Simulated water flows, swirls, mixes, falls, refracts light, and interacts with objects in games, movie special effects, and commercials.

Modeling Water for Computer Animation

NICK FOSTER AND DIMITRIS METAXAS

AN ONGOING GOAL OF COMPUTER GRAPHICS is to provide tools not only for the artistic rendition of our physical world but for re-creating as much of our world in as realistic a way as possible. Until the late 1980s, this effort generally involved interpreting the way light interacted with the surfaces or volumes of objects. More recently, in applications ranging from feature films to computer games, the trend has been toward creating virtual worlds with increasingly realistic physics-based models.

The reasons for wanting greater physical realism vary depending on the application. For example, achieving greater physical realism in computer animation allows the animator to better integrate computer graphics elements. In movie special effects, it allows computer-generated image elements to be inserted into live action sequences with better overlap, so real and virtual components interact with the environment in the same way visually. Similarly, game developers want to provide as immersive an atmosphere as possible; physical interactions between players and objects in the environment need to be real-

3D FLUID DYNAMIC ENGINE (ARETÉ IMAGE SOFTWARE, A DIVISION OF ARETÉ ENTERTAINMENT) istic. For graphics researchers, physics-based modeling has also inspired special interest. Not only is it a fascinating topic, it leads to results that generate applause at conferences.

Modeling physics on a computer and visualizing the results using graphics techniques can lead to complex pictures as dazzling as the real-world phenomena they are intended to represent, especially for such fluid effects as the motion of water, fire, and smoke. It isn't surprising that a great deal of effort has been put into modeling such phenomena for computer graphics [2]. Here, we are concerned with modeling and animating water. Although modeling water for computer graphics is not a new research area, only recently have graphics researchers sought to take advantage of the huge body of literature on computational fluid dynamics in the interests of generating highly realistic animations.

Mechanical engineers and physicists have been modeling the behavior of liquids on computers for nearly 40 years [8]. However, their approach in general has focused on very specific goals, such as modeling the pressure around a newly designed ship hull as it undergoes various ocean conditions or calculating how the coolant in a nuclear reactor core flows around spherical rods. This focus on a few specialized engineering applications provides students of computer graphics with extremely useful techniques with which to achieve their own more general goals of modeling water so it looks visually convincing, moves realistically, and can be simulated on a desktop computer in a reasonable amount of time. To achieve this result, animation tool writers need to find ways to characterize the motion of a liquid so an animator can:

• Model and animate it quickly on a computer;
• Model how it interacts with traditional computer graphics environments;
• Allow the model to be manipulated in order to break the boundaries of real-world physics; and
• Create great-looking pictures.

These four goals are at the core of today's mainstream computer graphics research. That core says, take enough of the complexity of the actual physical model to get the realism the animator wants, but make sure the animator gets that realism in a reasonable amount of computational time while providing mechanisms to control the results. Even though these last two goals—pragmatic efficiency and control—are at odds with the aims of computational simulation, they can, when applied to numerical techniques, be invaluable tools for special effects and animation. Here, we describe a pragmatic approach to the problem for water simulation. It isn't enough to concentrate on the overall speed of the calculation. Sacrificing detail or user control to a theoretically more elegant solution makes for good research papers but does not, unfortunately, always translate into practice. In any case, once some basic rules are established, it is possible to use any of a number of available mathematical tools to solve the equations for the motion of water and apply them to the methodology described here.

