Computer-Generated Solar Halos and Sun Dogs

In the January column, I described Greenler's technique for creating dot patterns that approximate solar phenomena such as sun dogs and solar halos. I closed by promising to extend the simulation to include energy considerations, smooth images, and color.

Here's a summary of the basic process for creating the dot patterns (for details and figures refer to the January column). We start by imagining a perfectly hexagonal ice crystal, which may be thin compared with its radius (in which case we call it a plate) or long and skinny (a pencil). To create a solar display, we first find an orientation for the crystal. If there's not much wind, each crystal will tend to fall with one of its biggest faces parallel to the ground. Wind can cause the crystals to tumble as they fall, which we simulate by adding some amount of random rotation.

Once we have rotated the crystal into the chosen orientation, we send a ray from the infinitely distant sun toward the crystal. If the ray hits the crystal, we follow its refractive path through the ice and find its outgoing direction as it leaves the crystal. To see that light, we must be looking in its direction back at the crystal, so we add some light from that direction into our image. I simply quantize the direction to the nearest pixel and color in that pixel. Then we pick a new orientation and trace a new ray, repeating the process over and over. The result is a cloud of dots. By following different paths through the crystals, constraining the possible orientations, and changing the crystal's length, we can simulate a wide variety of solar phenomena.

Energized by old friends

The first thing we'll do to enhance this model is include energy effects, which we will handle very casually. There are lots of places where the light ray can pick up and lose energy, but we'll focus just on the crystal faces. When the ray strikes the first face, it represents a whole beam of light. Thus we need to apply Lambert's Law, diminishing the energy by the cosine of the angle the ray makes with the face normal; this is our old friend from polygonal shading. Similarly, we should apply Fresnel's Law, which accounts for the fact that glancing rays are generally reflected, while rays more per
shown in Figure 4 (left). My favorite formula for approximating $\eta$ as a function of wavelength $\lambda$ is Sellmeier’s formula, which in two-term form is $\eta(\lambda) = A + B/\lambda^2$ (you can add more terms for more accuracy, but I found that the two-term form fit the data with excellent precision). Since the table gave values at different temperatures, I made the coefficients in Sellmeier’s formula temperature-dependent:

$$\eta(\lambda, T) = A(T) + B(T)/\lambda^2$$

I used a symbolic algebra program to compute $A$ and $B$ at each of the 11 temperatures, so now I had 11 samples of the functions $A(T)$ and $B(T)$. I found that quadratic polynomials matched both functions very well; these polynomials are

$$A(T) = 1.32491 - 0.0000039278T - 0.0000012678T^2$$
$$B(T) = 3105.31 + 1.2552037 - 0.0355608T^2$$

where $T$ is the temperature in degrees Celsius.

So now to find $\eta$ at any wavelength and temperature, I compute $A(T)$ and $B(T)$, and plug that into Sellmeier’s formula along with the wavelength in nanometers. The maximum absolute and relative errors in this approximation are both less than 5 parts in 10,000. The relative error is shown in Figure 4 (center), and the values generated by the approximation are plotted in Figure 4 (right). It’s winter now in Seattle, so I decided to send my simulations on vacation. All of the figures in this month’s column were made at a balmy 30 degrees Celsius.

3 A close-up of the overlaid upper-left corner of the 400- and 700-nm simulations in Figure 2. Note that the red dots appear inside the blue ones, and the blue dots predominate at the outside.

4 Reference data for the index of refraction of ice (left), relative error in the approximation used here (center), and sampling of the approximation function (right).
Scanned image, scan results

I first tried to combine the color runs by simply adding the images together. I hoped that the dense regions on all plots would sum to white and that smooth regions would emerge where the different colors faded out. Unfortunately, the result looked just like Figure 1—speckles. I realized that the density plots had to be smoothed before they could be combined.

My impulse was to abandon the dot patterns and generate a smooth picture directly by inverting the image-making process. Rather than orienting the crystal and then tracing rays, I would scan the screen and find the probability of a crystal being in the right orientation to send light through each pixel in the image. This sounded like a simple enough problem: the crystal is rigid, the geometry of refraction is well known, and everything should be straightforward. Indeed, it is straightforward to a point, but things get very messy very fast.

Just setting up the equations is complicated because of all the trig functions involved. Then I realized that in fact there was no single solution. Lots of different crystal orientations can send rays back in any given direction. Just as a quick test I selected one pixel and ran the simulator. Dozens of different sets of angles sent light back in almost the same direction, passing through the same pixel. I decided to stick with the dot patterns and try to smooth them out.

