



Rapid, Stable Fluid Dynamics for Computer Graphics

Michael Kass and Gavin Miller

Advanced Technology Group
Apple Computer, Inc.
20705 Valley Green Drive
Cupertino, CA 95014

ABSTRACT

We present a new method for animating water based on a simple, rapid and stable solution of a set of partial differential equations resulting from an approximation to the shallow water equations. The approximation gives rise to a version of the wave equation on a height-field where the wave velocity is proportional to the square root of the depth of the water. The resulting wave equation is then solved with an alternating-direction implicit method on a uniform finite-difference grid. The computational work required for an iteration consists mainly of solving a simple tridiagonal linear system for each row and column of the height field. A single iteration per frame suffices in most cases for convincing animation.

Like previous computer-graphics models of wave motion, the new method can generate the effects of wave refraction with depth. Unlike previous models, it also handles wave reflections, net transport of water and boundary conditions with changing topology. As a consequence, the model is suitable for animating phenomena such as flowing rivers, raindrops hitting surfaces and waves in a fish tank as well as the classic phenomenon of waves lapping on a beach. The height-field representation prevents it from easily simulating phenomena such as breaking waves, except perhaps in combination with particle-based fluid models. The water is rendered using a form of caustic shading which simulates the refraction of illuminating rays at the water surface. A wetness map is also used to compute the wetting and drying of sand as the water passes over it.

CR Categories and Subject Descriptors: I.3.7: [Computer Graphics]: Graphics and Realism; Animation; G.1.8: [Mathematics of Computing]: Partial Differential Equations; I.6.3 [Simulation and Modeling]: Applications.

Additional Keywords and Phrases: Wave equation, fluid dynamics, flow, finite-difference, height-field, caustic.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

INTRODUCTION

The problem of realistically modeling scenes containing water has captured the attention of a number of computer-graphics researchers in recent years[1; 2; 3; 4; 5]. The omnipresence of water as well as the complexities and subtleties of its motion have made it an attractive subject of study. Yet existing computer-graphics models of water motion adequately cover only a very small range of interesting water phenomena. Among other effects, they fail to account for wave reflections, net transport of water and boundary conditions with changing topology. A computationally inexpensive method of simulating these phenomena will be presented here. Based on solving a partial-differential equation on the surface of a height-field, the method is easy to implement and very stable. The approximations involved may not be suitable for high-precision engineering applications, but they produce pleasing animation with little effort.

Many popular methods for modeling water surfaces work well for producing still images, but are unsuitable for animation because they do not include realistic models for the evolution of the surface over time. Examples of these techniques include stochastic subdivision [6] and Fourier synthesis [5]. Other techniques work well only in large bodies of water away from boundaries [7; 1; 8]. Recently, the realism of water modeling in computer graphics was substantially improved by three papers [2; 3; 4] that took into account refraction due to changing wave velocity with depth. In each case, specialized methods based on tracking individual waves or wave-trains were used to avoid the need to directly solve a differential equation. These papers deal adequately with waves hitting a beach, but they leave a wide range of water phenomena unexplored. None of the papers includes simulations of reflected waves. In addition, the underlying model in each case is that particles of water move in circular or ellipsoidal orbits around their initial positions, so there can be no net transport or flow. Finally, none of the papers considers situations in which the boundary conditions change through time altering the topology of the water — for example a wave pushing water up over an obstacle and down the other side to create a puddle. It appears to be very difficult to deal with these phenomena efficiently by tracing waves.

Two alternatives to tracing the propagation of waves or wave-trains exist. One is to simulate the fluid by the interaction of a large number of particles [9; 10], and the other is to directly solve a partial differential equation describing the fluid dynamics [11; 12; 13]. Both have been used by hydrodynamicists to create iterative simulations of fluid flow. The problem is that a truly accurate simulation of fluid mechanics usually requires computing the motion throughout a volume. This means that the amount of computation per iteration grows at least as the cube of the resolution. If there are linear

