Duck!

Andrew Glassner

When ducks swim on a smooth deep lake, they create a V-shaped ripple of waves behind them. Boats and ships do the same thing (see Figure 1), as do human swimmers. I saw a duck swimming across a glass-smooth pond a few weeks ago, and I wondered what it might be like if I could choreograph a flock of trained ducks to swim as I wanted. Could the ducks be induced to make interesting patterns out of their overlapping wakes?

Since I probably couldn't get real ducks to obey my instructions, I decided to build a flock of virtual ducks and experiment with them. The only problem with this scheme was that it required first finding a way to compute the wake that a duck creates behind it as it swims. Writing an accurate, general-purpose, fluid-flow simulator is a difficult job. Happily, finding the wake created by any object moving with constant speed in deep water is much easier.

Finding a duck's wake is of course just a special case of the more general problem of the modeling and simulation of water. Computing water flow is nothing new in either engineering or computer graphics. There have been thousands of papers and hundreds of books written on the simulation of water, many of which are intricately detailed.

Water is very special stuff and has some surprising properties. Hydraulics engineers describe water as a nonviscous incompressible fluid that moves under the influence of gravity. The important word for us in this definition is incompressible.

Air is compressible. If you walk into a sealed room, then your body will displace the air that was where your body now is. That air mixes with the other air in the room, creating a slightly higher air pressure on your skin. You can even compress a lot of air into a small space and store it efficiently, which is why underwater divers get so much time out of a single tank of air.

1 A photograph of the wake behind a moving boat.

Water, by contrast, isn't compressible. If you try to walk into a closed room full of water, you won't get in the door. There's nowhere for the water to go when your body tries to push it out of the way.

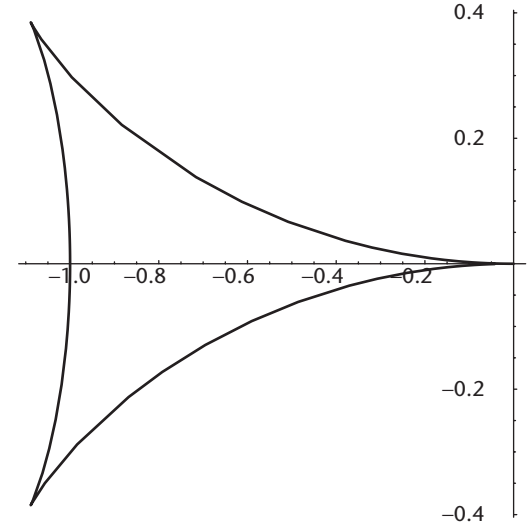
This principle is illustrated by a famous (and probably apocryphal) story of the Greek philosopher Archimedes of Syracuse (who lived around 240 BC). The story begins with the king coming to Archimedes with a problem. The king had purchased a new crown of solid gold, but the king was suspicious and thought that perhaps the artisans who made the crown had cheated and simply melted a layer of gold leaf over a core of some cheaper metal. He asked Archimedes to determine if the crown really was made of nothing but solid gold, but with the condition that Archimedes not damage or alter the crown in any way. This seemed an impossible problem. Then one day Archimedes climbed into a bathtub, causing the water to slosh over the sides. According to legend, Archimedes suddenly saw how to solve his problem, and ran naked down the street shouting "Eureka!"

Archimedes' insight was that the volume of water that had spilled over the side of the tub was exactly equal to the volume of his immersed body. In other words, because water can't be compressed, the volume of water displaced by an object is equal to that object's volume. So Archimedes dunked the crown into a tub of water and measured how much water spilled out. Then he weighed the crown and divided the weight by the volume to get the crown's average density. It turned out to be the density of gold, so it meant that unless the artisans had found a way to make another metal with the same density as gold (which nobody knew how to do), then the crown was indeed made of solid gold after all.

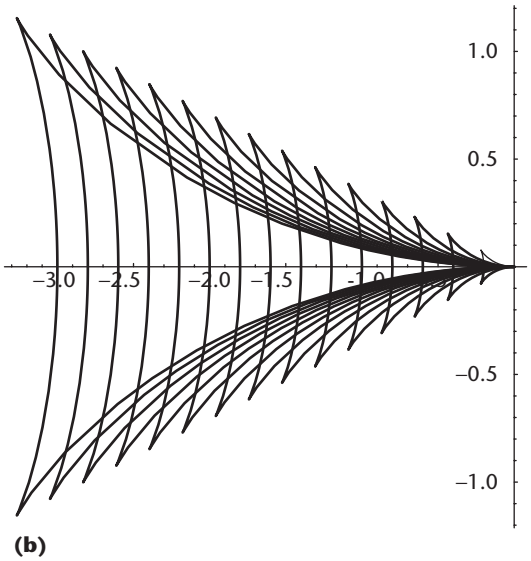
Drawing our inspiration from this legend, let's push around the water displaced by a swimming duck and create the wave pattern that flows behind it.

Kelvin waves

To find a duck's wake, I'll assume the duck is in a pool of water that's infinitely deep. This means that no waves bounce off the bottom and come back up at us, and it also



(a)

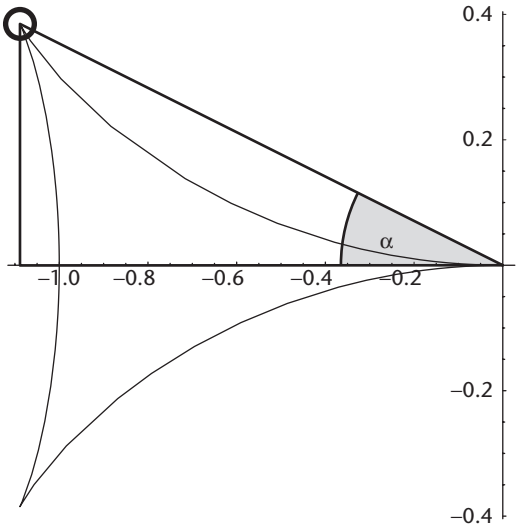


(b)

2 (a) A Kelvin wave for ψ = 1. (b) Superimposed Kelvin waves for several values of ψ.

means that we don't have to worry about where the water goes when we push it downward. It just goes down, pushing down all the water below it as well. In real life, of course, eventually that downward-moving water will hit the bottom and then move to the sides. With this assumption, it's possible to start with a few basic equations of fluid motion and related boundary conditions and find an explicit formula for the wave's shape propagating behind the duck. I won't provide this derivation here, because it's complicated and not very revealing. You can find the details in chapter 2 of *Water Waves and Ship Hydrodynamics: An Introduction* by R. Timman, A.J. Hermans, and G.C. Hsiao (Delft University Press, 1985).

