Andrew Glassner's Notebook

http://www.glassner.com

Soap Bubbles: Part 2

Andrew Glassner

Soap bubbles are fragile, beautiful phenomena. They're fun to make and play with, and their geometry is as clean and elegant as anything in nature, which makes them particularly suited to computer graphics.

In the last issue I discussed the nature of soap films. These thin sheets of soapy water have less surface tension than water itself, so they can billow out into curved surfaces like bubbles. In this column I'll talk about where a bubble's beautiful colors come from, the geometry of bubble clusters, and how to make bubbles in a 3D modeler. I'll use some of the material discussed in previous columns: reflection, refraction, and Snell's Law (January/February 1998 and March/April 1998) and complex numbers (January/February 1999).

Making waves

The easiest way to talk about bubble colors is to think of light as a wave phenomenon. As you may recall, light sometimes appears to behave as a wave and sometimes as a particle.

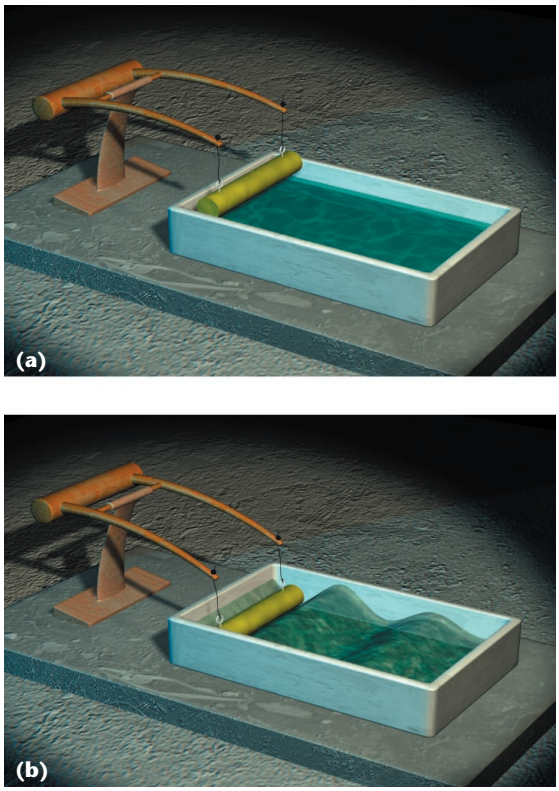
This wave/particle duality has been a source of much philosophizing in physics. In computer graphics, we typically take a pragmatic course and treat light exclusively as a collection of particles. We do this because the particle (or photon) interpretation has simpler and more intuitive mathematics and physics, and makes our programs easier to write and faster to run. Although some wave-based rendering algorithms exist, most fall firmly in the particle camp. The particle interpretation is usually a good choice because it covers almost all of the normal phenomena associated with light.

Almost all. One of the things that the particle model doesn't explain very well is *interference*—the source of soap bubble colors, often called *interference colors*. To set the stage for discussing interference, I'll use a classic piece of physics-lab gear called the *ripple tank* (or, if you have a big one, a *wave tank*). Figure 1 shows the idea: a basin filled about halfway with water. Then you plunge a heavy, long object into the water at one end, causing a ripple. In Figure 1, the object is a cylindrical pipe, hanging from the end of a rotating arm.

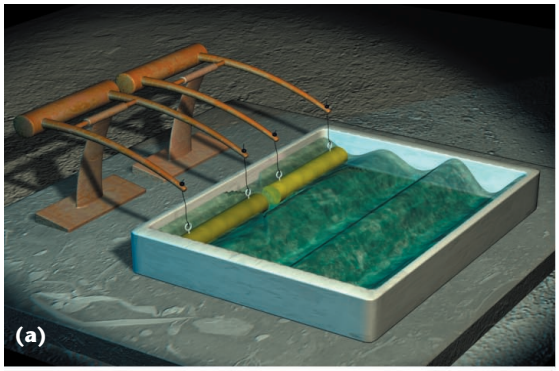
If you lower the pipe into the water once and pull it right back, you'll get a ripple spreading out from the point of impact. Of course, in the real world the ripple will bounce off the sides and far end of the basin and come back. Although those reflections are interesting, they introduce a lot of complexity without illuminating the points I want to make, so I'll just ignore them.

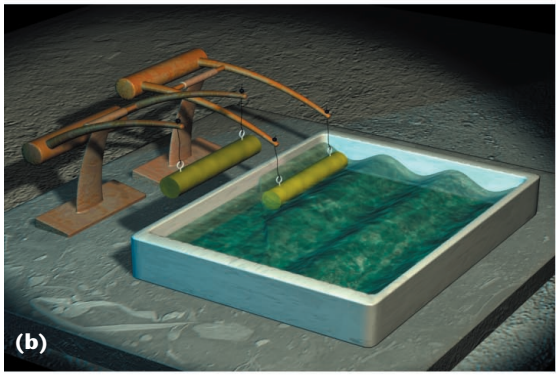
To get an endless succession of waves in the tank, plunge the cylinder in repeatedly at just the right rate. Each time the pipe drops into the water, it creates a wave, which starts to spread outward. If you measure the distance from the crest of one wave to the crest of the next one—as they move down the length of the tank—you'll find that that distance doesn't change as the wave moves out. This distance is called the *wavelength*. As we watch the wave move, it's like a disturbance moving through the water.

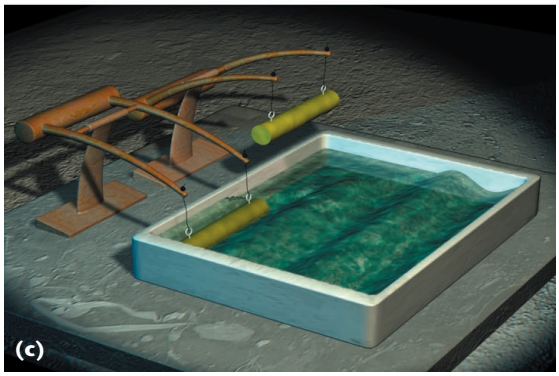
If we set a piece of foam on top of the water as the wave goes by, we'll see the foam rise up and down. If only one wave goes by, the foam starts at the normal water level, rises, drops, and then returns to its starting height. We call any particular point along the cycle of one wave its *phase*. The number of times the foam bobs up and down in a given interval of time is called the *frequency* of the wave,



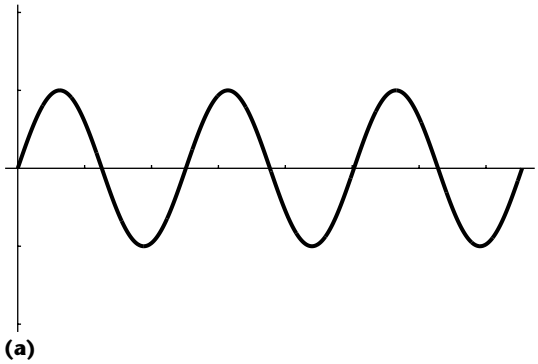
1 (a) A ripple tank. (b) The block has just plunged into the water, creating a wave.

