The PracTEX Journal, 2012, No. 1
Article revision 2012/06/25

Seeing Stars

Jim Hefferon

Many web sites let users rate content. On a site of mine, visitors can rate things on a scale of 0 to 5 stars and then the page shows the overall rating.

This is probably not an integer. (It is not quite an average of all user ratings, since averages are a problem when there are only a few votes, but it is close to an average.) For instance, I may need to display 2.5 stars out of 5.

Cascading Style Sheets can do this. The idea is that I write a box with a repeating background of stars in some muted color, with the width of the box set to exactly the width of five stars. Over that I draw repeating stars in a more distinctive color, with the width of this overlay box set to show the correct number of stars. Here is the HTML.

```
<span class="rating_bar"><span style="width:{{rating}}%"></span></span>
```

Muted stars are drawn in the `<span>` with class `rating_bar`. Distinctively-colored stars are drawn in the inside `<span>`. I use the Django web framework,[1] and to produce the above screenshot the `{{rating}}` is replaced by the number 50.0, leaving `style="width:50.0%"`.

Here is the CSS.

```
.rating_bar {
  display: inline-block;
  width: 100px;
  background: url(/graph/star-grey.png) 0 0 repeat-x;
}
.rating_bar span {
  display: inline-block;
  height: 20px;
  background: url(/graph/star-rust.png) 0 0 repeat-x;
}
```

1. www.djangoproject.org

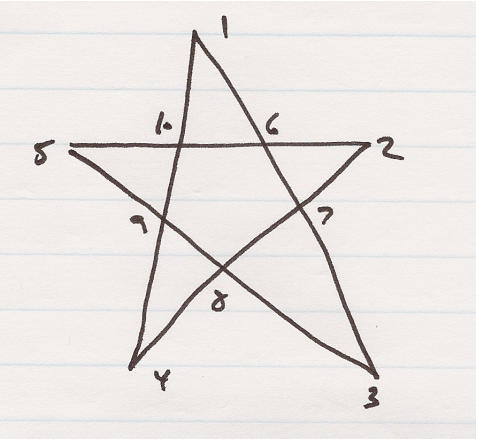
The stars are the graphics `/graph/star-grey.png` and `/graph/star-grey.png`. This is the problem. I cannot do art. I tried opening a drawing program and making a muted star for a half hour before giving up, no closer than when I began.

But there is a happy ending. METAPOST drew my stars.

A star is made

METAPOST is a programming language for drawing graphics, especially two-dimensional line art. It is closely related to METAFONT, the language in which TEX's original fonts were specified. METAPOST is not interactive, that is, it does not entail drawing with the mouse and menus (good thing, because I cannot do interactive). Instead, you make up an input file and run it through a compiler.

I started by sketching a star.



I told you that I cannot do art.

I started by labeling the points of interest. In the METAPOST source file below they are called `z1`, `z2`, etc.

The stars, like rust

Here is the input file `star.mp`. I'll go through part-by-part at the bottom.

```
% star.mp
%
%  2011-Oct-29 Jim Hefferon jhefferon@smcvt.edu  Written
outputtemplate := "%j-%c.mps";

numeric line_width_light;
line_width_light:=0.4pt; % TeX's rule width

color rust;
  rust=(171/255,50/255,41/255);

def draw_star(expr c) =
  save width, height;
  numeric width, height;
  width=20pt; height=width;
  z0=(.5width,.5height);   % center
  z1=(x0,height);    % top corner; 12 oclock
  z2=((z1-z0) rotated (360/5))+z0;  % mid right corner
  z3=((z1-z0) rotated (2*360/5))+z0;  % bot right corner
  z4=((z1-z0) rotated (3*360/5))+z0;  % bot left corner
  z5=((z1-z0) rotated (4*360/5))+z0;  % mid left corner
  z6=whatever[z1,z3]=whatever[z2,z5]; % mid right of internal pentagon
  z7=whatever[z1,z3]=whatever[z2,z4]; % bot right of internal pentagon
  z8=whatever[z2,z4]=whatever[z3,z5]; % bot of internal pentagon
  z9=whatever[z1,z4]=whatever[z3,z5]; % bot left of internal pentagon
  z10=whatever[z1,z4]=whatever[z2,z5]; % mid left of internal pentagon
  % Outline
  path p;
  p=z1--z6--z2--z7--z3--z8--z4--z9--z5--z10--cycle;
  fill p withcolor c;
  pickup pencircle scaled line_width_light;
  draw p;
  enddef;

beginfig(0) % muted star
  draw_star(.9[black,white]);
endfig;

beginfig(4) % exciting star
  draw_star(rust);
endfig;
end
```