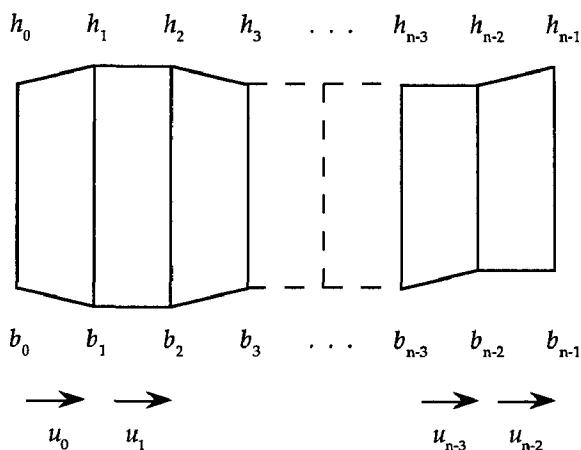


Fig. 1: Discrete two-dimensional height-field representation of the water surface h , the ground bottom b , and the horizontal water velocity u .

systems to be solved at every iteration, the computational cost can grow even faster. In addition, the number of iterations required may grow as the resolution is increased. As a consequence, accurate simulation of fluid mechanics is typically reserved for vectorized supercomputers or very highly parallel machines.

For the purposes of animation, accuracy is much less important than stability and speed. An animator using techniques of physical simulation will typically have to experiment with a number of different conditions of a simulation before achieving satisfying motion. If the experiments take too much time or if the numerical methods become unstable, the process can become excruciating.

Here, we examine the differential equation approach with the goal of constructing the fastest stable simulation which yields a wide range of convincing motion. We begin by considering a very simplified subset of water flow where the water surface can be represented as a height field and the motion is uniform through a vertical column. This subset of water flow is representative of a variety of non-turbulent shallow-water phenomena. Under these conditions, we can approximate the equations of motion of the water in terms of a grid of points on a height-field. The amount of computation can then be proportional to the number of samples on the surface of the water which varies as the square of the resolution instead of the cube.

Integration of the partial-differential equations is done with an alternating-direction implicit technique [14]. The result is a very stable integration scheme which is also very fast. Stability derives from the use of an implicit integration scheme; speed derives from the tridiagonal structure of the required linear systems which are solvable in linear time. Because of the stability, the time-step of the solution can be made equal to the frame time of the animation in most cases.

SHALLOW WATER EQUATIONS

In lieu of simulating the full Navier-Stokes equations of fluid flow, we begin with a vastly simplified set of equations which has been widely used for shallow water [15; 16; 17]. The simplification arises from three approximations. The first approximation is that the water surface is a height field. This, of course, has some obvious limitations. The water cannot splash and waves cannot break. However, so long as the forces on the water are sufficiently gentle, the height-field as-

sumption will not introduce error. The second assumption is that the vertical component of the velocity of the water particles can be ignored. Once again, the limitations of this assumption are fairly clear. If a disturbance creates very steep waves on the water surface, the model will cease to be accurate. The third assumption is that the horizontal component of the velocity of the water in a vertical column is approximately constant. If there is turbulent flow or unusually high friction on the bottom, this assumption will break down. Nonetheless, the experience of hydrodynamicists suggests that this is a very useful approximation to phenomena ranging from the effect of a single rain drop to the refraction of waves in a sea port.

For simplicity, we begin with a height-field curve in two dimensions. Later, the same techniques will be extended to a height-field surface in three dimensions. Let $z = h(x)$ be the height of the water surface and let $z = b(x)$ be the height of the ground. If $d(x) = h(x) - b(x)$ is the water depth and $u(x)$ is the horizontal velocity of a vertical column of water, the shallow water equations that follow from the above assumptions[16; 17] can be written as follows:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + g \frac{\partial h}{\partial x} = 0 \quad (\text{eq. 1})$$

$$\frac{\partial d}{\partial t} + \frac{\partial}{\partial x} (ud) = 0 \quad (\text{eq. 2})$$

where g is the gravitational acceleration. Eq. 1 expresses Newton's law $F=ma$ while eq. 2 expresses the constraint of volume conservation. Note that even with the above three simplifying assumptions, the resulting differential equations are non-linear. A further simplification which is often used is to ignore the second term in eq. 1 and linearize around a constant value of h . This will be reasonable if the fluid velocity is small and the depth is slowly varying. The resulting equations are then:

$$\frac{\partial u}{\partial t} + g \frac{\partial h}{\partial x} = 0 \quad (\text{eq. 3})$$

$$\frac{\partial h}{\partial t} + d \frac{\partial u}{\partial x} = 0 \quad (\text{eq. 4})$$

If we differentiate eq. 3 with respect to x , then differentiate eq. 4 with respect to t and finally substitute for the cross-derivatives, we end up with

$$\frac{\partial^2 h}{\partial t^2} = gd \frac{\partial^2 h}{\partial x^2} \quad (\text{eq. 5})$$

which is the one-dimensional wave equation with wave velocity \sqrt{gd} . While this degree of simplification is suspect for many engineering purposes, our experience suggests that the resulting equations are quite adequate for a wide range of animation applications.