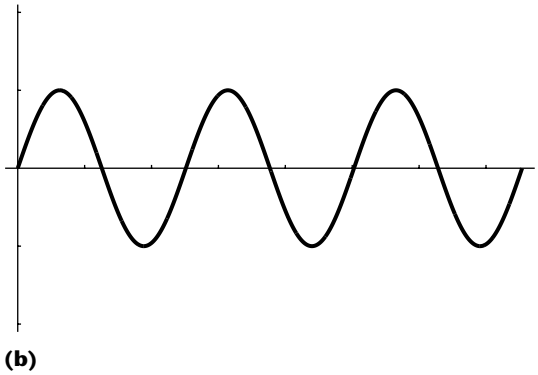


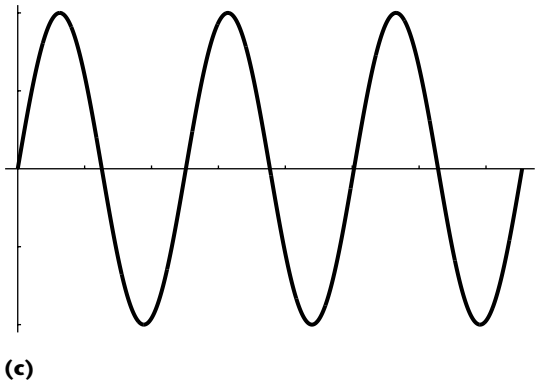




2 (a) Two ripple makers side by side, working in unison. (b) The wave due to one machine. (c) The wave due to the other.







3 (a) and (b) Two equally high waves in phase. (c) They constructively interfere to create a new wave with twice the height.

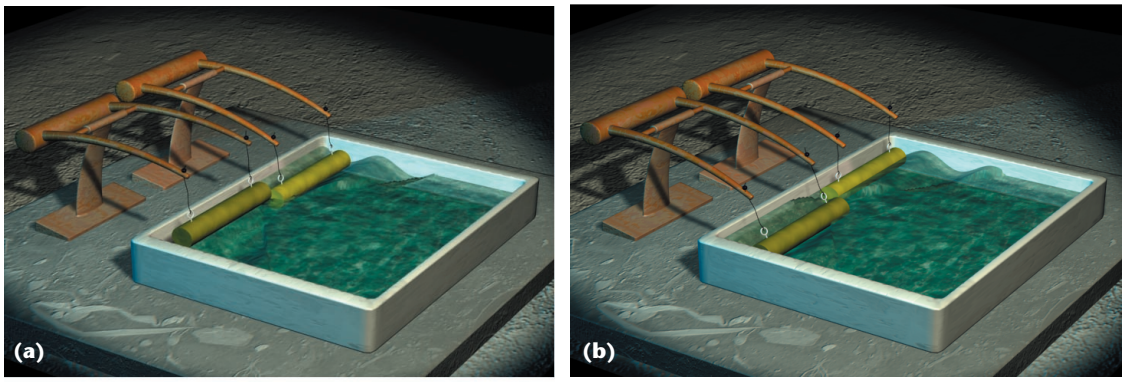
since it describes how frequently the water rises and falls. There's a simple mathematical relationship between the frequency and the wavelength, so both words are basically describing the same wave characteristic.

To make a never-ending series of equally spaced waves, we just need to drop the cylinder back into the water at the end of the previous wave. That is, just as one full wave has spread out from the cylinder, we inject a new one.

Let's now create two side-by-side wave machines, as in Figure 2. In Figure 2a, I've got both of them running together, as though they were connected as a single bar. Suppose that each one creates a wave that is 1-inch high. The wave from each plunger spreads out into the tank, and the two eventually overlap. We can see from the figure that in the overlap region, the wave looks taller than the wave due to either source. Why is that?

Figure 3a shows the idea. When two waves are in the same place at the same time, it turns out that they add up—the composite wave is just the sum of the two components. This property of the real world is called *superposition*. In Figure 3, the two waves are in sync with each other, so the high point of one wave coincides with the high point of the other. This is called *constructive interference*; we also say that the waves are *in phase* or *reinforcing* each other. The result looks like a single wave with the same wavelength, but twice the height.

Now let's get them going in exactly opposing motion. When one plunger goes up, the other goes down. Figure 4a shows the wave tank—the region where the two waves overlap is flat. Figure 5 shows a schematic. What's happening is that the two waves exactly cancel each other out. We say that these two waves are undergoing *destructive interference*, or that they're *out of phase*, or



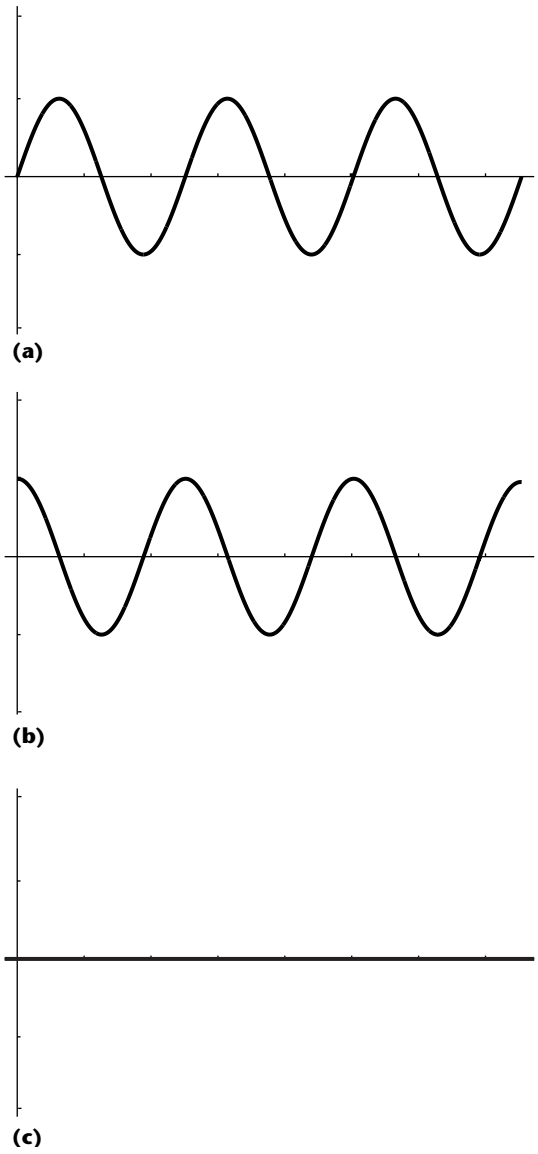
4 (a) The two ripple makers out of synchrony. (b) Equivalently, they're in synchrony but the bars are shifted by a half wavelength.

canceling each other. Figure 4b shows another way to achieve this, where the two machines go up and down in unison, but the wave-making pipes have been displaced by a half wavelength relative to each other.

When two waves of the same wavelength aren't exactly in phase or out of phase, the result is somewhere between perfect reinforcement and perfect cancellation. The rule of thumb is that the smaller the *phase difference* between the waves, the larger the resulting wave.