The bottom line is that we can find the wave by tracing out a parametric curve, which I'll call $K(\psi, u)$. The curve K is given by two expressions, which give us its x and y components. Each component of K is defined by two parameters. The first, ψ, is called the *phase* of the



3 Finding the angle α that points to a Kelvin wave's cusp.

wave. Basically this is a shape control that accomodates the fact that the wave gets larger as time passes. The second parameter, u, sweeps out the curve for a given value of ψ. Here's the definition of K:

$$K_x(u) = \frac{-\psi}{4}(5\cos u - \cos 3u)$$

$$K_y(u) = \frac{-\psi}{4}(\sin u - \sin 3u)$$

where $-\pi/2 \le u \le \pi/2$.

Sometimes K is called a *phase wave*. It's also called a *Kelvin wave* after the physicist Lord Kelvin (William Thomson), who was the first to study this phenomenon mathematically. You can think of ψ as a measure of how long the wave has been expanding.

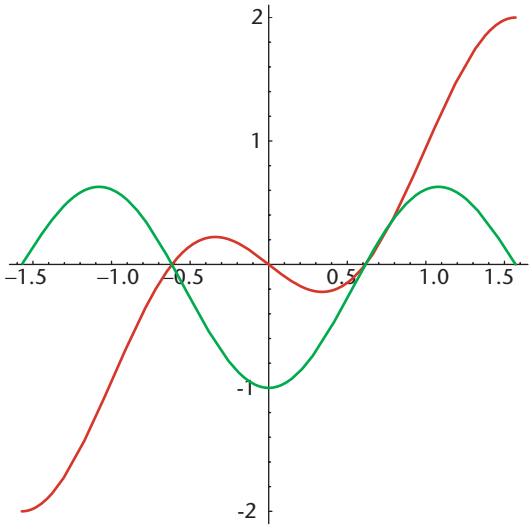
Figure 2a shows this curve for ψ = 1 behind a duck swimming from left to right. In Figure 2b I've overlapped several plots of K for different values of ψ.

Notice that the waves all lie in a cone behind the duck. For some applications it's useful to know the angle of this cone, since everything outside of the cone is guaranteed to be unaffected by this particular duck. The half-angle at the tip of this cone is called the *Kelvin angle*, labeled α in Figure 3. To find the Kelvin angle, we can determine the locations of the two points, or *cusps*, of the phase wave.

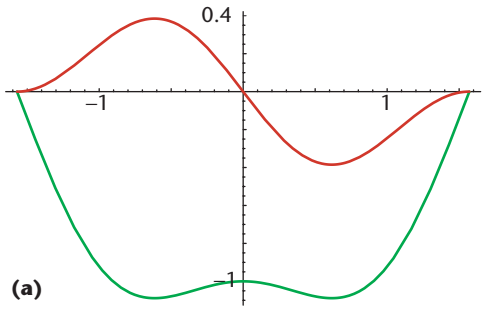
These cusps appear where the derivatives of K_x and K_y are both simultaneously zero with respect to the variable u. These derivatives are

$$\frac{dK_x}{du} = \frac{-\psi}{4}(-5\sin u + 3\sin(3u))$$

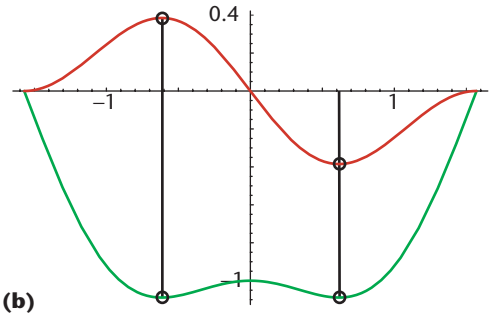
$$\frac{dK_y}{du} = \frac{-\psi}{4}(\cos u + 3\cos(3u))$$



4 Plots of the derivatives of a Kelvin wave. The dx/du is in red, while dy/du is in green. The Kelvin angle is where these are both zero simultaneously.

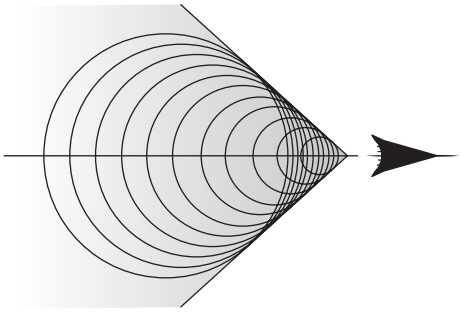


(a)

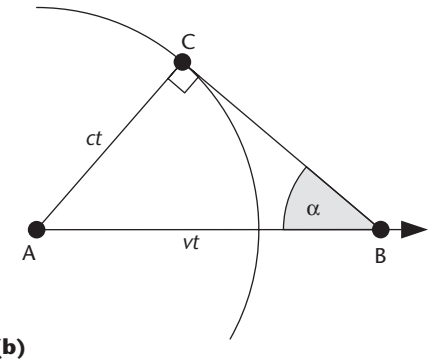


(b)

5 (a) The Kelvin functions $x(u)$ (red) and $y(u)$ (green) for $\pi/2 \le u \le \pi/2$. (b) The Kelvin angle is where these functions have zero slope.



(a)



(b)

6 (a) Forming a supersonic boom. Each of the circles is an expanding pulse of air from a point on the plane's path. (b) Finding the angle at the apex of a cone.

and are plotted in Figure 4. It's easy to confirm that these functions are both zero at $u = \cos^{-1}(\sqrt{2/3})$. If we plug this value for u into the curve K above, we get a point at about (–1.088, –0.385), as Figure 3 shows. The angle between the X-axis and the line from the origin to this is about 19.5 degrees.

I plotted $K_x(1, u)$ and $K_y(1, u)$ in Figure 5a, and marked the locations of $u = 19.5$ degrees and $u = -19.5$ degrees in Figure 5b. Reassuringly, both graphs are flat at those points.