DISCRETIZATION

In order to solve eq. 5, we need to construct a discrete representation of the continuous partial-differential equation. There are two established techniques for doing so. The first is the finite-difference technique where the continuous functions are represented by a collection of samples. The second is the finite-element technique where the continuous functions are represented as the sum of a collection of continuous basis functions. Here, the finite-difference technique works particularly well because of the simple height-field representation. The resulting algorithm is very easy to implement and the

linear systems involved are tridiagonal.

Figure 1 shows the discrete representation of the height-field in two dimensions. Note that the samples for u lie half-way in between the samples of h . After experimenting with a number of finite-difference approximations to equations 3 and 4, the most stable version we have found is

$$\frac{\partial h_i}{\partial t} = \left(\frac{d_{i-1} + d_i}{2\Delta x} \right) u_{i-1} - \left(\frac{d_i + d_{i+1}}{2\Delta x} \right) u_i \quad (\text{eq. 6})$$

$$\frac{\partial u_i}{\partial t} = \frac{-g(h_{i+1} - h_i)}{\Delta x} \quad (\text{eq. 7})$$

where Δx is the separation of the samples along the x direction. Putting the above two equations together, we get

$$\begin{aligned} \frac{\partial^2 h_i}{\partial t^2} &= -g \left(\frac{d_{i-1} + d_i}{2(\Delta x)^2} \right) (h_i - h_{i-1}) \\ &\quad + g \left(\frac{d_i + d_{i+1}}{2(\Delta x)^2} \right) (h_{i+1} - h_i) \end{aligned} \quad (\text{eq. 8})$$

which is a discrete approximation to eq. 5.

INTEGRATION

The finite differences convert the partial-differential equation into an ordinary differential equation involving h and its time derivatives. The remaining problem is to solve the ordinary differential equation. While there are a number of possible choices of solution method, the wave equation is a notoriously bad example for explicit differential equation methods such as Euler's method or Runge-Kutta integration. As the wave velocity approaches one sample per iteration, explicit methods tend to diverge very rapidly. Since the wave speed is proportional to the square-root of the depth, an ordinary explicit method would have to use a time-step appropriate for the deepest water in the model. Implicit methods, on the other hand, do not suffer from these difficulties.

For simplicity, we use a first-order implicit method which appears to be perfectly adequate. Let $h(n)$ to denote h at the n th iteration and let dots denote differentiation with time. Then the first-order implicit equations can be written

$$\frac{h(n) - h(n-1)}{\Delta t} = \dot{h}(n) \quad (\text{eq. 9})$$

$$\frac{\dot{h}(n) - \dot{h}(n-1)}{\Delta t} = \ddot{h}(n) \quad (\text{eq. 10})$$

Note that the right-hand sides of these equations are evaluated at time n which corresponds to the end of the iteration rather than time $n-1$ which corresponds to the beginning of the iteration. This is what makes the iteration implicit and stable. Rearranging the above, we get

$$\begin{aligned} h(n) &= h(n-1) + \Delta t \dot{h}(n-1) \\ &\quad + (\Delta t)^2 \ddot{h}(n) \end{aligned} \quad (\text{eq. 11})$$

$$h(n) = 2h(n-1) - h(n-2) + (\Delta t)^2 \ddot{h}(n) \quad (\text{eq. 12})$$

$$\begin{aligned} h_i(n) &= 2h_i(n-1) - h_i(n-2) \\ &\quad - g(\Delta t)^2 \left(\frac{d_{i-1} + d_i}{2(\Delta x)^2} \right) (h_i(n) - h_{i-1}(n)) \\ &\quad + g(\Delta t)^2 \left(\frac{d_i + d_{i+1}}{2(\Delta x)^2} \right) (h_{i+1}(n) - h_i(n)) \end{aligned} \quad (\text{eq. 13})$$