The reason for looking at these ripple tanks is because they give us an analogy for light waves. Light can be considered a form of radiation, traveling through the air just like water waves travel through water. Our eyes are sensitive to the wavelengths from 380 to 780 nanometers (nm), corresponding to the spectrum from red to violet. Higher amplitudes (or energy) in the wave correspond to brighter perceived light. When two light waves travel in the same location and direction and are in phase, they reinforce each other just like the water waves. If they're out of phase, they cancel each other out. Since light waves are represented mathematically as complex sine waves, we speak of the phase of a wave as a number of radians from the start (a full sine wave contains 2π radians, equivalent to 360 degrees).

Normally these interference effects are all but invisible in our daily world, but they do show up in special cases. One of those cases, of course, is at the surface of a soap bubble. Let's see how that happens.

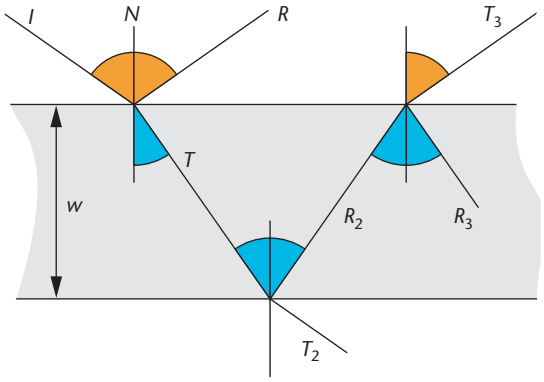


5 (a) and (b) Two equally high waves out of phase. (c) They destructively interfere and cancel each other out.

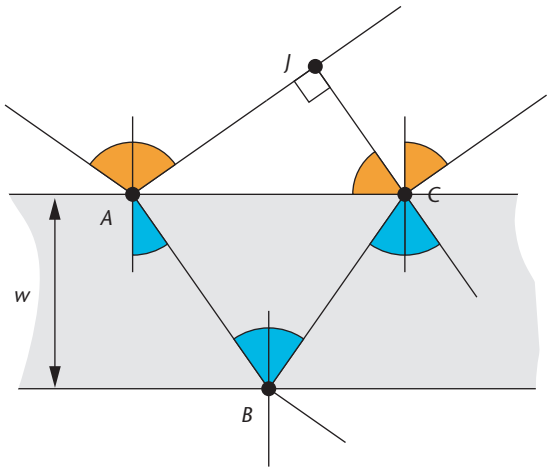
Interference colors

Consider a little piece of soap film, as in Figure 6 (next page). A beam of light, from direction I, strikes the upper surface. Some of that light is reflected in the direction R, while the rest is transmitted into the film in direction T. On the bottom of the film, some light passes into the inside of the bubble in direction T_2, while the rest is bounced back upward in direction R_2. This light hits the top of the film, where some is reflected back into the film in direction R_3, while the rest comes back into the air in direction T_3. In the January/February and March/April 1998 issues, I talked about the geometry of reflection and refraction, so here I'll just summarize the important result: T_3 is parallel to direction R. So the light traveling on R is going to interact with the light on T_3. Like the waves in the ripple tank, the resulting combined light will depend on the wavelength and phase of each constituent wave.

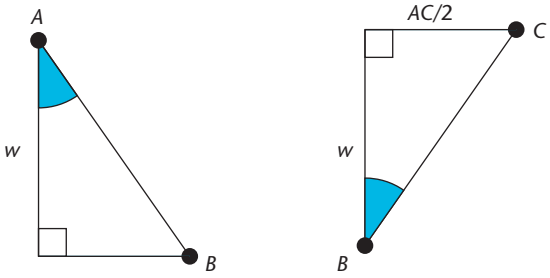
Let's assume that the light coming in along I has a wavelength of λ (measured in nanometers). The film has thickness w, and the index of refraction of the film is η (about 1.4 for soap films). Finally, the incident angle between I and the surface normal N is θ_i. I've identified a few points in the figure. The light that travels through



6 Light at a thin film arrives along direction *I*. After several interactions, light leaving in direction T_3 is parallel to light leaving in direction *R*, so the two waves interfere. All angles equal to the incident angle θ_i are gold. All angles equal to the transmitted angle θ_t are blue.



7 Some labels for Figure 6.



(a) **(b)**

8 Geometry associated with Figure 7. The blue wedges are angle θ_t. (a) cos $\theta_t = w/AB$ and (b) tan $\theta_t = (AC/2)/w$.

the film and exits parallel to *R* will be *phase-shifted* with respect to the light that simply reflected off the surface. As we saw earlier, the phase difference between these two waves will determine whether they reinforce or cancel each other, entirely or partly. Thus our goal is to determine the phase difference between the light on *R* and the light on T_3. This has two components: first, the extra distance traveled on the longer path, and second, an optical phase shift at the interface. Let's look at these in order.

We want to find the total difference in distance taken by the two beams. If we divide that by the wavelength, it will tell us how much of a cycle the light will pass through along this journey. In Figure 7 I marked an additional point *J*. Now we can talk of two side-by-side beams of light—one travels from *A* to *J*, the other from *A* to *B* to *C*. The geometric difference in distance is $AB + BC - AJ$ (as usual, I'll write *AB* for the distance from *A* to *B*). If we find this difference and convert it into the number of radians it represents for that wavelength, we'll have the phase difference, and thus the ultimate brightness of the combined light coming from the film.

You may recall that light travels more slowly in a medium that's denser than air, such as soapy water. Because it's going more slowly, it passes through more of its cycle when going through the water than when passing through an equivalent distance of air. We model that effect by multiplying the traveled distance by the index of refraction, which scales up the length to the equivalent amount of distance the wave would have covered in air in the same amount of time.

This part of the phase difference is thus $d = \eta(AB + BC) - AJ$. A little trig lets us figure out the value for *d*. Referring to Figure 8a, we can see that $\cos\theta_t = w/AB$, and that $AB = BC$. That takes care of the first term, so let's find *AJ*. Since angle $\angle ACJ = \theta_i$, we see that $AJ = AC \sin\theta_i$. We'll use Snell's Law to replace $\sin\theta_i$ with its equivalent $\eta \sin\theta_t$. We also see from Figure 8b that $\tan\theta_t = (AC/2)/w$, so $AC = 2w \tan\theta_t$. Plugging this value for *AC* into the expression for *AJ* we just got and putting it all together, we find

$$d = \frac{2\eta w}{\cos\theta_t} - 2w \tan\theta t\, \eta \sin\theta t$$

To my eyes, that looks pretty complicated. We can simplify it quite a bit with a few steps of rearranging and substituting:

$$d = 2w\eta\left(\frac{1}{\cos\theta_t} - \frac{\sin^2\theta_t}{\cos\theta_t}\right)$$
$$= \frac{2w\eta}{\cos\theta_t}\left(1 - \sin^2\theta_t\right) = \frac{2w\eta}{\cos\theta_t}\cos^2\theta_t$$
$$= 2w\eta\cos\theta_t$$

This is much better. Now we've taken care of the first part of the phase difference.