A percent sign marks the rest of the line as a comment. The `outputtemplate` line determines the name of the files that METAPOST outputs. Here, `stars.mp` will output two files, named `stars-0.mps` and `stars-4.mps` because further down in the file there are sections for `beginfig(0)` and `beginfig(4)` (I have other colors in my file but I've cut them out for this discussion).

Next I give a width for lines, and I define a color. One of the colors for my site I named *rust*. Its RGB specification is $(171, 50, 41)$. It is what I used for the

distinctively-colored stars in the screenshot from the start of this article.

Then comes the main work. As I described above, I will draw two same-sized stars in different colors, so I defined a single section of code to draw a star and I'll call that code once with a gray color and once with rust.

That `draw_star` code starts by defining the five points `z1` through `z5`, by making the first at the top and then rotating for the other four. No artistic eye required.

Making the interior points `z6` through `z10` is where we get a peek at METAPOST's powers. It will find intersections. For instance, to find `z6` it intersects the line from `z1` to `z3` with the line from `z2` to `z5`. (Because METAPOST is finding the intersection, I don't have to. This is particularly useful when I fiddle with a drawing somewhat, trying to get it right. I don't have to recompute the intersection each time with METAPOST.)

Here is the construct to intersect the lines.

```
z6=whatever[z1,z3]=whatever[z2,z5];
```

The square brackets find "of-the-way" points. Thus `.25[z1,z3]` is the point one quarter of the way from `z1` to `z3`. To intersect the lines we want a point that is an unknown part of the way from `z1` to `z3`, and also is an unknown part of the way from `z2` to `z5`. METAPOST's `whatever` is a throwaway variable, so with the above code the system knows that `z6` is on both lines. There is only one such point and METAPOST is smart enough to find it.

At the end of the definition, I give the star's outline path `p`, fill it with the color, set the line width with the `pickup` command, and draw the path around the now colored-in star.

To output the files with the star graphics I now run the two routines with different colors. I had to decide how gray to make the muted star. After a couple of tries I decided that a good choice was the color 0.9 of the way from black to white.

Done.

When you wish you had a star

To get the graphic files, I followed these steps.[2] I first ran METAPOST.

```
$ mpost stars
```

I next made a proof sheet to see how it looked.

```
$ tex mproof stars-0.mps stars-4.mps
$ dvips -Ppdf -omproof.ps mproof
```

Now I can view the output with Ghostview.[3]

```
$ gv mproof.ps
```

Browsers want the graphic in PNG format so I used the swiss-army knife graphics conversion program.[4]

```
$ convert star-4.mps star-rust.png
```

I copied the two files to the directory where my web server could find them.

My stars!

Here are the completed images, post processing.[5]

Now they are the right colors, they *look* like stars, and they are in a format that browsers can understand. Success.

2. I run Ubuntu Linux with TEX Live. Your setup may require variations on these steps.
3. See http://pages.cs.wisc.edu/~ghost/.
4. See http://www.imagemagick.org/script/index.php.
5. The graphics conversion program made each graphics 20 pixels wide. I believe this is because I defined them to be 20 pt wide in the METAPOST file and a point is close to a PostScript big point.