We are still left with non-linear equations because d depends on h . In order to solve these equations rapidly, we need a final linearization. Once again there are several possible choices, but a particularly well-behaved linearization is to regard d as a constant during the iteration. This means that the wave velocity is fixed as a function of x . It limits the non-linearities to changing the wave velocities in-between iterations and virtually ensures that the iteration will not diverge. With this linearization the next value of h can be calculated from previous values with the symmetric tridiagonal linear system

$$Ah_i(n) = 2h_i(n-1) - h_i(n-2) \quad (\text{eq. 14})$$

where the matrix A is given by

$$A = \begin{pmatrix} e_0 & f_0 & & & & & \\ f_0 & e_1 & f_1 & & & & \\ & f_1 & e_2 & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & e_{n-3} & f_{n-3} & \\ & & & & f_{n-3} & e_{n-2} & f_{n-2} \\ & & & & & f_{n-2} & e_{n-1} \end{pmatrix} \quad (\text{eq. 15})$$

and the elements of A are as follows:

$$e_0 = 1 + g(\Delta t)^2 \left(\frac{d_0 + d_1}{2(\Delta x)^2} \right)$$

$$e_i = 1 + g(\Delta t)^2 \left(\frac{d_{i-1} + 2d_i + d_{i+1}}{2(\Delta x)^2} \right) \quad (0 < i < n-1)$$

$$e_{n-1} = 1 + g(\Delta t)^2 \left(\frac{d_{n-2} + d_{n-1}}{2(\Delta x)^2} \right)$$

$$f_i = -g(\Delta t)^2 \left(\frac{d_i + d_{i+1}}{2(\Delta x)^2} \right) \quad (\text{eq. 16})$$

Note that right-hand side of eq. 14 can be regarded as an extrapolation of the previous motion of the fluid surface. Some interesting effects are possible by slightly changing the extrapolation. In particular, if the equation is changed to be

$$\begin{aligned} Ah_i(n) &= h_i(n-1) \\ &\quad + (1-\tau)(h_i(n-1) - h_i(n-2)) \end{aligned} \quad (\text{eq. 17})$$

then τ introduces some damping in the extrapolation. If

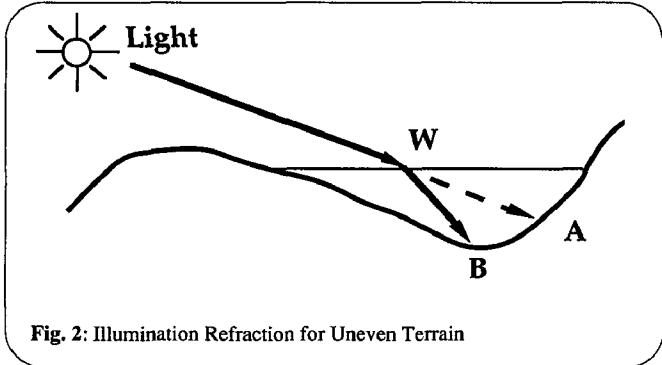


Fig. 2: Illumination Refraction for Uneven Terrain

$\tau = 0$, then it reduces to eq. 14, but if τ is between zero and one, it will make the waves damp out over time. The visual effect is that of a viscous fluid.

There is one further subtlety of importance in the two-dimensional case. Even though eq. 14 was derived from eq. 6 which specifies conservation of volume, there is no guarantee that the results of the iteration will precisely conserve volume. The primary cause of departures from volume-conserving behavior is that the iteration may leave $h_i < b_i$ for some index i . To compensate for this negative volume, the iteration will create excess positive volume elsewhere. While the effect is small, it can accumulate over time and create substantial drift. If the entire surface acquires a small net upwards velocity it will very quickly create noticeable amounts of water. To combat this effect, the following simple projection appears to be adequate. After each iteration, find the connected pieces of the fluid. This can be done by scanning the h and b vectors in order and testing whether $h_i < b_i$. For each connected piece of the fluid, calculate the old volume and the new volume. If the new volume is different, distribute the difference uniformly over the samples in the connected region.

We can now state the entire algorithm for the two-dimensional case in some detail:

```

Begin by specifying  $h(0)$ ,  $h(1)$  and  $b$ .
Loop for  $j$  starting at 2 incrementing by one.
  If there are net sources or sinks of water, add or subtract
    the amounts from the current and last values of  $h$ .
  Calculate  $d$  from  $h(j-1)$  and  $b$ . If  $h_i < b_i$  then  $d_i = 0$ .
  Calculate the new value of  $h$  from  $h(j-1)$  and  $h(j-2)$ 
    using eq. 14.
  Adjust the new value of  $h$  to conserve volume as above.
  If  $h_i < b_i$  for some index  $i$ , set  $h_i(j)$ 
    and  $h_i(j-1)$  to  $b_i - \epsilon$ .
The resulting value of  $h$  is  $h(j)$ .