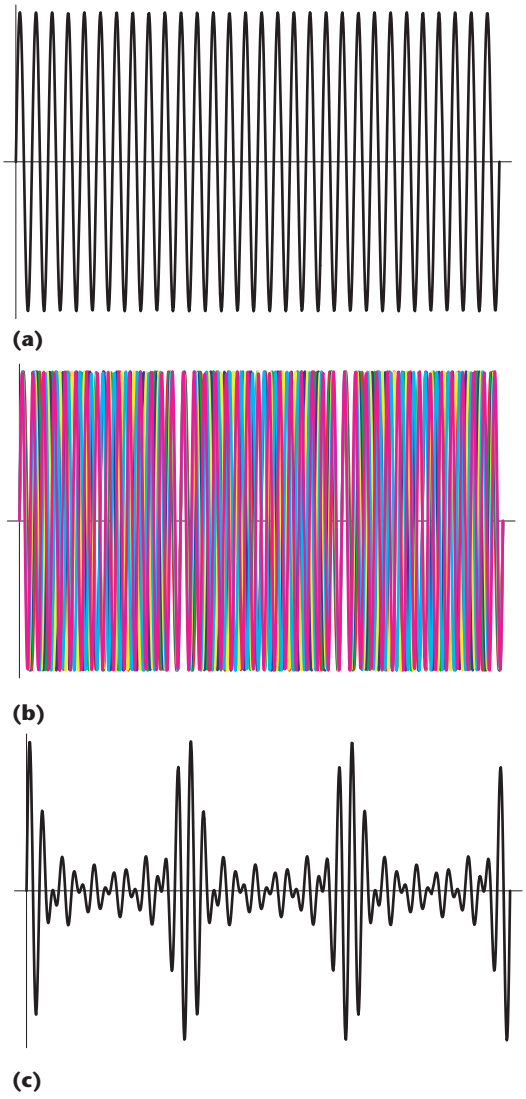
We found the Kelvin angle's size with an algebraic computation. It turns out that there's also a nice geometric argument that yields the same result. I like geometric approaches in general for their intuitive appeal, but this one also helps us understand something about water waves in the process.

To start with an analogy, think of an airplane flying left to right with constant velocity v, as Figure 6a illustrates. We'll just look at a 2D, simplified version of the complex 3D waves created by the airplane.

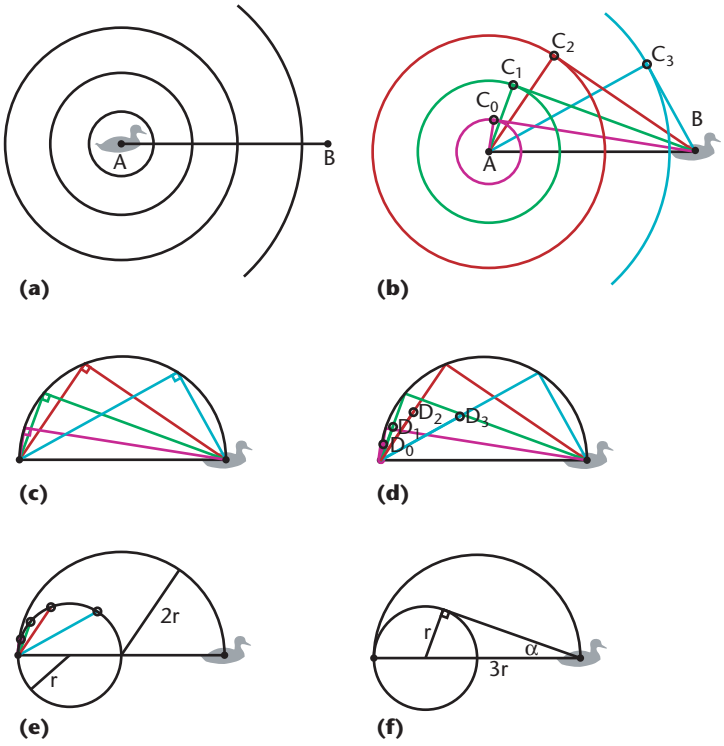
Imagine that as the plane flies, at every point it creates a whole bunch of circular waves of different frequencies, all radiating away at once. I drew these as circles in Figure 6a. An interesting property of sound waves in air is that they all travel at the same speed. That is, the speed of sound doesn't depend on wavelength. Although the plane makes disturbances of many different frequencies, they all radiate away from the point at the same speed.

In Figure 6a the airplane is going at supersonic speed. That means the plane is moving faster than the waves are expanding. What's the angle of the cone that contains these waves? Looking closely at Figure 6a we can see that all the circles are tangent to the cone that contains them. In Figure 6b I picked any circle. The circle was created when the plane was at point A, but the plane is now at point B. If the plane is flying with velocity v and it took t seconds to get from A to B, the distance $|AB| = vt$. If we write c for the speed of sound, then the radius of the circle is ct. Let's call the point of tangency C. Now we have a right triangle ACB where $|AB| = vt$, and $|AC| = ct$, and we want to find angle α at point B. Since $\sin \alpha = |AC|/|AB| = ct/vt$, then $\alpha = \sin^{-1}(c/v)$.

Now let's return to the water and more duck-like speeds. The situation will be somewhat different than the airplane because water waves of different wavelengths travel at different speeds.



7 **Several sine waves of almost the same frequency add up to make clear packets. (a) sin(x) for $0 \le x \le 60\pi$. (b) Sine waves from sin(x) to sin(1.5x) in steps of 0.1x. (c) sin(x) + sin(1.1x) + sin(1.2x) + sin(1.3x) + sin(1.4x) + sin(1.5x).**



8 **A geometric construction for the Kelvin angle. (a) A few different waves radiated from point A. (b) The angle each forms at point B, with a point of tangency given by a point C_i. (c) The points of Figure 8b lie on a semicircle. (d) The waves from A move at the group velocity, which is half of the phase velocity, so they only get halfway, to points marked D_i. (e) The points D_i lie on a semicircle. (f) Finding the angle of the semicircle in Figure 8e.**

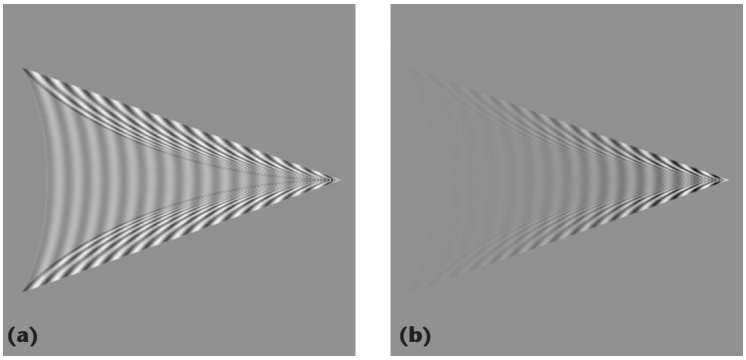
Thinking now of a duck traveling from A to B, at each point in its journey the duck creates a whole bunch of different waves at slightly different wavelengths. When many different waves of almost the same wavelength combine with each other, the result is that they tend to cancel out in some places and reinforce in other places. Figure 7 shows this phenomenon for six sine waves of only slightly different frequencies. In Figure 7a I show a sine wave from 0 to 60π. In Figure 7b, I plotted that wave as well as five more of slightly higher frequencies, and it looks like a mess. In Figure 7c, I added up the six waves from Figure 7b, and a surprising structure emerges. The waves have combined to create a series of *packets*.

