



intertwingly

Search

It's just data

SVG TIDY

FRI 01 FEB 2008 AT 20:37

Normally, [I hand craft my images using vim](#). I also try to limit myself to 1K bytes. I occasionally find it convenient to start from an existing image. Sometimes that works out. Sometimes it does not.



[This image](#) nearly did not work out. In retrospect, I would have spent less time writing it from scratch, but some good did come of the effort. But first, the problem. Or rather problems.

The first problem can be illustrated by this excerpt:

```
<svg viewBox="0 0 128 102">
  <g>
    <g transform="matrix(1.8,0,0,1.8,-556.831,-272.2498)">
      <g transform="translate(-44.25663,-9.440477)">
        <g transform="translate(-225.2675,-211.7949)">
          <path transform="matrix(0.738729,0,0,0.738729,168.6073,57.12999)
            style="stroke:none;stroke-width:9.14700031"/>
        </g>
      </g>
    </g>
  </g>
</svg>
```

What this defines is the width of a stroke (typically the border of a shape) to nine digits of precision. And then specifies two scalings and four translations (movements about on the page) to be applied to the path. All for a stroke that is `none` i.e. not to be shown at all. I realize that the interior is filled, but why specify the stroke to so many digits of precision and have so many transformations defined?

But that's just number crunching. The second problem is that the icon is too busy and too detailed. It doesn't work well as an icon. In fact, if shown full screen and centered on a 1080p widescreen display, there are a number of curves so delicate that they would not deviate by so much as one pixel from a straight line. There also are a number of lines

that would show up as zero pixels in length. In fact, there even is one curve that also would not occupy any pixels at all.

I've not quite gotten [this](#) down to my self-imposed limit of one K bytes, but I am getting closer. In the meanwhile, this effort has motivated me to collect up a number of my ad-hoc scripts that do various tedious number crunching exercises and package them up as a [script](#). I even have a [spec](#) written using [bacon](#) as bacon is [known to work on Ruby 1.9](#).

While you should be able to use your choice of Rubies, you will need a recent version of REXML. Preferably one from [SVN](#).

Sat 02 Feb 2008

"NORMALLY, I HAND CRAFT MY IMAGES USING VIM"

“Normally, I hand craft my images using vim” - Sam Ruby...

Excerpt from [Application Error](#) at [12:31](#)