The second part is the optical phase shift I mentioned above. It turns out that when light moves from one medium to a medium of higher refractive index (for example, from air into soapy water), it undergoes a phase shift equal to half its wavelength (interestingly, this doesn't happen on the way back out). So we need to add $\lambda/2$ to the difference calculated above (since the light goes from air to soap at point *A*), giving us the *effective optical path length* (or EOPL):

$$\text{EOPL} = d + \lambda/2 = 2w\eta\cos\theta_t + \lambda/2$$

Whew! After all that work, the payoff had better be pretty good. Happily, it is—this last equation is the key to understanding interference color. It makes up the heart of the soap-film shader presented later in the column.

Because it's so important, let's discuss this result for a moment. For any given thickness w and wavelength λ, we can calculate the EOPL from A to C —remember that this is the effective extra distance traveled by the light that goes through the film compared to the light that bounces off the top. If this distance is exactly equal to some multiple of the wavelength, then the resulting two beams will be in phase and interfere constructively. Visually, this results in light that combines the energy in the two beams. If the distance is exactly equal to half a wavelength, then the two waves will be exactly opposite in phase, and will cancel each other out. Visually, we'd see blackness, or the absence of light. Since the color of light is a function of its wavelength, we would expect that for different values of w we would see one color looking especially bright, others less so, and some colors completely missing.

That's exactly what happens in soap films. If there were no extra distance to be covered, the $\lambda/2$ term would shift one wave by exactly a half wavelength with respect to the other, causing them to cancel each other out. If that extra distance is an exact multiple of the wavelength, this condition still holds. On the other hand, if the extra distance is an exact multiple of the wavelength plus a half wavelength, then we have perfect constructive interference, or reinforcement.

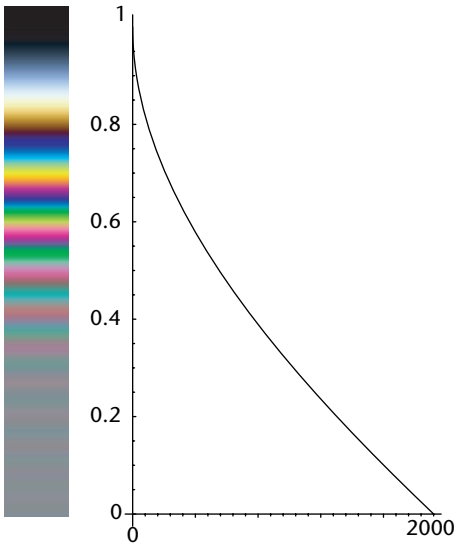
Suppose we illuminate a soap bubble with white light just above our heads (this is light that contains all wavelengths). After bouncing off the bubble, some wavelengths will be reinforced, others will be canceled out, and most will be attenuated to some degree. The final color will be the sum of all the noncanceled wavelengths of light.

If we create a vertical, flat soap film, then the effect of gravity is to pull down the water in the film, making a wedge that's thicker at the bottom than the top. The result is a sequence of horizontal, colored bands as the thickness gradually increases along the height of the film, shown in Figure 9.

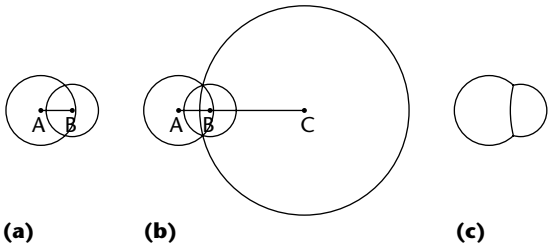
You've probably wondered about the light in Figure 6 that follows path R_3. Doesn't it then bounce off the inner surface and then return to the top of the bubble, where some light will escape into the air? Wouldn't that light also affect the color we perceive? The answer is yes to both questions. These higher-order terms can be visible under the right circumstances, but they have a lot less energy in them than the first bounce. It's easy to find their phase shift—after k bounces, the effective path length (including the phase change on entering the film) is

$$\text{EOPL}_k = kd + \lambda/2 = 2kw\eta \cos\theta_t + \lambda/2$$

This wraps up our discussion of interference color. To summarize, we treat light as a wave. When it strikes the outside of the bubble, some bounces right back. The rest of the light enters the film and eventually returns to the air, where it will interfere with the light that bounced off directly. The film's thickness and the light's wavelength determine the phase shift of this second beam, and thus the intensity of the final, combined light. This process happens in parallel for every wavelength of light. So some wavelengths get cranked up and others get shut down, giving rise to a pattern of bright colors over the bubble's surface.



9 A soap-film wedge. (a) The interference colors. (b) The width of the film as a function of height. Notice that the shape isn't linear.



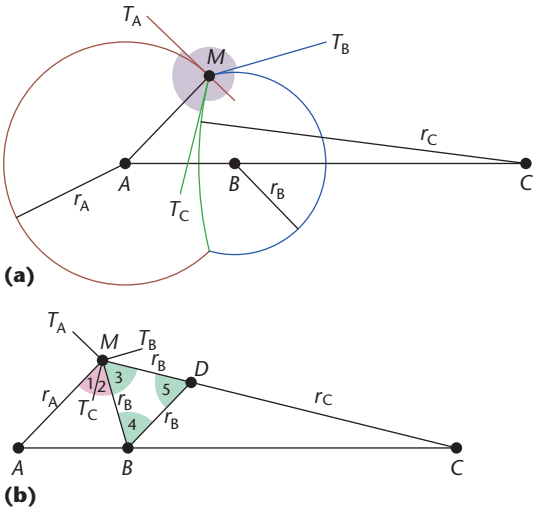
10 (a) Circles A and B (notice $r_A > r_B$). (b) The circle with radius r_C that forms the interface. (c) The joined pair.

Pairs of bubbles

In my last column we found that one of the properties of soap films is that when they meet in triples, each pair of films forms a 120-degree angle. Let's use that property to discover the geometry of a pair of merged bubbles. We'll look at everything in 2D to keep it simple.

Figure 10a shows the basic idea. We start with two bubbles, A and B, with radii r_A and r_B where $r_A > r_B$. When they come together, their common wall merges. Figure 10c shows that the 120-degree angle at top and bottom causes the common wall to bulge into bubble A. Last time we discussed how soap films want to minimize their area. You may recall that a sphere is the shape with the smallest surface area that encloses a given volume. That's why bubbles are spherical. This means that the common wall is spherical as well (or, in the 2D case, circular).

Since this common wall is a piece of a sphere (or circle) itself, we'll say it has center C and radius r_C; symmetry leads us to place C on the line AB, and the bulge into A places C on the B side of the line. So the question



11 (a) Two bubbles meeting at point *M*. All three bubbles meet at 120 angles—the three shaded angles are all 120 degrees. (b) The geometry associated with two bubbles. The pink angles are 30 degrees, and the green angles are 60 degrees. The numbers refer to the steps in the derivation in the text.

now is to locate point *C*, or find radius r_C (either one will give us the other).