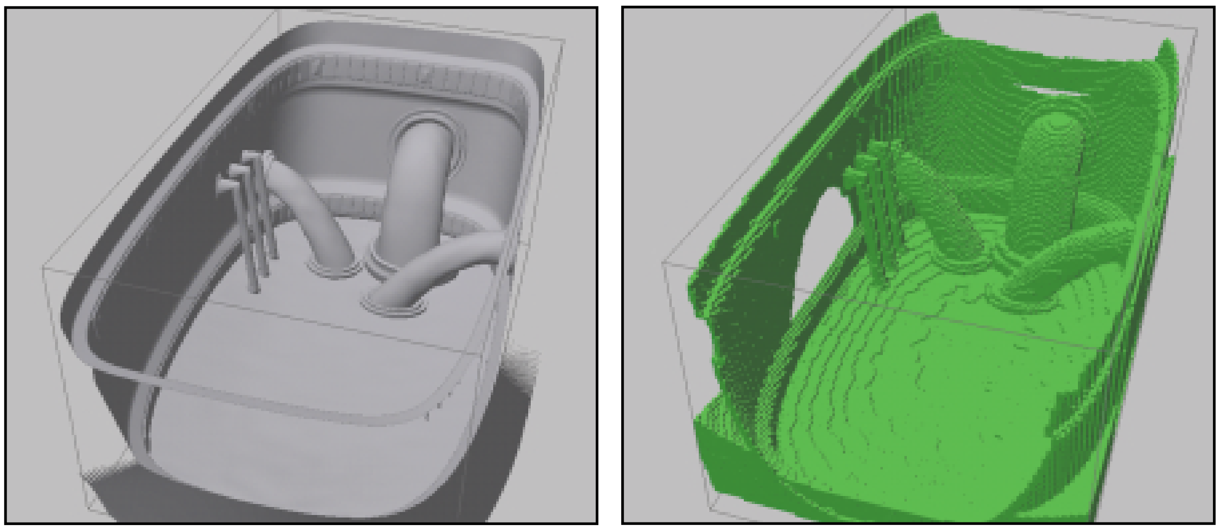
Rotational motion is the most important visual part of a liquid's motion and must be part of any realistic graphics model.

Look and Feel

Water has two qualities that give it its distinctive look. The first is purely visual; no matter how much a particular volume of water is distorting or splashing, the surface of that volume is always smooth and well defined. In addition, its surface has both refractive and reflective effects on light incident upon it. Combining these effects with absorption and scattering of the light as it passes through water produces a surface that is highly specular and distinctive.

The visual appearance of water in motion is even more complex. The motion of the water's surface is driven by interactions of the water molecules making up the entire volume. These interactions form what is called a "flickering cluster," whereby links between molecules (relatively weak hydrogen bonds) are constantly being broken and reformed as they slide over and around one another. This breaking and reforming of molecular bonds leads to a coupling between the velocity and pressure within the volume and also causes the natural viscosity of water.

Figure 1. (a) A polygonal model of a simple scene; (b) the cubic cell approximation of the same scene.



Consider a small element of water (much larger than molecular scale). At any moment in time, this element experiences forces on it from a number of physical factors. For example, it is pushed along by its neighbors while experiencing the effects of body forces, including gravity. This element is also viscous, so it tends to get caught up in the motion of its neighbors. Getting caught up in this way is called "drag" and leads to all the rotation we see as water moves around objects and mixes and flows. If graphics researchers can find a way to represent these effects with a computational model and then find a way to apply that model to traditional computer graphics environments, they can provide a computational tool for calculating the behavior of water as if it were part of a traditional computer graphics animation.

The Animation Environment

In computer graphics, objects are typically modeled as collections of parametric surfaces or 3D polygons. These objects can be of arbitrary complexity (within computational limits) and shape and can be defined in more intricate and accurate detail than that of a single pixel in the final image. To accurately take into account the interaction between a liquid and these complex surfaces would require a lot of computational effort that may not even be visible in the final image. This system is completely automatic and transparent to the animator. Therefore, the first task of the animation tool writer is to reduce this complexity, so the objects in the scene are more manageable. The level of detail reduction is driven by the rotational detail the animator wants from the final water motion. Once the animator decides on that level of detail, the system approximates the entire environment as a series of regular cubic cells.