This is what happens to the water waves created by our duck. At each point the duck creates a variety of waves of slightly different wavelengths, but rather than disturbing all the water uniformly, they create easily discerned packets, as in Figure 7. If you watch a fine-grain water wave start behind a duck, you can see it quickly move out until it reaches an existing packet, where it combines with the packet for a while and then emerges from the other side to dissipate in the open water.

How fast does the packet move? Individual waves move at a speed given by the *phase velocity*. But the packet moves at the *group velocity*, which in deep water is exactly half the phase velocity.

As a rough analogy, think of a highway where an accident has recently been cleared. There's still a clump of slowly moving cars near the scene of where the problem was. New, fast-moving cars arrive at the tail of the pack and are forced to slow down, while cars near the front can regain their speed and drive away. Thus, there's a slow-moving clump of cars traveling up the highway, constantly being refreshed with new cars and losing old cars. The car membership in the blob constantly changes, and the cluster moves more slowly than any of its components.

Let's see how we can use this information to find the Kelvin angle. In Figure 8a, I drew a number of circular waves as radiated from point A (the duck is now at point B). If we imagine that each one of these waves defines a cone behind B, then we can draw a line from B tangent to each circle, creating a point C_i for each wavelength, as in Figure 8b. Figure 8b shows four different waves and thus four different tangency points. In reality,



9 (a) The height field for a series of Kelvin waves, where the amplitude varies with the sine of the phase. (b) Adding damping to Figure 9a.

there's a smooth distribution of these waves from short to long.

Because every line from B to one of A's circles is always tangent to the circle at the corresponding point C_i, the angle formed at each of these points C_i is always a right angle. You may remember the geometric fact that all right triangles built above a common hypotenuse end up with their third vertex on a shared circle that uses that hypotenuse as a diameter. Figure 8c illustrates this point.

These points C_i are the points that enclose the individual waves. But remember that these waves combine to form a packet, as in Figure 7. So what we really want to find is the cone that encloses the packet, not the individual wave crests. Remember from above that the group velocity in deep water is exactly half of the phase velocity. Since every line AC_i gives us the distance to a wave crest, if we mark a point halfway between A and C_i we've found the packet's location. I marked these points in Figure 8d as D_i. So now our goal is to find the cone from B that encloses all the points D_i.

If you enjoy geometry problems, you might want to stop reading here and work out the shape formed by the points D_i.

Figure 8e shows the answer. It's a new circle that has a radius of $|AB|/4$, and is located 1/4 of the way from A to B. All of the wave packets generated at point A live inside this circle. Thus, to find the angle of the wake behind the duck when it's at B we need only draw a line from B that's tangent to this circle and read off the angle at B. Figure 8f shows the setup. If we write $r = |AB|/4$, then $\alpha = \sin(r/3r) = \sin(1/3)$. If we evaulate $\alpha = \sin^{-1}(1/3)$, we get a Kelvin angle of about 19.5 degrees, just as we found before. It's always a good day when we can reach the same result by two entirely different approaches!

In summary, all the Kelvin waves created by any object moving with constant speed in deep water lie within a cone with a half-angle given by the Kelvin angle of about 19.5 degrees.

Wake up!

Now that we know how to create Kelvin waves $K(t)$, the only question left is how to use them to make a trail behind a swimming duck. The easiest way is to let them grow as the duck moves. But as we can see from Figure 2, just superimposing the waves on each other will eventually lead to a black blob, which of course isn't what we see in nature.

What we want to do is have the waves go up and down, as they do in the real world. As time goes by, any given Kelvin wave undergoes three changes. First, it gets larger. This is automatically handled by the phase variable ψ. If we plot $K(t\psi, u)$ where t is time, then the wave naturally grows with t. Second, the wave goes up and down, or oscillates. We can model that easily by setting the amplitude A of the wave to $A = \cos(ct)$, where c is the speed of the Kelvin wave.

If we draw a great many waves with these rules and let them accumulate, the result is Figure 9a, where here I computed the waves for a duck swimming left to right.

In this and all similar figures in this column, the waves are represented as a height plot. We're looking straight down on the water. White points are closest to us (that is, they correspond to crests) and black points are farthest (corresponding to valleys). Neutral gray is the undisturbed water level.

The third change to Kelvin waves as they expand is that they lose energy. That is, the amplitude diminishes with time. I use an exponential function to control this, setting $A' = Ae^{-(bt)^2}$ for a user-defined constant b. The result of incorporating this third component gives us the wake of Figure 9b.

To create Figure 9b, I first created in memory a grid of floating-point values and initialized them all to 0. This is the sampled wave pattern into which all the Kelvin waves will be added (let's call it W).

Now I start at point B (I found it easier to move from the end to the start) and draw a Kelvin wave with a phase of $\psi = 0$ and amplitude of $A = 1$. Using the equations K_x and K_y, I step u in many small pieces from $-\pi/2$ to $\pi/2$. The number of steps is user-controlled; a smaller number results in coarser patterns but faster running time. Typically I use a small number of steps for u while working on my patterns and then crank it up high for the final images. For each value of u I get a floating-point location (x, y). I then find the four nearest pixels in W to that point and add to them the appropriately weighted values of A using bilinear interpolation.

When I've finished the u sweep, I then take a small step backwards on the path toward A, creating a new point P. As with the control of u, the number of steps taken from B to A is user-defined. Again, more steps result in smoother images. First I calculate the path length from P to B. If AB is a straight line, this is just the distance $|PB|$. But if the duck is swimming on a curved path, I need to find the distance covered by the duck along the path from P to B. Sometimes I can find this analytically—I have special-purpose code for simple paths like straight lines and circular arcs. For more general curves, I have to compute an approximate arc length by taking many little steps along the path from P to B and adding up all the little straight lines. Of course, this is the reason that I work backwards. Each time I compute a new point P, I don't need to compute the entire distance $|PB|$, but just the shorter distance

to the previous point P and then add that in to the previous distance.

Given a point P and its distance along the path to B, I use the duck's speed to find the time t it took the duck to reach B from P. Using this value of t, I calculate the phase ψ and the amplitude A. Then I walk though the values of u again, finding points along the way, and adding in the value A to those points. Then I take another step toward A and repeat the process. When I'm done with all the paths for all the ducks in the scene, I scan the grid W and scale all the values within it to the range [0,1]. Then I multiply each pixel by 255 and write it out as a grayscale image.

To make the images in Figure 9 I used a grid that was 512×512 elements large. If we call this grid 512 units on a side, I plotted from 50 to 100 values of ψ for every unit of length of the path on the grid. I typically took 500 to 1,000 steps in u to generate the wave for each value of ψ.

