementing PDF 1.4 transparency, and, after that worked out pretty well, joined Artifex as a full time employee. Ghostscript is, as you point out, a core component of the graphics and printing infrastructure in free software, and I'm still very friendly with them and wish them the best.

KB: I'll close with a general question. As we all see every day, there are seemingly an infinitude of fonts already in existence (many for centuries), some of insurmountable beauty, others of insurmountable ugliness. You've talked about the technical reasons you had for creating Inconsolata, but it seems to me that very few new fonts have such a background.

Why do you think fonts are still being created in such incredible proliferation? What is it that makes them such an attractive goal?

RL: Good question. Mostly, designing fonts is very satisfying creative work, I guess for certain people anyway. I enjoy drawing but can't do it very well, and I find that with fonts I can continually refine and improve the curves until it looks like what I wanted.

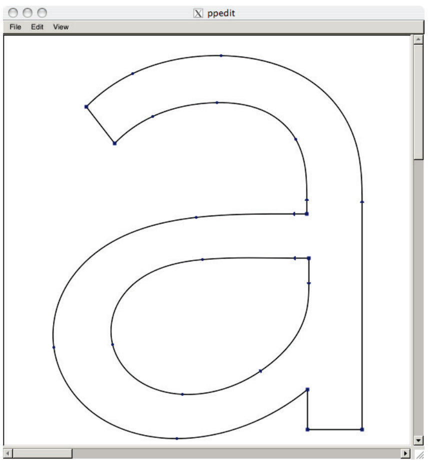
Also, fonts (especially good ones) tend to last a pretty long time, while most software gets thrown away quickly. It's especially satisfying to me to create something which might be more durable.

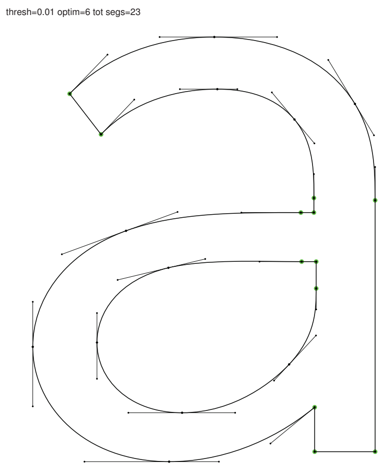
KB: Thank you very much for participating in this interview, and all your excellent work.

RL: My pleasure!

Endnote: Raph provided several figures showing the Inconsolata development process. He writes:

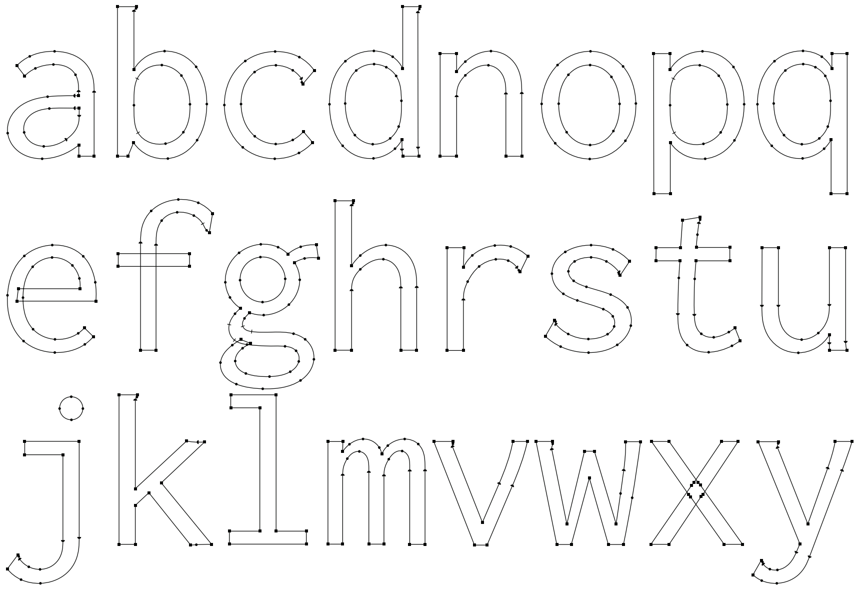
> The left-hand image shows the prototype GTK software I used for drawing all the Inconsolata glyphs — extremely minimal, but I found the clean and beautiful antialiased rendering really helped me focus on the shapes of the letters. The right-hand image is the result of converting that to optimized Beziers:



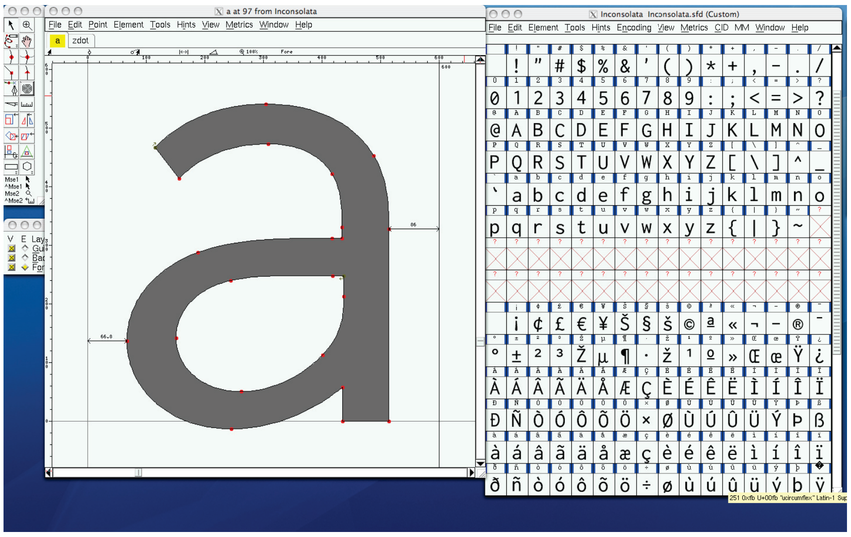

thresh=0.01 optim=6 tot segs=23

(continued)

Here are the Spiro sources for most of the lowercase:



Last, this is a screenshot of FontForge (`http://fontforge.sf.net`), which I used to put all the shapes together into a font, and produce Type 1 and OpenType font files.


a at 97 from Inconsolata
File Edit Point Element Tools Hints View Metrics Window Help
zdot
Inconsolata Inconsolata.sfd (Custom)
File Edit Element Tools Hints Encoding View Metrics CID MM Window Help
251 0xfb U+00fb "ucircumflex" Latin-1 Sup