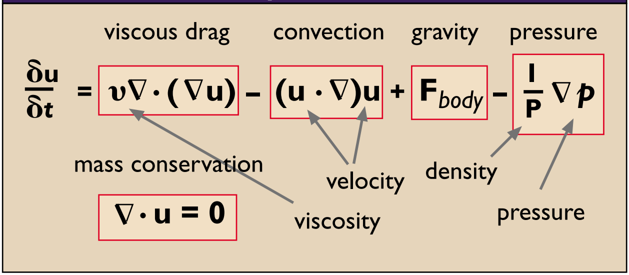
For each cubic cell in the grid, the system decides whether that cell contains a solid object or is empty and thus free to have water move through it. This extremely simple environment model is sufficient to produce realistic-looking interaction between water and static objects while being completely general in the sense that any environment can be modeled automatically, given a grid of appropriate size. Figure 1 shows an example of a polygonal environment together with its approximation.

At the center of each grid cell, a vector velocity value and a scalar pressure value are defined automatically. At each step during the simulation, these grid variables completely describe a smooth, continuous motion field. This field evolves over time as the method proceeds. Thus, although the resolution of the cubic grid remains relatively coarse, the resulting water motion appears visually smooth and continuous. By "discretizing" the environment in this way, traditional methods for solving the equations of motion for an incompressible fluid can be applied in a general and straightforward manner to computer graphics environments.

The Physics Model

The goal of physics-based graphics modeling tools is a compromise between getting visual realism and reasonable computational times. To reach it, modeling researchers need to find the smallest subset of motion components contributing to the overall motion the animator would ultimately want. By

Figure 2. The Navier-Stokes equations for incompressible fluid flow.



$$\frac{\delta \mathbf{u}}{\delta t} = \boxed{\upsilon \nabla \cdot (\nabla \mathbf{u})} - \boxed{(\mathbf{u} \cdot \nabla)\mathbf{u}} + \boxed{\mathbf{F}_{body}} - \boxed{\frac{1}{\rho} \nabla p}$$

$$\boxed{\nabla \cdot \mathbf{u} = 0}$$

examining how large bodies of real water interact and move, the animator can write the following short list of effects the system would have to account for:

Momentum. Water obeys the laws of energy and momentum conservation.

Gravity. Most water waves are the result of interaction between gravity and pressure.

Viscous drag. This is the key to producing the realistic flow of water—the component of motion leading directly to self-propagating rotation. Rotational motion is the most important visual part of a liquid's motion and must be part of any realistic graphics model.

Pressure. While not obvious, pressure plays an important role in the behavior of the surface of a volume of water. Pressure waves internal to the volume (acoustic waves) can travel much faster than the surface waves we are all familiar with. Thus, much of what we see at the surface has been influenced by subtle (and sometimes not-so-subtle) changes in pressure within the volume.

Other motion elements, including turbulence and surface tension, are vital to water behavior under certain conditions, but we assume that, at a human scale, their effects are dominated by those just listed. Conveniently, there is a physics model that includes all of them. In the early 1800s, the French engineer Claude Navier and the Irish mathematician George Stokes independently derived a set of equations describing all the forces acting on a small quantity of liquid [7, 9]. The equations, shown in Figure 2, are really quite elegant; even if you aren't familiar with the symbols, what they describe is straightforward. Each of the four parts of the figure account for one of the motion components just described. The physics the equations describe are changes in velocity and pressure over time—the same variables defined within our environment grid. If we can somehow break the equations down into a form that allows them to be applied to the grid, a simulator would have a way of calculating exactly how water would move if it were suspended within the grid.

To adequately apply the equations to our approximated computer graphics environment, two concepts are needed. The first allows us to apply the equations directly to the grid, so we can calculate how water moves through it. The second then provides a mechanism for representing the water within the grid. These issues have all been dealt with in the past in the scientific field of computational fluid dynamics, so we don't have to look far for useful techniques.