This is a pretty good rough-and-ready simulation, but it has one big problem. As we walk along K in equal steps of u, the points we generate aren't equally spaced along the curve. Figure 10 shows 100 points created by chopping the u interval $[-\pi/2, \pi/2]$ into 100 equal steps and then plotting the resulting points. Obviously, they tend to bunch up near the cusps.

This can create some unpleasant artifacts at the edges of the duck's path. The value of plotting a great many K waves is that they all add up to make a smooth field. But if there are pixels that only get written to once or twice, they don't have a chance to combine with the other waves. If these pixels happen to have a large amplitude, when I search the image for the largest and smallest values prior to the scaling step, these can stand out. The result is that most of the wake is neutral gray, with a smattering of white and black pixels along the edges.

To solve this, I precede the scaling step with a smoothing step. First I create a copy of W in a new grid called $W2$. Then I look through all the pixels in W that have been written to and compare each pixel's value to the average of its four immediate neighbors. If the difference between the pixel and its neighbors is above a threshold, then I replace that pixel's value in $W2$ with the average of its four neighbors in W. When I'm done with this pass, I copy $W2$ back to W and throw $W2$ away. Now that the numerical rogues have been tamed, I scale the data and write it out as a grayscale file, as before.

Figure 11 shows a duck on rather calm water. The wake pattern behind the duck is Figure 9b. The only "real" thing in this image is the photo of the shoreline in the background that I took of Green Lake in Seattle. Everything else is synthetic.

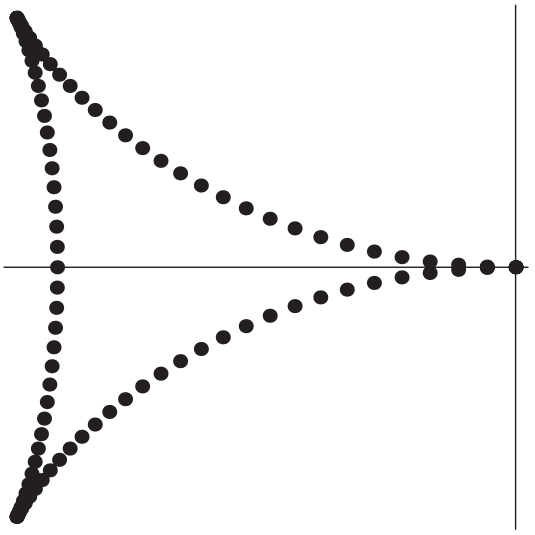
Throwing a curve

I've described how I plot a path from A to B, but what if the path isn't a straight line? I specify a duck's overall journey by stitching together a series of short paths. My palette of available paths contains line segments, circular arcs, spirals, Bezier curves, and cubic splines.

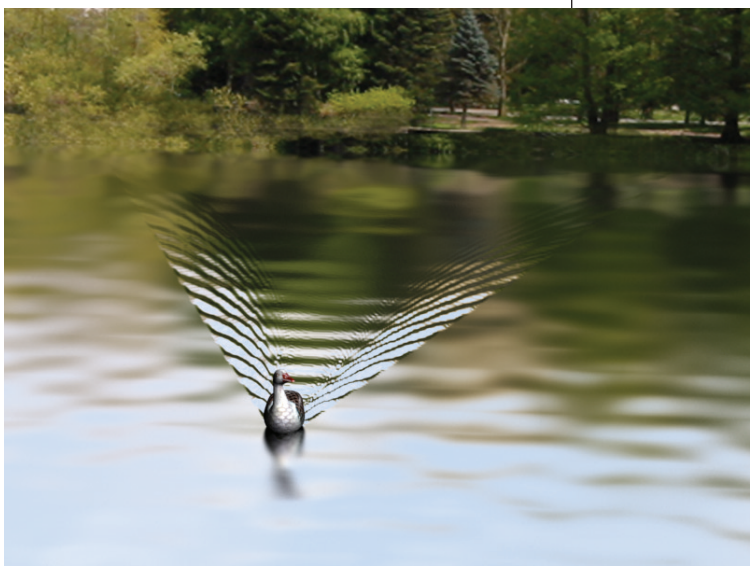
We saw an example of a straight-line wake in Figure 9b. Figure 12a (next page) shows the pattern caused by a cubic spline, where I've exaggerated the waves' heights so you can see the beautiful patterns. Figure 12b shows a proper result. In Figure 12c, I highlighted the path taken by the slowly swimming duck in red. Here I've kept the damping factor to a low value so that you can see the interesting shapes created by the Kelvin waves. Notice that the back waves are huge compared to those in the front because of the additional amount of time they've had to spread. Figure 12d shows this pattern in 3D behind our swimming duck.

I show another spline in grayscale in Figure 13a (next page), and in 3D in Figure 13b. I turned the damping back on here so that the oldest waves have just about (but not quite) died out.

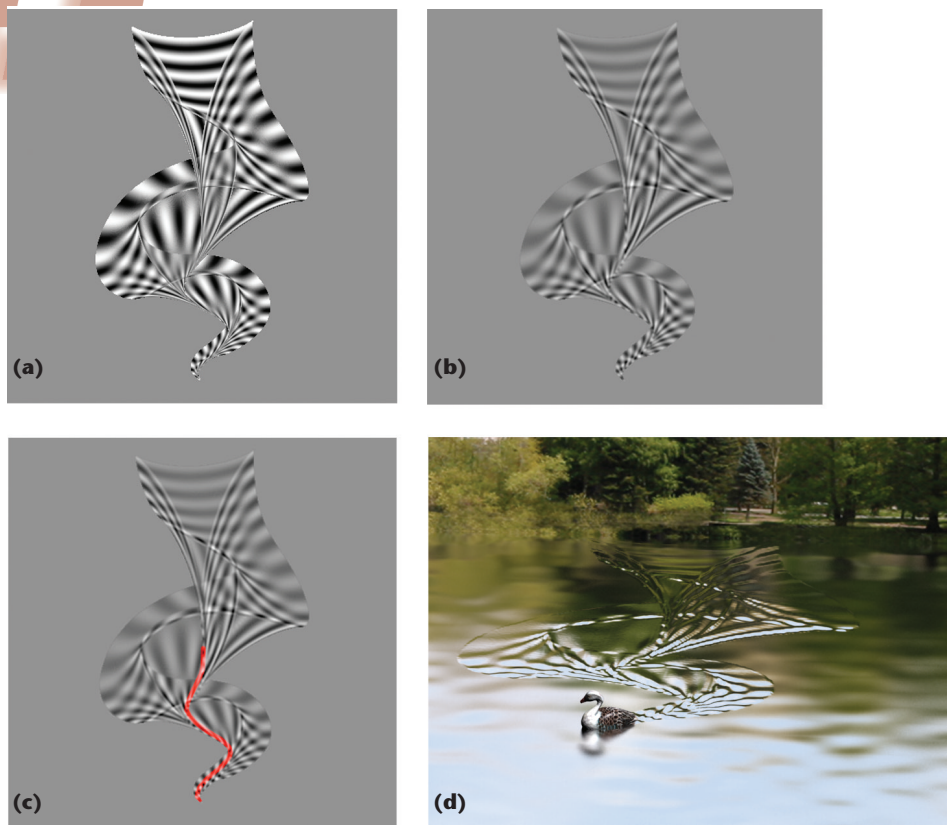
The only trick to stitching together these different segments is to ensure that the end phase of one segment of



10 The range $-\pi/2 \le u \le \pi/2$ cut into 100 equally-spaced points and plugged into K. Note that the points aren't equally-spaced along the path of K.