Blurs and blobs

Looking at the dot pattern of Figure 5a, my first thought was to blur the density plot. But no value of blur worked well. The small blur of Figure 5b didn’t get the dots to join up and form a smooth field, and the large blur of Figure 5c made the whole picture go fuzzy. Next, I drew a Gaussian blob over each spot, like the splatting technique used in volume visualization. I used the distance from each ray to its nearest neighbor to determine the blob’s radius, and scaled the blob’s height by the amount of energy carried by the ray. The radius was set so the blob was at half-height at its nearest neighbor.

Finding the nearest neighbor involved saving all the ray locations (at subpixel accuracy) and then searching for neighbors for each ray. To speed the search, I built a data structure of overlapping rectangles in the image; then for each ray I searched only the other rays in its rectangle. Because the rectangles overlapped, a ray could belong to more than one such rectangle; so I avoided the problem where two nearby rays straddle a boundary between rectangles and don’t see each other. When the image was built up, I normalized the pixel values to use the whole display range.

Figure 5d shows the result of the blobs. The bright spots come from places where lots of rays just happened to land on top of one another (this artifact isn’t visible in Figure 5a because all of these rays land on the same pixel). This result was disappointing not only because it looked so bad, but because it required a whole lot of additional computation and storage. I thought maybe if I replaced the blobs by flat disks it would look a bit better; but as you can see from Figure 5e, this wasn’t a success either.

Fuzzy thinking

The bridge to a better answer came from looking at the blurry picture and the sharp picture together. I thought that if I could get the blurry picture to show up where the sharp picture was black, I’d get a smooth field. So I turned the blurry picture into its own matte and composited it with the sharp picture. Where the blurry picture was bright, it dominated the result. As you can see in Figure 5f, this method smoothed out the inner
At first I simply added the three blurriest pictures together equally, as in Figure 7a. That still showed a little too much of the speckling for my taste. So I tried again, this time changing the recipe to use three parts of the 50-pixel image, two parts of the 100-pixel image, and one part of the 200-pixel image. Figure 7b shows the result. The inner edge is still relatively sharp, and the outer edges fade out nicely.

**Nothing new under the sun**

As soon as I made these images, I recognized this process as multiresolution compositing, which is a standard technique in image processing and computer graphics production. I wished I had thought of it before trying all the other approaches, but I did enjoy playing around with this problem looking for a good answer. This approach also has a big efficiency advantage over the Gaussian blob-type solutions, because it’s simple image processing and doesn’t require lots of additional data structures and processing. Just draw the dot pictures, blur them out, and add them up.

I applied the multiresolution blur and compositing to each of the dot patterns in Figure 2, and generated the composite 22-degree halo shown in Figure 8a. It’s a pretty good match to the photographs shown in last month’s column. Figure 8b shows a close-up of the inner edge; it has a satisfyingly red tint. I ran through the same process for sun dogs, as shown in Figure 8c. Note that the sun dogs aren’t just slices of the halo; their color fringes have a subtly different shape. The upper and lower tangent arcs for a sun elevation of 30 degrees are shown in Figure 8d.

**The movie**

I made a movie of the upper and lower tangent arcs for the rising sun from 0 to 90 degrees. You can find it on my Web page at http://www.research.microsoft.com/research/graphics/glassner/ or on the CG&A Web page at http://www.computer.org/. I will also maintain a list of notes and errata for these columns on those pages.

There’s a lot more going on up in the sky than I’ve talked about in these two columns. I hope to return to the topic again sometime. One thing we haven’t addressed is what happens to light that reflects off the crystals, rather than passing through them. These reflections give rise to a phenomenon called sun pillars. I encourage you to write a little simulation program and investigate them yourself.
Further reading

The basic technique for creating the dot patterns in this pair of columns was developed by Robert Greenler and his colleagues and is described in his book, Rainbows, Halos, and Glories (Cambridge University Press, 1980).

If you want to learn more about the colors in the skies, you can look at Minnaert's classic book, *Light and Color in the Outdoors* (Springer-Verlag, 1937, revised in 1985). A more recent volume with lots of good information is *Sunsets, Twilight, and Evening Skies* by Aden and Marjorie Meinert (Cambridge University Press, 1983).


8 (a) Multi-resolution reconstruction of the 22-degree halo.
(b) A close-up of (a), showing the red inner and blue outer bands.
(c) Multi-resolution reconstruction of sun dogs.
(d) Multi-resolution reconstruction of the upper and lower tangent arcs for a sun elevation of 30 degrees.