Figure 11a shows the essential geometry. The three bubbles meet at point *M*. I've also marked three tangent lines at *M* as T_A, T_B, and T_C, which are each tangent to their namesake circle. We'll find that it's straightforward to find r_C once we've labeled everything in sight.

Let's dig in. I'll number our results so you can see how Figure 11b gets filled in. We know that triplets of films always meet at 120 angles, so $\angle T_A M T_C = 120$ degrees. By construction, $\angle T_A M A = 90$ degrees. We can find $\angle A M T_C = \angle T_A M T_C - \angle T_A M A = 30$ degrees, which is result 1. By similar arguments, $\angle T_B M T_C = 120$ degrees, and $\angle T_B M B = 90$ degrees. So $\angle B M T_C = \angle T_B M T_C - \angle T_B M B = 30$ degrees, which is result 2. To finish up what's happening at *M*, we observe that $\angle C M T_C = 90$ degrees, and $\angle B M T_C = 30$ degrees, so $\angle C M B = \angle C M T_C - \angle B M T_C = 60$ degrees, which is result 3.

We've finished most of the heavy lifting. Now I'll pull a rabbit out of my hat and create an auxiliary line. In Figure 11b I drew a line parallel to *AM* passing through point *B*. The line cuts side *MC* at point *D*. By construction, ΔBDC is similar to ΔAMC. Because *BD* and *AM* are parallel, $\angle AMB = \angle MBD$. Since $\angle AMB = 60$ degrees, $\angle MBD = 60$ degrees, which is result 4.

From results 3 and 4, we see that $\angle BDM = 60$ degrees (this is result 5), so ΔBDM is equilateral, meaning all three sides are the same length. Since $BM = r_B$, that means $BD = r_B$ and $MD = r_B$ as well (these are results 6 and 7).

Because ΔBDC is similar to ΔAMC,

$$\frac{BD}{AM} = \frac{DC}{MC}$$

$$DC = \frac{MC \times BD}{AM} = \frac{r_C r_B}{r_A}$$

Observing that $MD = MC - DC$, we can use this value for *DC* to find:

$$MD = MC - DC$$

$$r_B = r_C - \frac{r_C r_B}{r_A}$$

Dividing both sides by $r_C r_B$, we get our final result:

$$\frac{1}{r_C} = \frac{1}{r_B} - \frac{1}{r_A}$$

This equation expresses the essential piece of geometry for a pair of bubbles. Given the radii r_A and r_B of the two bubbles that join up, we can compute the radius r_C of the bubble that forms on their common wall.

Bigger clusters of bubbles are handled the same way. Just take them pairwise and find the common wall for each pair.

Triples of bubbles

Enough theory! Let's make some bubbles!

There are two steps to bubble making on the computer: computing the geometry and running the shader. I'll discuss the geometry first and the shader in the next section.

I'll assume that you have access to some form of high-level modeling language. Most of the popular commercial 3D packages have a scripting or plug-in system of some kind. To make my bubbles, I wrote a little modeling plug-in script for 3D Studio Max that makes clusters of triples. I just dial in the three radii I want, and it computes all the pieces and cuts them up properly.

In the following, I'll use a generic pseudo-code for most of this description, since the various popular programs and shading languages vary in their syntax. Figure 12 shows the code. For clarity, I used some parentheses that aren't strictly necessary, and I also included some operations with the origin (at point [0, 0]), which have no effect, but illuminate the logic.

The top of Figure 12 features three little utility functions—we'll see their purpose in a moment.

Upon entering the routine on line 5, I do a little bookkeeping and swap around the radii so that they're sorted $r_A > r_B > r_D$. For simplicity, I left that out and replaced it with a comment.

The first real job, starting on line 9, is to determine the radius of bubble C that forms the wall between bubbles A and B. It's just a matter of using the geometry in Figure 11b. I'll place point *A* at the origin, and points *B* and *C* on the positive *X*-axis. Since all we have are the radii, the first job is to determine the distance from *A* to *B*. We know $AM = r_A$ and $MB = r_B$. Writing θ_{M1} for angle $\angle AMB$, the cosine rule tells us that

$$AB^2 = r_A^2 + r_B^2 - 2 r_A r_B \cos \theta_{M1}$$

Life is good to us because we can see from Figure 11b that angle θ_{M1} is 60 degrees, so $\cos \theta_{M1} = 1/2$. That cancels the 2, leaving us with

```
 1. real getR(real R, real r) =
      (R*r)/(R - r)
 2. real getD(real R, real r) =
      sqrt((R*R) + (r*r) + (R*r))
 3. real getDist(real R, real r) =
      sqrt((R*R) + (r*r) - (R*r))
 4.
 5. makeBubble(rA,rB,rD) {
 6. — swap to make rA > rB > rD
 7.
 8. — use A and B to find bubble C
 9. dAB = getDist(rA, rB)
10. rC = getR(rA, rB)
11. dAC = getD(rA, rC)
12.
13. — use A and D to find bubble E
14. dAD = getDist(rA, rD)
15. rE = getR(rA, rD)
16. dAE = getD(rA, rE)
17.
18. — use A and B to find bubble F
19. dBD = getDist(rB, rD)
20. rF = getR(rB, rD)
21. dBF = getD(rB, rF)
22.
23. — locate D at distance dAD,
      rotate by angle BAD
24. thetaA = acos(((dBD*dBD)-
      (dAB*dAB)-
      (dAD*dAD))/(2*dAB*dAD))
25. dCenter = [dAD*cos(thetaA), -
      dAD*sin(thetaA)]
26.
27. — locate E on the line D-A
28. normDminusA = normalize(dCenter
      - [0,0])
29. eCenter = [0,0] + dAE *
      normDminusA
30.
31. — locate F on the line D-B
32. normDminusB = normalize(dCenter
      - [dAB, 0])
33. fCenter = [dAB,0] + (dBF *
      normDminusB)
34.
35. — make bubble spheres
36. bA = sphere([0,0], rA)
37. bB = sphere([dAB, 0], rB)
38. bC = sphere([dAC, 0], rC)
39. bD = sphere(dCenter, rD)
40. bE = sphere(eCenter, rE)
41. bF = sphere(fCenter, rF)
42.
43. — use CSG to make the cluster
44. — build the three outer sections
45. secA = bA - (bB + bD)
46. secB = bB - (bA + bD)
47. secD = bD - (bA + bB)
48.
49. — build the inner walls
50. secC = (bC inside bA) - bD
51. secE = (bE inside bA) - bC
52. secF = bF inside (bD intersect bE)
53.
54. cluster = group(secA, secB,
      secC, secD, secE, secF)
55.
56. — delete temporary objects
57. delete(bA, bB, bC, bD, bE, bF)
58.
59. return(cluster)
60. }
```