```

While there are a number of possible refinements, this is the basic version of the two-dimensional case. It can be implemented in one to two pages of very efficient, straightforward C code.

THREE DIMENSIONS

Height fields in two dimensions are interesting, but moving to three dimensions opens up a much wider range of possibilities. Fortunately, the three-dimensional equations can be approximated by a series of two-dimensional equations, so the complexity does not increase radically. The basic wave equation for water in three dimensions is the same as the two-dimensional case except that the second derivative of h with respect to x is replaced with the Laplacian.

$$\frac{\partial^2 h}{\partial t^2} = gd \left(\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} \right) = gd\nabla^2 h \quad \text{eq. 18}$$

In order to solve the equations in three dimensions, we rely on the alternating-direction method[14]. The basic idea of the method is to take eq. 18 and split the right-hand side of it into the sum of two terms, one of which is independent of y and the other of which is independent of x . We then divide the iteration into two sub-iterations. In the first sub-iteration, we replace the right-hand side of eq. 18 with the first term, and in the second sub-iteration, we replace the right-hand side of eq. 18 with the second term. More specifically, in the first sub-iteration, we solve the equation

$$\frac{\partial^2 h}{\partial t^2} = gd \frac{\partial^2 h}{\partial x^2} \quad \text{eq. 19}$$

and in the second sub-iteration, we solve the equation

$$\frac{\partial^2 h}{\partial t^2} = gd \frac{\partial^2 h}{\partial y^2}. \quad \text{eq. 20}$$

The advantage of this technique is that the required linear systems remain tridiagonal so the computational cost per iterations is proportional to the number of samples on the surface. The resulting implementation remains very simple. For the first sub-iteration, we compute the update as before on each row of the height-field. For the second sub-iteration, we do the same for each column in the height-field. While artifacts can potentially arise from the favored directions, our experience with the alternating-direction formulation of these equations is very favorable. More so than in two-dimensions, it is important to be careful with the details of the volume conservation. Errors manifest themselves as line artifacts along the x or y axes.

RENDERING WITH CAUSTIC SHADING

Given a realistic way of simulating water motion, the next step is to render it convincingly. Several different effects must be taken into account. Firstly, rays of light which are incident on a water surface are refracted by that surface. This results in uneven illumination of the terrain underneath the water. This effect is illustrated in Fig. 2 in which the ray from the light source is shown being deflected by the water surface.

Instead of the incident light ray hitting the terrain at A, the ray ends up at B. The total illumination at B will depend on all the rays which are refracted in that direction for the entire water surface. To compute this exactly, it would be necessary to render a hemispherical around B of all of the water. This is prohibitively expensive, especially since the shading of the water surface depends on the view point unlike the typical diffuse case in radiosity. An alternative approach[18], feeds rays forwards from the light source and accumulates the results in an illumination texture for the surface to be shaded. Such an algorithm requires the intersection of a ray both with the water surface height field and the terrain height field for each illuminating ray. A very large number of rays is required for this technique, especially for grazing-incidence illumination. For samples evenly spaced around the light source, the average sample density on the terrain is inversely proportional to the cosine of the angle between the incoming ray and the surface normal. For grazing incidence illumination, very large numbers of illuminating rays are required.

To avoid this expense, we use two approximations. The first is the flat bottom approximation which is illustrated in Fig. 3. If the terrain is locally flat, then the destination of the

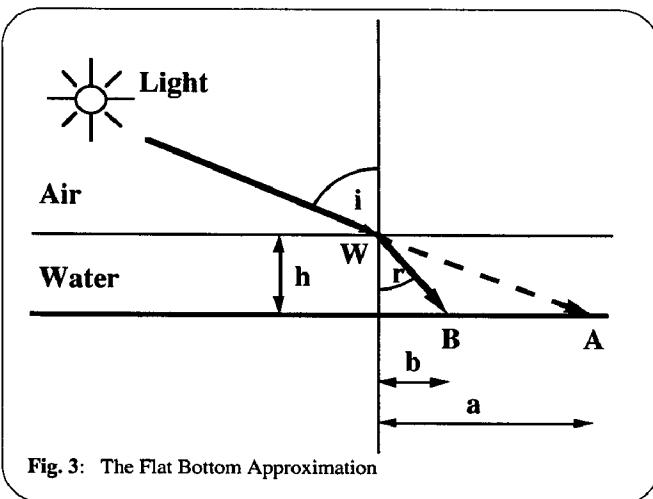


Fig. 3: The Flat Bottom Approximation

ray may be computed using simple trigonometry.