11 A synthetic image using the wave pattern of Figure 3b.



12 (a) A wave pattern for a duck swimming on a twisty path. The heights are exaggerated for legibility. (b) Including damping into Figure 12a. (c) The duck's path highlighted in red. (d) A synthetic image using the wave pattern of Figure 8b.

the path matches the start phase of the next. This is easy to manage by working backwards along the segments. The phase ψ just keeps growing with your accumulating, approximate arc length and everything goes together smoothly.

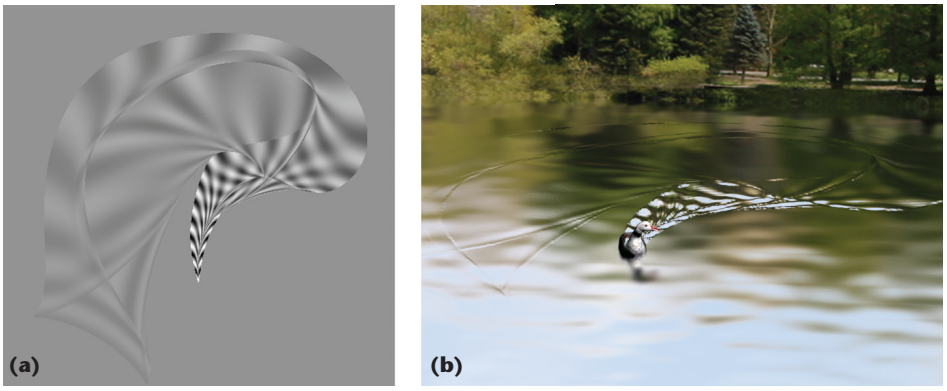
Water ballet

Now that we can create a wake pattern, let's put several swimmers together and see how the patterns interact.

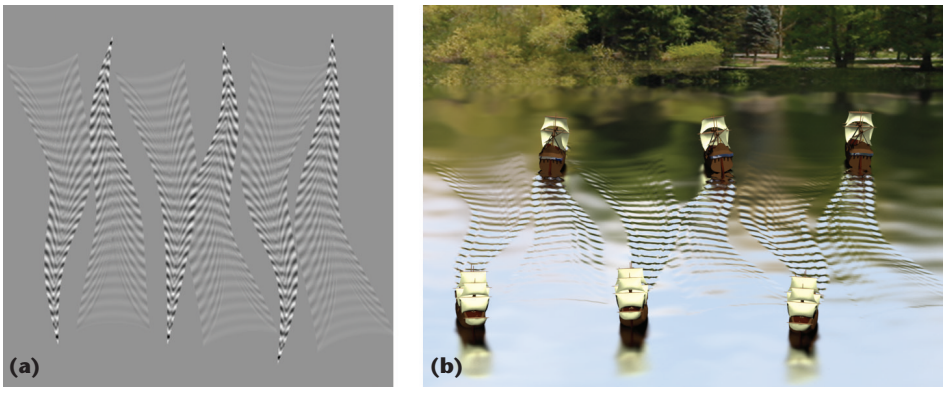
Sail away

It appears the local radio-controlled model galleon club has been practicing their close-formation steering skills. Figure 14a shows the wake pattern created by six galleons traveling in alternating directions. They're trying to steer a straight course, but the wind is pushing them around a little. Figure 14b shows the wake in 3D.

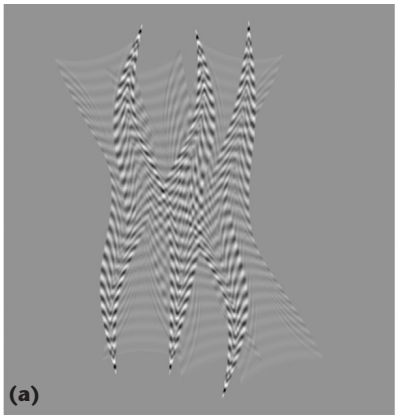
Because these pilots are so good, they decided to try another go at closer quarters. In Figure 15a we can see the waves created when the ships sail much closer to one another—the



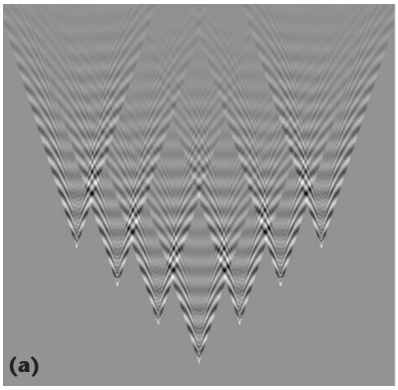
13 (a) A wave pattern for a duck swimming on a twisty path. (b) A synthetic image using the wave pattern of Figure 8a.

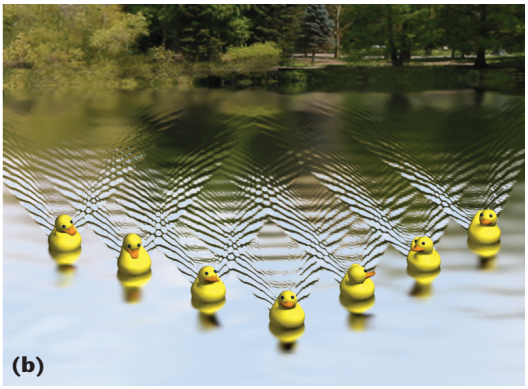


14 (a) A wave pattern for six boats on roughly parallel paths. (b) A synthetic image using the wave pattern of Figure 14a.