12 Pseudo-code for modeling a cluster of three soap bubbles.

$$AB = \sqrt{r_A^2 + r_B^2 - r_A r_B}$$

This is what our little helper function **getDist** computes for us. Now we want to find r_C, and we know from the last section that

$$\frac{1}{r_C} = \frac{1}{r_B} - \frac{1}{r_A}$$

Multiplying both sides by $r_A r_B$ and rearranging for r_C gives us

$$r_C = \frac{r_A r_B}{r_A - r_B}$$

which is what the function **getR** returns. Line 10 squirrels this away as r_C. Finally, we want to find the position of point *C*. We know it lies on the line from *A* through *B*, and we need to find the distance from *A* to *C*. Writing θ_{M2} for angle $\angle AMC$ in Figure 11b and applying the cosine rule again, we get

$$AC^2 = r_A^2 + r_C^2 - 2\, r_A r_C \cos \theta_{M^2}$$

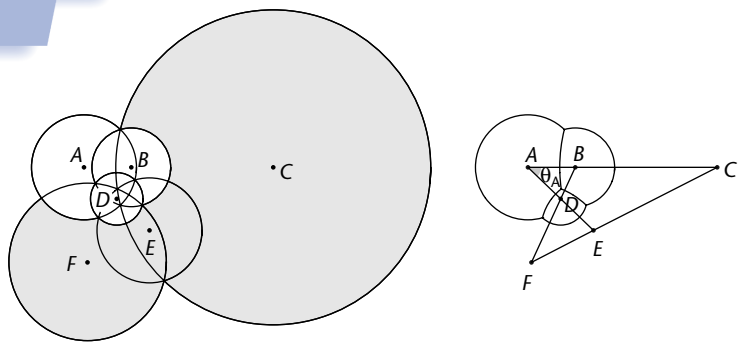
We know that θ_{M2} is 120 degrees, so $\cos \theta_{M2} = -1/2$. That cancels the 2 and changes the sign on the last term, resulting in

$$AC = \sqrt{r_A^2 + r_C^2 + r_A r_C}$$

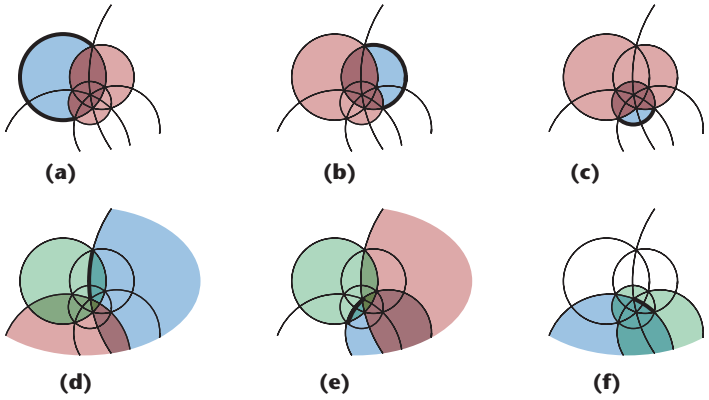
This is what our helper function **getD** computes, and we save this value in **dAC** on line 11.

Nothing in this process has been special for points *A* and *B*, so we repeat the same calculations to find the centers and radii of the other two circles.

Now we need to locate all the centers. As I mentioned before, to make life easy I place *A* at (0, 0). *B* is at (*dAB*, 0). *C* is also easy to place, at (*dAC*, 0). Now things get interesting, since we need to include the third bub-



13 Geometry for the cluster of three bubbles.



14 CSG to make a triple of three bubbles as in Figure 13. The lavender circle is the one we start with. Then we remove anything inside a red circle. If there are green circles, we keep only what's inside them. These expressions are one way to make the necessary pieces. (a) $A - (B + D)$, (b) $B - (A + D)$, (c) $D - (A + B)$, (d) (C in A) $- D$, (e) (E in A) $- B$, and (f) F in (D and E).

ble centered at D. Take a look at Figure 13 for the following geometry.

I start with the determination of point D on line 23. I build triangle ΔABD as in Figure 12 and find the angle θ_A with the cosine rule:

$$BD^2 = AB^2 + AD^2 - 2(AB)\ (AD)\cos\theta_A$$

Line 24 shuffles this around to compute $\cos\theta_A$. Now I have the polar coordinates of D: it's at an angle of θ_A clockwise from the X-axis through A, and at a distance of dAD. I find the X and Y values for D using the standard trig for polar coordinates on line 25.

To find E, I first build a unit length vector from A to D. Then I scale that by the distance from A to E, and voila! For clarity in the pseudo-code, I explicitly include the coordinates of A at [0, 0]; of course, in practice it's reasonable to leave that off. Locating F is done in the same way, except the vector is from B to D and the distance is from B to F. This one I need to add to B because it's not at the origin.

By the way, you may notice from Figure 13 that it appears that points C, E, and F—the centers of the circles forming the internal walls—all lie on a straight line. They do! If you like solving geometry problems you might want to give it a whirl proving it yourself—it would be too big a detour to prove it here.

Now it's just a lot of constructive solid geometry (CSG) slicing and dicing, as in Figure 14. The basic idea is to make six spheres—one for each the center sphere we just computed. Thanks to the symmetry of the situation, we have a few choices for how to isolate the necessary parts of these spheres. For example, to extract the right piece of bubble A, remove from it anything inside of B or D. Similarly, to get the right part of C, choose the surface that's inside bubble A, but outside of bubble F. Figure 14 shows a variety of recipes for building the cluster.

When you're done, make sure to clean up any loose vertices or disconnected edges that might have been produced by all this CSG surgery.

All the pretty colors

The last step to making bubbles is to make a soap-bubble shader. As almost always seems to be the case, writing a good shader seems to involve some judicious trading off of accuracy and realism with approximations and pragmatism. I mean, we could simulate all of this at the molecular or even atomic level, but it wouldn't show up in the results. The trick is to find a nice balance between simplicity, efficiency, and verisimilitude.

Some shaders kind of write themselves, and some involve personal choices. Here I'll describe my approach to a soap-film shader, which is actually pretty easy given the equations we derived earlier. A few more steps get us to a simple, accurate formula for efficient computing.

We found that the effective extra distance covered by the ray that goes into the bubble and then comes back out is given by

$$2w\eta\cos\theta_t + \lambda/2$$

We find the phase change, δ, or the number of radians that this distance represents, by multiplying by the number of radians in a cycle (2π) and dividing by the length of one cycle (λ):

$$\delta = \frac{2\pi}{\lambda}\left(2w\eta\cos\theta_t + \lambda/2\right)$$

Now that we know the phase shift δ, how can we compare the combined amplitude of the two waves that interfere with each other? When treating light as a wave, the magnitude of the wave is a complex number rather than the single real number that's often used in geometrical optics. Let's assume that the amplitude of the incident wave is given by the complex number A. If this wave is delayed by δ, then the phase-shifted wave is $Ae^{i\delta}$. The amplitude of the combined wave A_r is thus just the sum of the reflected wave and the delayed one:

$$A_r = A + Ae^{i\delta}$$

The *intensity* of light is the square of the wave's magnitude. If $A = (a + bi)$, we want $a^2 + b^2$ (remember $i = \sqrt{-1}$). There's an easy way to find this. For any com-

plex number $\omega = (a + bi)$, we can write $\bar{\omega} = (a - bi)$ and compute the product $\omega\bar{\omega} = a^2 + b^2$ ($\bar{\omega}$ is called the *complex conjugate* of ω). Let's find the intensity I_r of the reflected wave:

$$I_r = A_r \bar{A}_r = (A + Ae^{i\delta})(A + Ae^{-i\delta})$$
$$= A^2(2 + e^{i\delta} + e^{-i\delta})$$

Now recalling that $\cos\delta = (e^{i\delta} + e^{i-\delta})/2$, we can write

$$I_r = 2A^2(1 + \cos\delta)$$

We can use the half-angle formula $(1 + \cos\delta) = 2\cos^2(\delta/2)$ to write

$$I_r = 4A^2\cos^2(\delta/2)$$

That's pretty simple, but we still have to find A. We do know I_i, the intensity of the incoming light. The link between I_i and A is given by *Fresnel's formulas*, which tell us how much light is reflected off a surface for a given angle of incidence. They're straightforward, but a little bit messy to look at. (To find them, and the real physical data to make them work right, look in just about any book on modern optics or modern rendering—some suggestions appear in the "Further Reading" sidebar.)