Making Water Move

The literature for computational fluid dynamics is full of many different ways to solve the Navier-Stokes equations, each involving advantages and disadvantages. (A good introduction to these methods is available in [4].) It's well beyond the scope of this article to discuss any of them in detail, but worth pointing out are the salient goals of the methods in general with respect to our problem.

Robustness. This is very important. Software engineers writing animation tools want to help animators simulate water for special-effects purposes. The very nature of animation means there are likely to be extremely violent things going on during the simulations. The computational method must therefore adjust dynamically to accommodate whatever it is the animator wants. It isn't reasonable to assume the animator has any knowledge of numerical issues.

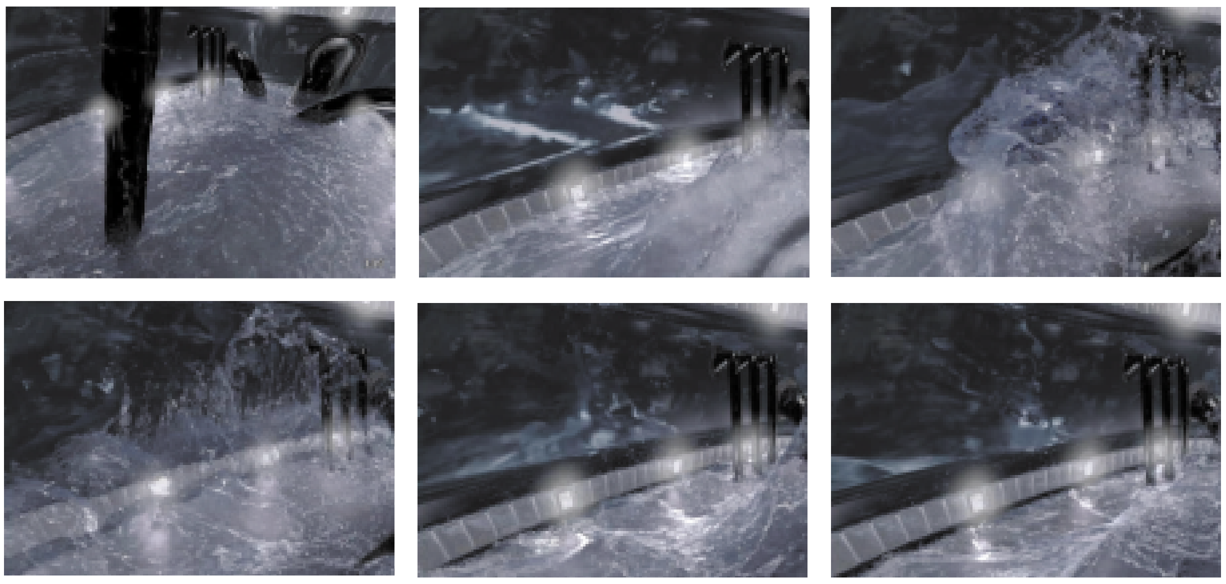
Generality. The simulation method has to be applicable to a wide range of different problems at a range of different scales in a way that's transparent to the animator.

Flexibility. The mathematical method used in the simulation needs to be extensible, so the simulator can handle interaction between liquids and moving objects without loss of generality or robustness.

In practice, as long as the animation tool solves the equations correctly, the choice of solution method tends to be just a trade-off between these issues of robustness, generality, and flexibility. For the purposes of this article, we value flexibility most of all and use a finite-difference method to approximate equations like those in Figure 2 as a Taylor series of much simpler elements [6]. For computer graphics, finite differences have distinct advantages. The most useful is that a finite-difference system is sensitive only to local change, allowing the simulator to directly modify velocity and pressure during a simulation. Thus, effects animators get all the motion control they want at the cost of loss of control over computation time (the disadvantage of finite differences). At the low-grid resolutions we describe here,

]

Figure 3. A sequence of images from an animation of a heavy object falling into a tank of water.



this is a less-serious trade-off than it might seem.