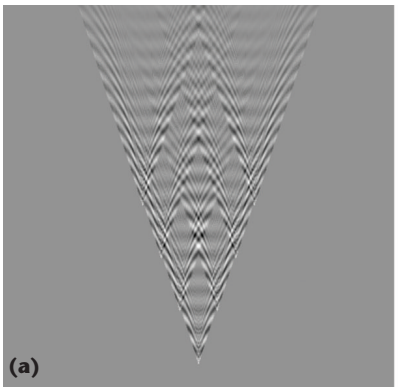


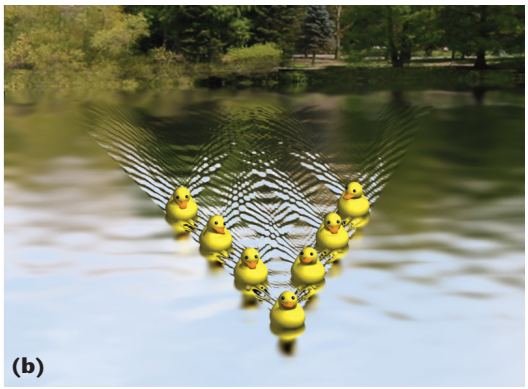
15 (a) A wave pattern for six boats on roughly parallel paths. (b) A synthetic image using the wave pattern of Figure 15a.





16 (a) A wave pattern for a V-shaped phalanx of seven ducks. Each is swimming outside the wake of the duck in front. (b) A synthetic image using the wave pattern of Figure 16a.





17 (a) A wave pattern for a V-shaped phalanx of seven ducks. Each is swimming just on the border of the wake of the duck in front. (b) A synthetic image using the wave pattern of Figure 17a.

wake patterns combine in beautiful ways. Figure 15b shows the result out on the water.

Rubber ducks unite

Everyone loves a rubber ducky in the bathtub. But it turns out that rubber ducks are social creatures and enjoy one another's company. In fact, rubber ducks like to swim in the same sort of V-shaped formations as other ducks.

In Figure 16a we can see the wake pattern created by a phalanx of seven rubber ducks swimming along in a V. Each duck is swimming outside of the Kelvin wave created by the duck ahead of it, so it's easy to see the starting point of each duck's wake. Figure 16b shows the results as they swim together creating the wake in Figure 16a.

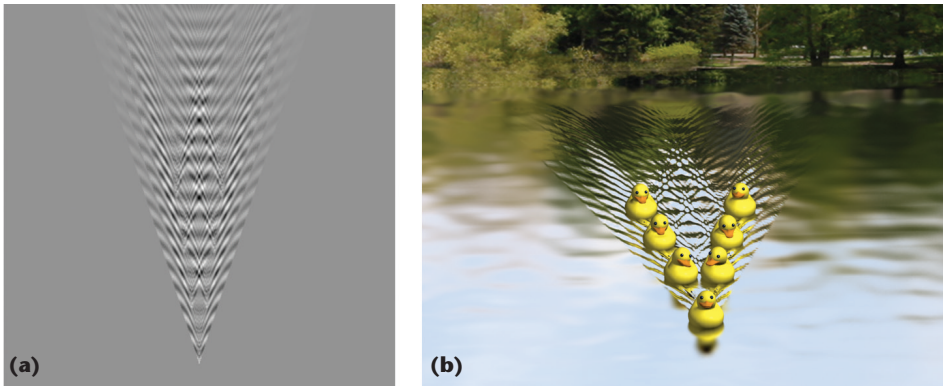
After a while the ducks move into closer formation. Each duck swims right on the border of the Kelvin wave created by the lead duck. Because all Kelvin waves share the same angle, the waves created by these ducks share the same outer border. Figure 17a shows the wake pattern, and Figure 17b shows how it looks out on the lake.

Then the ducks move in closer. When each duck starts to swim within the Kelvin wave of the duck ahead, they create the wake pattern of Figure 18a (next page) . Out on the water, this shows up as Figure 18b.

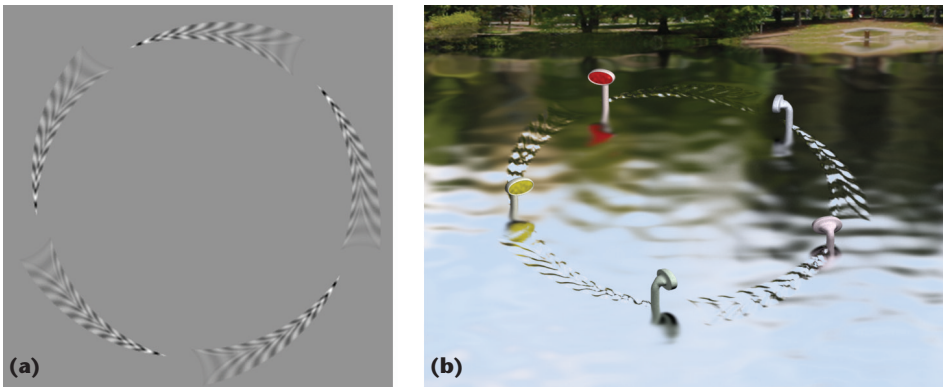
Up periscope

The recruiting poster exhorted young men to join the submariners and sail the seven underwater seas. What they didn't realize was that they would have to go on extensive training exercises out at the lake.

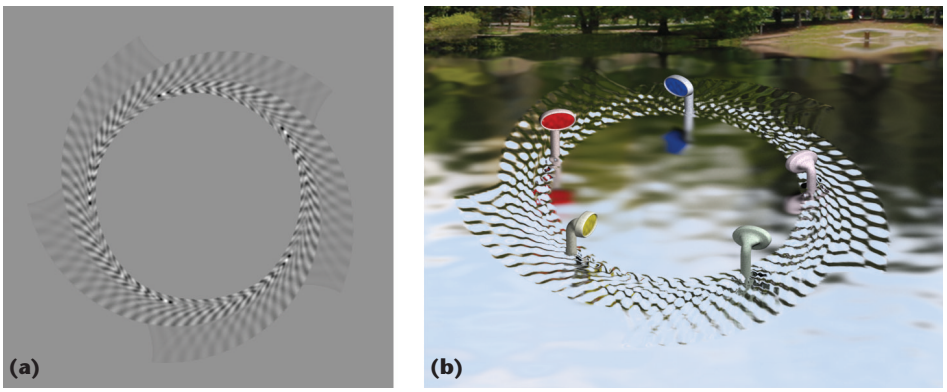
Their first exercise involved chasing each other in a



18 (a) A wave pattern for a V-shaped phalanx of seven ducks. Each is swimming inside the border of the wake of the duck in front. (b) A synthetic image using the wave pattern of Figure 18a.



19 (a) A wave pattern for five submarines chasing each other in a circle. Each one is pretty far behind the one in front of it. (b) A synthetic image using the wave pattern of Figure 19a.