$$b = h / (\tan(i))$$

So, instead of tracing a ray from **W** towards the terrain to find the intersection point **B**, a simple expression suffices. To avoid undersampling the terrain with incident rays, we take samples which are evenly spaced in terrain grid coordinates. If there is no water present the samples will just remain on the surface point. This results in a uniform illumination map. We have effectively divided out the cosine of the angle between the normal and the illumination direction. To compute the correct shading intensity, the illumination map value must be multiplied by the cosine term. (This is unnecessary for the correctly ray-traced case since the decreasing sample density automatically takes this effect into account.) The second simplification is the flat water approximation. The point **A** is considered as the destination of a ray as if it was undeflected by any water. The height of the water above **A** is then used to compute an approximate position for a ray intersection with the water surface.

$$a = h / (\tan(i))$$

This value of *a* gives an estimate of the position of **W**. The refracted ray is computed from the water depth and normal at **W** using Snell's law. The refracted ray is then used in combination with the flat bottom approximation to compute the position of **B**. The net result of this process is that a sample moves from **A** to **B**. No ray-tracing against terrain is done at any stage and the intersection calculations are extremely economical. Each sample acts as the center for a conical convolution kernel which is added into the illumination map. The caustic illumination for the images in this paper took 40 seconds on a Silicon Graphics 4D/210 workstation. The final stage of the rendering is to scan-convert the height field into a Z-buffer. The shading for the scan-conversion again uses the flat bottomed approximation for the shading of the surface of the water. A ray is assumed to pass from the eye through the surface point, and is refracted towards the terrain. The terrain intersection point for this ray is computed in the same way as point **B** in Fig. 3. The illumination map value for that point is then used for the terrain shading contribution to the ray color. A second contribution to the ray color arises from light being reflected towards the eye by the water surface. An environment map for the sky is used for this term and the two are blended together using the Fresnel equations for unpolarized light [19].

To get a realistic appearance for water on sand it is necessary to take into account the fact that sand becomes wet on contact with water and then gradually dries when the water is no longer covering it. This effect may be achieved using a

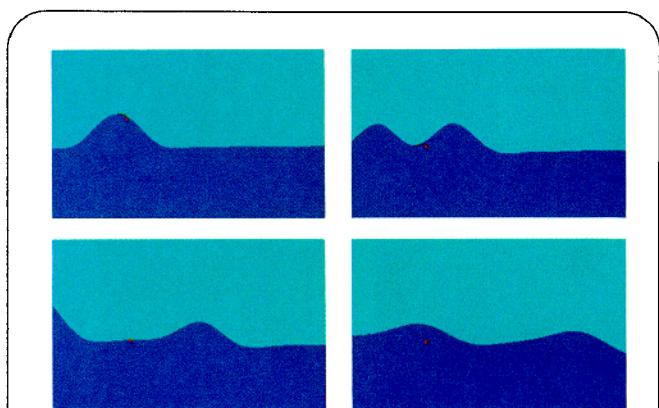


Fig. 4: Real-time two-dimensional example showing wave reflection. Top: A user has created a disturbance in the water producing two waves going in opposite directions. Bottom left: The wave is reflecting off the left wall. Bottom Right: After the reflection, both waves are travelling to the right.

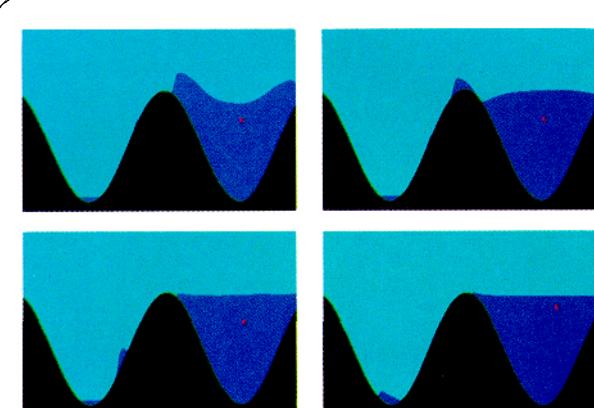


Fig. 5: Real-time two-dimensional example showing changing boundaries. Left to right, top to bottom: A user has created a wave that carries some water over the hump to fall on the other side.

wetness map. At each frame time, the water is tested to see if it is above the terrain for each sample position. When this is the case, the wetness value for the sample is set to one. For samples in which the water is not above the terrain the wetness value is decreased by a constant amount with the wetness value clipped to zero if necessary. During rendering, the wetness value is used to darken the diffuse shading of the terrain by up to 50%. Also, the same wetness value is used to blend between the terrain shader and the water shader. Unfortunately, the wetness map has to be the same resolution as the water grid so, for a 256 by 256 simulation, aliasing artifacts are visible in the boundary between wet and dry areas. These artifacts may be partially reduced by filtering the wetness map before using it for shading.