The basic approach to a finite-difference solution to the Navier-Stokes equations first appeared 30 years ago [5]. Essentially, the equations are broken down, or "discretized," so they can be applied individually to each cell in the grid. Thus, given an existing velocity field, it's possible to calculate how the water is going to move during a small time increment. The animation can then be described by simulating the water motion over successive increments and taking snapshots (frames) of its position every 1/24th of a second. This timestep length (1/24) produces the standard animation playback rate of 24 frames per second.

Although the grid we have just described is relatively coarse, the water motion has to be visually smooth at all times. In real-world water-tank experiments, fine powder can be dropped into the water so experimenters can see exactly how the volume is moving. We achieve the same thing within our simulated water environment by distributing particles within the grid to delineate the location of the water. Each particle represents a small volume of water but takes no part in the calculation other than to signal which cells contain liquid. Each particle is then moved according to the local water velocity, as determined by interpolation from within the grid.

Applying realistic forces should produce realistic-looking motion.

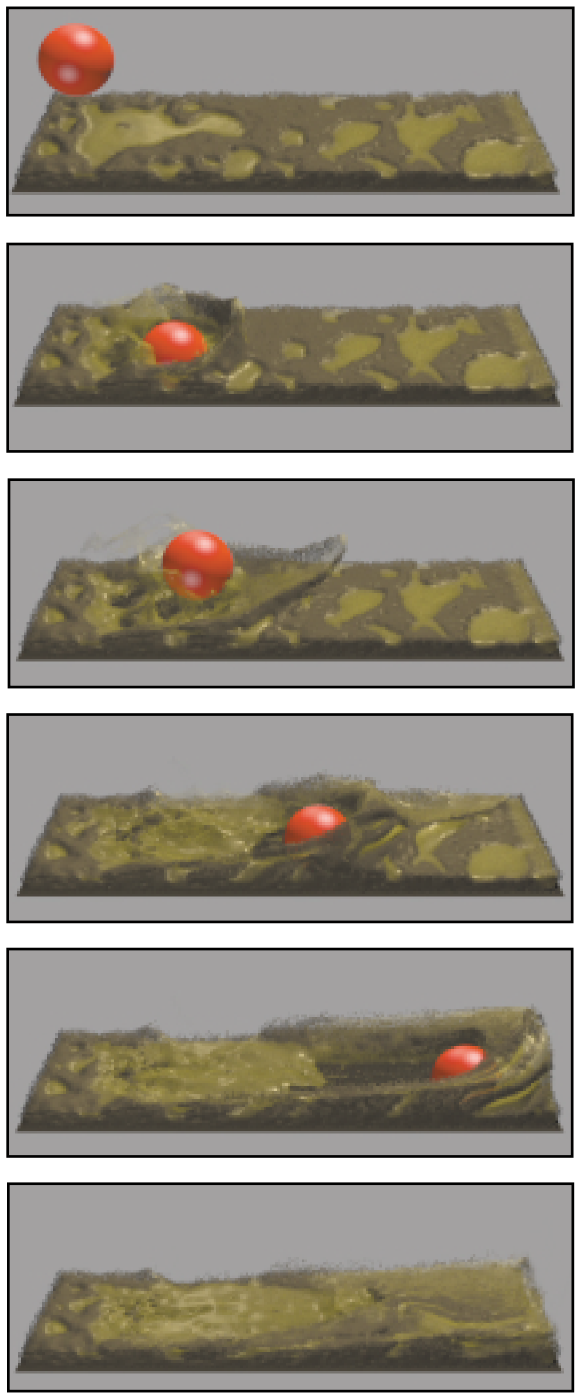
Particles are an ideal way of defining a highly dynamic volume that can constantly and arbitrarily change shape without introducing additional computational overhead. They are also one of the basic primitives of computer graphics, and a lot of work has been done regarding their use for surface and volume visualization.