Fresnel's equations are implemented in almost every modern renderer. They're particularly convenient in graphics because you can pretty much just type them in and use them right away. But if you don't want to implement them for real, you can create a cheap approximation by scaling the reflectivity by the incident angle's cosine. When the incident angle is almost 0, there's basically no reflection; when the angle is nearly 90, there's nearly total reflection.

```
1. a = 2 * PI * w * eta *
     cos(theta_t);
2. for (index=0; index<81;
     index++) {
3.   lambda = 380. + (400. *
     (index/80.));
4.   sina = sin(a/lambda);
5.   ref[index] = 4. *
     incident[index] *
     fresnel[...] * sina * sina;
6. }
```

15 Psuedo-code for my soap-film shader.

Given a reflectivity R_f from Fresnel, we can write $A^2 = I_i R_f$. So our equation becomes

$$I_r = 4 I_i R_f \cos^2(\delta/2)$$

If we plug in half our value from δ above, we get

$$I_r = 4 I_i R_f \cos^2\left(\frac{2\pi}{\lambda} w\eta\cos\theta_t + \pi/2\right)$$
$$= 4 I_i R_f \sin^2\left(\frac{2\pi}{\lambda} w\eta\cos\theta_t\right)$$

where I substituted $\cos(\theta + \pi/2) = \sin(\theta)$.

Ta-da! This formula is all we need to make a soap-film shader. Figure 15 gives the basic steps.

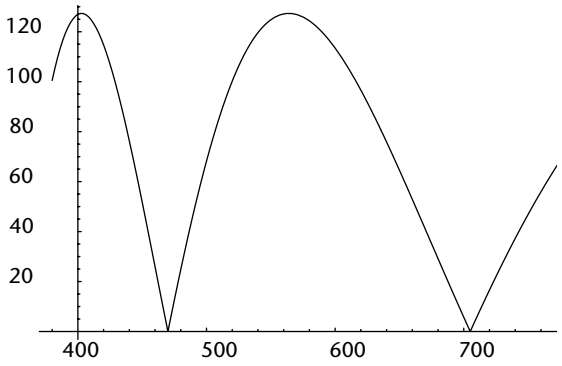
Further Reading

Three books that I mentioned in the last issue continued to be useful to me in preparing this column. They are *Soap Bubbles: Their Colors and Forces Which Mold Them* by C.V. Boys (Dover Publications, New York, 1959), *The Science of Soap Films and Soap Bubbles* by Cyril Isenberg (Dover Publications, New York, 1978), and *Demonstrating Science with Soap Films* by David Lovett (Institute of Physics Publishing, Bristol, 1994).

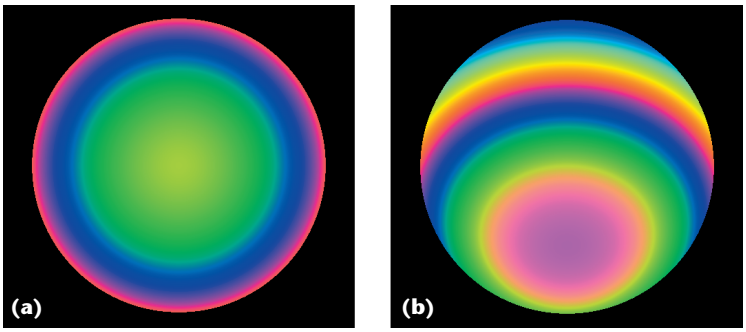
You can see some lovely soap-bubble images rendered by John Sullivan with a custom RenderMan shader on the Web at http://www.math.uiuc.edu/~jms/Images. There are lots of interesting Web sites that run the gamut from bubble mathematics to how to make good soap-film solutions. You might want to take a look at the Bubblesphere at http://bubbles.org/index.htm, the Bubbles Theme Page at http://www.cln.org/themes/bubbles.html, and Maarten Rutgers' page at http://www.physics.ohio-state.edu/~maarten/work/soapflow/soapintro/basicsoap.html.

Some computer-graphics articles on interference colors are "Newton's Colors: Simulating Interference Phenomena in Realistic Image Synthesis" by Brian E. Smits and Gary Meyer (*Proc. Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, June 1990), "Ray Tracing Interference Color" by Maria Lurdes Dias (*IEEE Computer Graphics and Applications*, March/April 1991), "An Approach to Geometrical and Optical Simulation of Soap Froth" by Isabelle Icart and Didier Arquès (*Computers & Graphics*, June 1999), and "Deriving Spectra from Colors and Rendering Light Interference" by Yinlong Sun, F. David Fracchia, Thomas W. Calvert, and Mark S. Drew (*IEEE Computer Graphics and Applications*, July/August 1999).

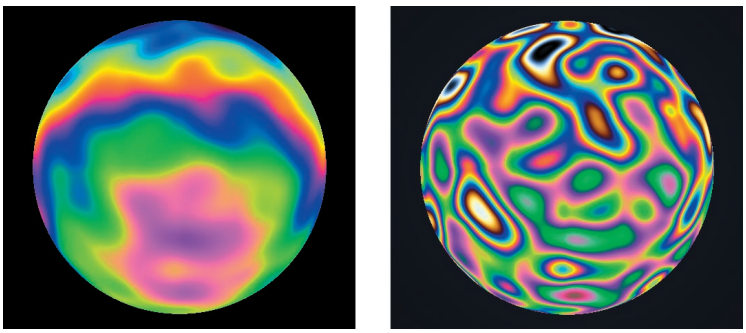
For more information on Fresnel formulas and color computations (including converting *XYZ* to RGB and creating colors from triples), you can look at most any modern text on rendering in computer graphics. Two good places to start are *Advanced Animation and Rendering Techniques* by Alan Watt and Mark Watt (Addison-Wesley, New York, 1992) or my own book, *Principles of Digital Image Synthesis* (Morgan-Kaufmann, San Francisco, 1995).



16 The result of passing a white spectrum through a 500-nm film.



17 (a) A bubble of radius 5,000 nm and thickness 500 nm. (b) After draining, the thickness is 300 nm at the top and 700 nm at the bottom.