The 3-D figures in this paper were rendered at NTSC resolution using a 256 by 256 array of surface points and two by two supersampling. Each frame took about ten minutes to compute on an SGI 4D/210.

EXAMPLES

We begin with some two-dimensional examples to illustrate basic wave phenomena. Fig. 4 shows four frames calculated by a real-time implementation of the two-dimen-

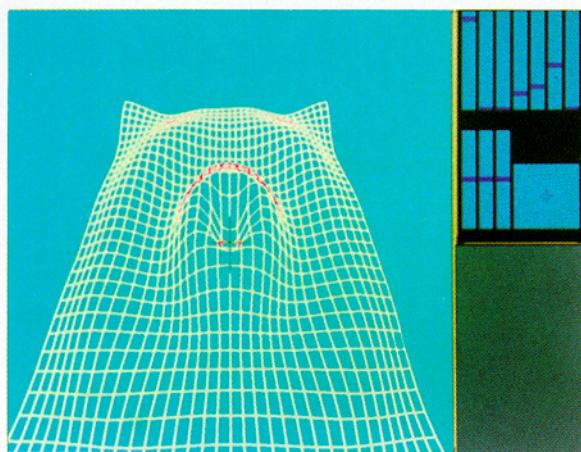


Fig. 6: Three-dimensional real-time implementation of the fluid equations. The implementation runs at 30 frames per second with a 32 by 32 grid size on an SGI 4D/210.

sional equations on an SGI 4D/210. A user creates a disturbance in the water surface which causes waves propagating in both directions. When the waves hit the walls, they are reflected back towards the center. The reflecting walls are created simply by making the ground function b very steep. Unlike techniques which explicitly track the behavior of individual waves, there is no need to create additional data structures to handle the reflections. They simply emerge as a consequence of solving the differential equation.

Fig. 5 shows four frames from a real-time simulation that exhibits changing boundary conditions. A user creates a wave which carries water over a hump and down the other side. These changing boundaries emerge very easily from eq. 13. As a wave-front propagates over previously dry ground, it can lift the next sample in the height field to a position above the ground at each iteration and continue on its way. The boundary cannot advance more than one sample per iteration because of the fact that d is considered constant throughout an iteration. This is not a serious problem because waves slow down as they approach shallow areas where the boundary conditions can easily change. In deep areas where the wave velocity is greatest, the waves are free to move multiple numbers of samples per iteration.

Fig. 6 shows a still frame from a real-time simulation in three dimensions. The simulation achieves a 30 frame-per-second update rate using a 32 by 32 grid on an SGI 4D/210. Interactive controls allow the user to select the position of a sinusoidal disturbance on the height field, adjust the time-step (and therefore the effective wave velocity) and control the viewing parameters. This type of interaction is very different from the traditional batch-oriented approach required by standard methods on supercomputers.

The first rendered animation example is shown in Fig. 7. Here rain drops fall on a concave surface and begin dribbling down towards the low point in the center. After enough drops have accumulated, waves begin to form on the puddle at the bottom and exhibit extensive reflections and refractions as they interact with the complex depth patterns that arise from the flow. The motion of the rain drops in the air is simulated using a standard particle system.

Fig. 8 shows two frames from an animation of fluid smoothly flowing down from a spring near the top of a hill. As the fluid flows downward, it goes around bumps in the terrain. The fluid has been made very viscous-looking by means of eq. 17. As a result, the waves are of very low amplitude and most of the interesting behavior is in the way the bound-

aries change.

Fig. 9 shows six frames from an animation which begins with waves hitting a beach. After the first wave goes up the beach and recedes, rain begins to fall.

CONCLUSION

There is a long history of people using differential equations to analyze and simulate fluid flow for engineering purposes. Here we have attempted to make use of that work to derive a simplified model that is well suited to the demands of animation. The model is stable, rapid and easy to program. The computation time is linear in the number of samples of the height field, making high-resolution simulations possible. In the three-dimensional case, the computation for each row and each column is independent, so it can be easily parallelized. Unlike models which rely on tracking individual waves or wave-trains, reflected waves, changing boundaries and net flow can be handled in a simple manner. As a consequence, this model extends the range of water effects which can be animated in a reasonable time.