Using these techniques, we produced a robust, relatively fast way of taking an arbitrary scene described using standard computer graphics primitives and calculating how a volume of water would interact with itself as well as with the environment around it. These results are by no means scientifically accurate, but they do contain all the components of motion giving a human viewer the visual cues that we are observing real water behavior. The next step is to apply this simulation mechanism to the general animation problem, answering how effects animators can make the water behave exactly as they would like it to behave.

Controls for Computer Animation

In a computer animation environment, only one type of control counts, a curve. Although there are many different mathematical ways to describe the

Figure 4. A sequence of test images showing the interaction between an object and liquids of different viscosities, including water and mud.



path of a curve, including Bezier, Bspline, and Linear, they all amount to the same thing. Any object (an arm or an elephant, a hand or a ball) follows a track through space defined by a set of time-varying curves. There's no notion of dynamics or interaction other than that defined by hand by the animator. But this subjective definition conflicts directly with a dynamic simulation of water, which is nondeterministic (at least to mere mortals) and obeys an inflexible set of rules. To be useful, a general tool needs to combine the Navier-Stokes simulator with a notion of a defined curve, that is, it should drive the overall motion of the liquid using a deterministic curve. Given the choices we made earlier to use a (fixed) grid together with finite differences means that incorporating control via a deterministic curve is quite straightforward.

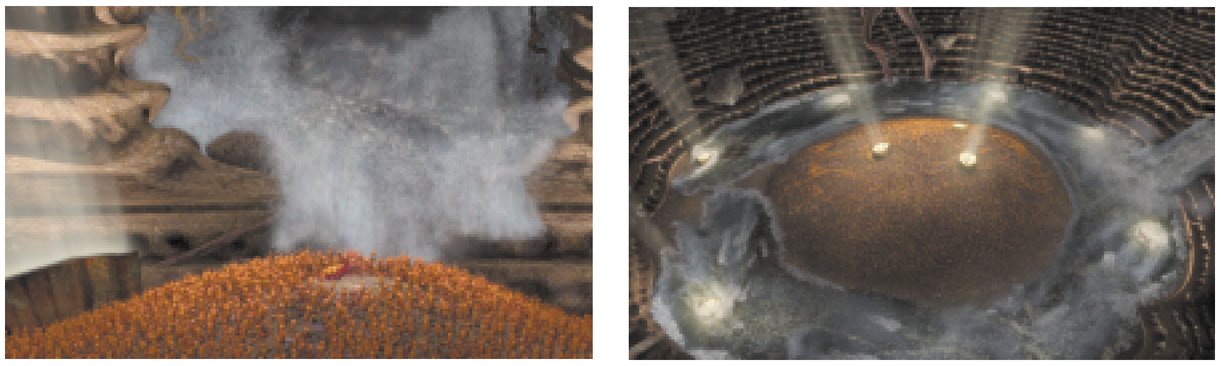
The system has two types of curves within the water environment, each representing a different form of control—one the path of a moving, solid object, the other a path we want the water to take. In the case of a moving solid object, the water should splash and flow away from and around the object, whereas water moving in the path we want it to take should be drawn along the path. In both cases, however, the overall motion should always look real. Therefore, applying realistic forces should produce realistic-looking motion. Applying unrealistic forces should produce motion that still has a realistic fluid component superimposed on top of it.

The motion of objects as they move through water is a well-understood topic. Given the differences in motion between object and liquid, together with the angle and direction of the object's surface, the system calculates the forces exerted by the object on the water. These forces can then be applied directly to the environment grid. The water in this grid behaves exactly as if an object were moving through it. Despite the grid's relatively low resolution, the approach works surprisingly well (see Figure 3).