18 Adding noise to Figure 17 to simulate the sloshing around of liquid as the bubble moves in the air.

We begin with incident light represented as a sampled spectrum. In my renderer I associate an amplitude with every 5 nm from 380 to 780 nm, creating an array with 81 elements (ways to get a spectrum from a color are available from the "Further Reading" sidebar). The shader is given this incident light and the geometry of the intersection.

The shader receives values for w and $\cos \theta_t$ from the system, derived from the scene information. Roll this all together into a temporary variable in line 1. Now step through the incident light array, compute each term one by one, and store them in the reflection array.

Figure 16 shows the result of passing a flat, white spectrum through a 500-nm film. Note that this reinforces some wavelengths and cancels others entirely.

That's pretty much it for the basic shader—I promised it would be simple! My version is an easy plug-in for 3D Studio Max, which I used to render the pictures in this article. Once you've computed the spectrum, you'll need to convert it to RGB. The usual way to do this is to run the spectrum through the International Commission on Illumination (Commission Internationale de L'Eclairage, or CIE) color-matching functions to get an *XYZ* color-space representation, and then convert those *XYZ* values to RGB for a particular monitor. This is standard computer-graphics stuff that you can find in any graphics textbook (see the "Further Reading" sidebar).

Let's look at the shader in action. Figure 9a shows a vertical film in front of a black background. Simulating the pull of gravity on the liquid, the film forms a wedge 60-nm thick at the top to 2,460 nm at the bottom. The wedge isn't linear; Figure 9b shows the thickness over the height of the film. We're looking directly at the film, which is illuminated by a light on our forehead.

Note that the wedge is black at the top. Remember that the light shifts $\lambda/2$ when it enters the film. The additional distance within the film is negligible at the top, so the light that passes through the film is just about $\lambda/2$ shifted from the reflected light, resulting in total destructive interference. As we work our way down the film, we first see the very bright colors we associate with soap bubbles, each resulting from some narrow band of light being subtracted out from the white light. When the film gets thicker, the time it takes the light to pass through and come back out lasts longer than the amount of time that the incident light stays coherent. That is, by the time the light comes back out of the film, the incident light is at a different, random phase. This effectively knocks out the interference effects, which is why we typically don't see interference in thicker, transparent objects, such as a piece of window glass.

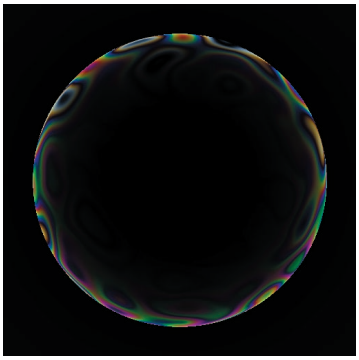
Let's look at a very simple soap bubble—a single sphere. For now, let's disregard the Fresnel term, which modulates the strength of the reflection based on the angle of incidence. If the bubble has just formed, then the bubble is a spherical shell with uniform thickness. Figure 17a shows an example of a bubble with a radius of 5,000 nm and a thickness of 500 nm. I modeled this by placing an air-filled sphere of radius 4,000 nm inside a soap-filled sphere of radius 5,000 nm.

I'm assuming that we're looking at the sphere in a room that's filled with light. So at each point on the bubble, we're looking at light that arrived from somewhere, split into two (some bouncing off the surface and some entering the film), and then rejoined to interfere and arrive at our camera. The different colors are due entirely to the different path lengths. For example, the light passing through the film in the middle of the picture arrives about normal to the surface, so it passes (twice) through 500 nm of film. But the light toward the outside arrives at a more oblique angle, and thus passes through more film.

After the bubble has floated around for a moment, it

starts to drain—that is, the liquid starts to move to the bottom. We can model this by moving the air-filled sphere upward. Figure 17b shows the result after the bubble has drained a bit, so the film is 3,000 nm at the top and 7,000 nm at the bottom. You can start to see the variation in color due to the draining.

Why don't we see bubbles looking like this in the real world? It's because our perfect spherical shells don't match the geometric reality of soap bubbles, which are liquids sloshing around in the air. Let's simulate that sloshing fluid with some noise. Figure 18 shows what happens if we add increasing amounts of noise to the thickness computation. Finally, Figure 19 incorporates the Fresnel term. Remember that we're looking at the bubble against a black background.



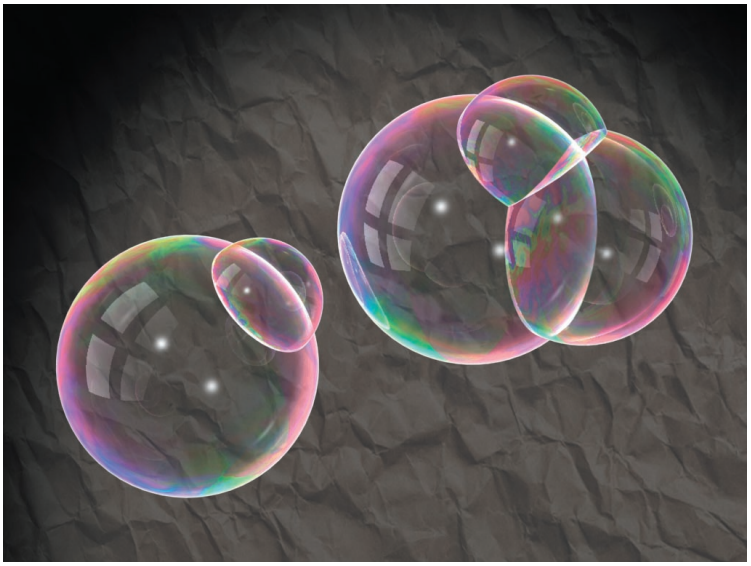
19 Including the Fresnel term to Figure 17.

All together now

The only thing left is to bring together the model and the shader. Figures 20 and 21 show some bubble clusters modeled and shaded using these techniques. Note the reflection of the four-paned window—I think it's a law that all bubble pictures reflect a window.

There's something very attractive to me about understanding nature well enough to create an accurate simulation. But I need to point out that at the very last step of the shader, I introduced some noise to account for the variation in thickness that normal bubbles experience when they're in the air. This noise had the effect of basically scrambling all of our careful calculations, giving us just a bunch of wild swirls of colors. Now if we'd done a real fluid-dynamics simulation we'd have something accurate, but noise is just noise.

This tells us how to take a giant shortcut in our bubble shaders. Instead of computing the thickness, adding noise, and computing the interference colors that result, we can simply take a gradient of bright colors, swirl them up with noise, and map them onto the surface. The result will look a lot like a real bubble. As long as the colors and noise were well controlled, you wouldn't be able to tell the difference between a properly computed shader and a total hack. I'm of two minds about this. One the one hand, it seems kind of sad that a simple hack can look as good as a proper simulation. On the other hand, I think that's kind of cool. ■



20 Two bubble clusters: a pair and a triplet.

21 A swarm of bubbles.

Readers may contact Glassner by e-mail at andrew_glassner@yahoo.com.