20 (a) A wave pattern for five submarines chasing each other in a circle. They've swum further than in Figure 19 and each one is now in the wake of the one in front. (b) A synthetic image using the wave pattern of Figure 20a.

circle. Figure 19a shows the wake pattern created by the periscopes of five subs following each other. You can see that they just recently brought up their periscopes. Figure 19b shows this training exercise in progress.

After some time elapsed, the subs continued to chase each other and the wake patterns spread out more. You can see the wake pattern in Figure 20a, and the churning waves created by the subs in Figure 20b.

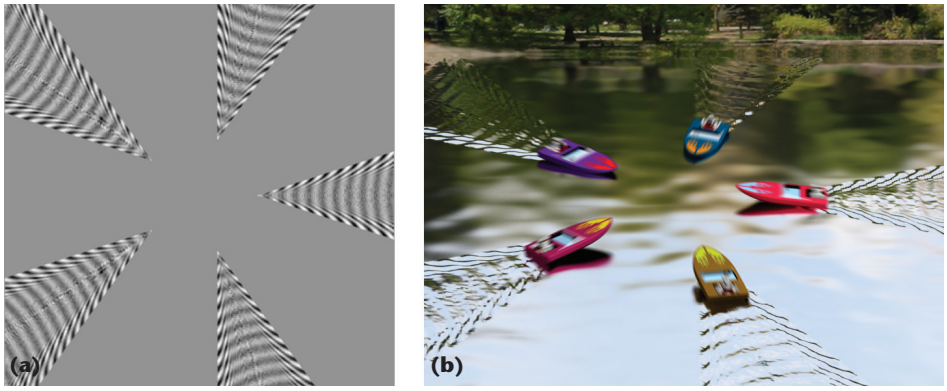
Toy boat, toy boat, toy boat

I was thinking about the Kelvin angle of about 19.5 degrees. Because the Kelvin angle is only half of the complete angle at the tip of the cone (as shown in Figure 3), the cone itself has an angle of about 39 degrees. This rattled around in my head for a while until I realized it was pretty close to 36 degrees.

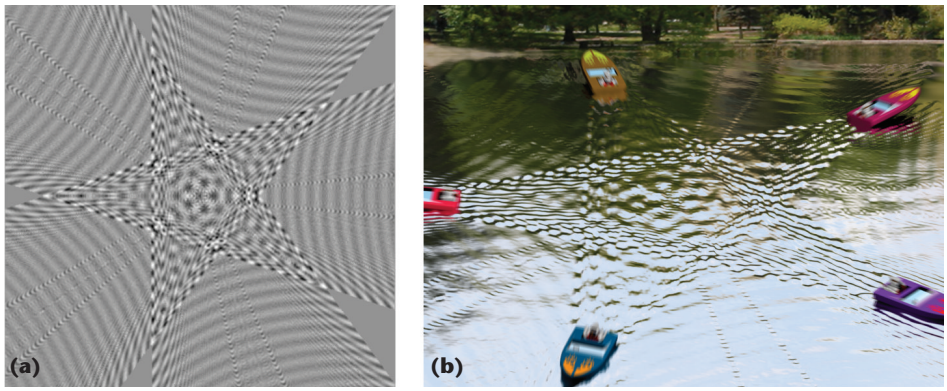
Why is that special? The angles at the tips of a five-pointed star are 36 degrees. I've never seen it reported before, but it seemed to me you could make a pretty cool star with Kelvin waves.

To check it out, I got five toy speedboats and sent them hurtling toward each other. Figure 21a shows the wake pattern during their initial approach, and Figure 21b shows the rendered 3D image out on the water.

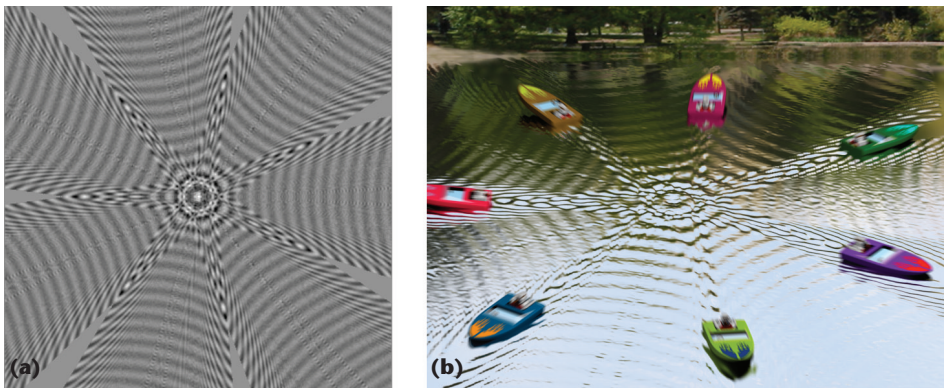
The boats raced toward one another at a high but constant speed. A massive collision was inevitable. But thanks to computer graphics, somehow they had managed to pass through a center point and out the other side unscathed! Figure 22a shows the wake pattern, and Figure 22b shows how the lake appears. In this figure, you can see small line structures in the oldest part of each boat's weight. They're visible here because I didn't use much damping in this figure. In real life, you can barely see these ripples behind a fast boat on calm water.



21 (a) Five speedboats heading toward a common point. (b) A synthetic image using the wave pattern of Figure 21a.



22 (a) Five speedboats that have passed through a common point. The geometry of Kelvin waves creates a near-perfect five-pointed star behind them. (b) A synthetic image using the wave pattern of Figure 22a.



23 (a) Seven speedboats that have passed through a common point. (b) A synthetic image using the wave pattern of Figure 23a.

Even though the angles are no longer magical, I created some other stars as well. One of the prettiest was a seven-pointed star. The seven-star wake is shown in Figure 23a, and the 3D version is in Figure 23b.

Returning to shore

One of the great pleasures of computer graphics is that we can play around with the world in ways that are difficult, expensive, or just plain impractical in real life. I don't know if there are professional duck wranglers out there, but even if there are, I doubt they can get ducks to swim in precision patterns. Certainly we can't get speedboats to pass through each other simultaneously without damage.

Yet we can do all of this with computer graphics. To me, the results are worth the effort, both for understanding nature and for the sheer fun of playing around with making interesting patterns. Don't listen to them when they tell you be quiet and don't make waves! ■

Acknowledgments

Thanks to Steven Drucker, Eric Haines, Christopher Rosenfelder, and Maarten van Dantzich for discussions and encouragement. The only real images in this column are the shoreline foliage, which I shot at Green Lake in Seattle. The ducks and galleons are public-domain 3D models, which I modified for use here. I computed the 3D wave models in this column with my Kelvin wave simulator. I assembled the scenes in Discreet's 3ds max and rendered all the 3D images with Splutterfish's rendering system Brazil r/s.

Readers may contact Andrew Glassner by email at andrew@glassner.com.