More care is required when applying the second type of curve, because the forces are inherently nonrealistic. We can't just apply the forces directly, just because we want a component of the motion to look realistic. The key to producing a realistic-looking animation is to encourage the water to follow the curve by adjusting the local pressure field, effectively fooling the liquid into thinking its neighbors are already following the curve, so it gets dragged along naturally. The system can then direct the water's motion by making pressure adjustments according to the differences between actual and desired velocity. In this way, the Navier-Stokes equations act to smooth out whatever motion is applied and keep it looking fluid-like.

With these two control mechanisms in place, the water-simulation system can be tied into a generic computer animation system. Users need not be familiar with fluid dynamics concepts nor with any

Figure 5. Images from *Antz* showing a range of water effects created using the system described in the article.



of the physics involved. The result is a general-purpose tool for creating water-based special effects.

Rendering

The cloud of particles representing the distribution of water in the environment needs to be rendered in such a way as to make it seem as if the particles are enclosed in a continuous surface. The standard (computer graphics) way to do this is to make each particle represent a spherical density function and define the aggregate surface implicitly from the density field created by the sum of the particles' fields [1], as in Figures 3 and 4. The particles can be rendered directly if there are enough of them. This method was used (in part) to render all of the images for the final sequence of Pacific Data Image's 1998 feature film *Antz* (see Figure 5).

The combination of low-resolution water simulator and computer-animation curve control also leads to an unprecedented capability for special effects. In the past, computer-generated special effects involving water were limited to surface-wave approaches combined with particle systems for splashing and flow. Now it's possible to produce large-scale interaction with characters and objects while maintaining traditional animation control techniques. The system is physics-based, so it can be scaled to handle the interaction among types of liquids, as in Figure 4. In addition, it can be used to provide a realistic component of motion to any effect involving fluids, including wind on grass and trees, smoke, and steam, and even, to some extent, flames on the surrounding atmosphere.

Conclusion

This system for modeling and animating the interaction between realistic-looking water and a computer graphics environment draws on the science of computational fluid dynamics to calculate an approximate velocity field describing the time-varying motion of a volume of water. It combines classical mechanics and computer graphics primitives to provide both motion control and generality for use as a basic animation tool. It is not a numerically precise simulator and was not designed for mechanics applications. The method, as described, has been used at a feature and effects animation studio (Pacific Data Images) for a number of projects, including the finale to *Antz*, proving itself to be a valuable special-effects tool. C

References

1. Bloomenthal, J., Ed. *Introduction to Implicit Surfaces*. Morgan Kaufman, San Francisco, 1997.
2. Foster, N. *Modeling and Animating Fluid Phenomena for Computer Graphics and Special Effects*, Ph.D. dissertation, University of Pennsylvania, Philadelphia, 1997.
3. Foster, N. and Metaxas, D. Realistic Animation of Liquids, Graph. Mod. Image Process. 58, 5 (1996), 471–483.
4. Fletcher, C. *Computational Techniques for Fluid Dynamics*. Springer Verlag, Sydney, 1990.
5. Harlow, F. and Welch, J. Numerical calculation of time-dependent viscous incompressible flow. *Phys. Fluids 8*, 12 (Dec. 1965), 2182–2189.
6. Mitchell, A. and Griffiths, D. *The Finite Difference Method in Partial Differential Equations,* John Wiley & Sons, Inc., New York, 1980.
7. Navier, L. Memoire sur les lois du mouvement des fluides. *Memoir. de l'Academ. Royale des Sci. 6* (1827).
8. Shaw, C. *Using Computational Fluid Dynamics.* Prentice Hall, London, 1992.
9. Stokes, G. On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids. *Transact. Cambridge Philos. Soc. 9* (1851).

Nick Foster (nickf@pdi.com) is a senior engineer, research and development, in Pacific Data Images in Palo Alto, Calif.
Dimitris Metaxas (dnm@central.cis.upenn.edu) is an associate professor in the Department of Computer and Information Science at the University of Pennsylvania in Philadelphia and director of Penn's Vision, Analysis, and Simulation Technologies Laboratory.