By using a number of approximations it is possible to render convincing caustic shading effects at little computational cost. A wetness map adds to the realism of water flowing over sand. When combined with the fluid dynamics model, the results are encouragingly realistic.

ACKNOWLEDGEMENTS

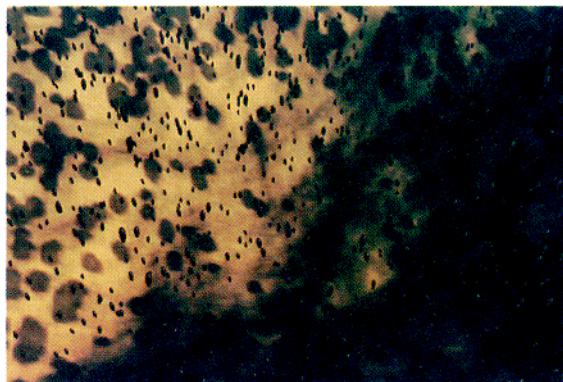
We thank the Advanced Technology Group at Apple Computer for supporting and encouraging this work and providing a stimulating work-environment. In addition, we thank the ATG graphics group for discussions and helpful advice. We also thank Scott Stein for system support and the Apple Library for aiding us with literature searches.

REFERENCES

- [1] Max, N., "Vectorized procedural models for natural terrain: Waves and islands in the sunset," *Proceedings of SIGGRAPH 81*, (August 1981) 317-324.
- [2] Peachy, D., "Modeling Waves and Surf," *Proceedings of SIGGRAPH 86*, (August 1986), 65-74.
- [3] Fournier, A. and Reeves, W., "A Simple Model of Ocean Waves," *Proceedings of SIGGRAPH 86*, (August 1986), pp 75-84.
- [4] Ts'o, P. and Barsky, B., "Modeling and Rendering Waves," *ACM Transactions on Graphics*, 6, 3 (July 1987), 191-214.
- [5] Masten, G., Watterberg, P. and Mareda, I., "Fourier Synthesis of Ocean Scenes," *IEEE Computer Graphics and Application*, 7, 3 (March 1987) 16-23.
- [6] Lewis, J., "Generalized Stochastic Subdivision," *ACM Transactions on Graphics*, 6, 3 (July 1987) 167-190.
- [7] Perlin, K., "An Image Synthesizer," *Proceedings of SIGGRAPH 85*, (July 1985) 287-296.
- [8] Schachter, B., "Long crested wave models," *Computer Graphics and Image Processing* 12 (Feb. 1980), 187-201.
- [9] Miller, G. and Pearce, A., "Globular Dynamics: A connected particle system for animating viscous flu-



- ids," *Computer Graphics* 13,3 (1989) 305-309.
- [10] Sims, C., "Particle Dreams," [Video] *Siggraph Video Review* 38/39, ACM SIGGRAPH, New York, segment 42 (1988).
 - [11] Patel, B. and Dvinsky, A., "The solution of the reynolds averaged Navier-Stokes equations in general curvilinear coordinates and its application to vehicular aerodynamics," in *Computers in Design, Manufacture and Operation of Automobiles*, Murthy and Brebbia, Eds., Springer Verlag, Berlin (1987).
 - [12] Kallinderis, Y. and Baron, J., "Adaptation methods for a new Navier-Stokes algorithm," *AIAA Journal*, 27, 1 (January 1989) 37-43.
 - [13] Miyata, H. and Nishimura, S., "Finite difference simulation of nonlinear waves generated by ships of arbitrary three-dimensional configuration," *Journal of Computational Physics* 60 (1985) 391-436.
 - [14] Press, W., Flannery, B., Teukolsky, S. and Vetterling, W., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge (1986).
 - [15] Le Mehaute, B., *An Introduction to Hydrodynamics and Water Waves*, Springer-Verlag, New York (1976).
 - [16] Crapper, G., *Introduction to Water Waves*, John Wiley & Sons, New York (1984).
 - [17] Stoker, J., *Water Waves*, Interscience, New York, (1957).
 - [18] Shinya, M., Saito, T. and Takahashi, T., "Rendering Techniques for Transparent Objects," *Proceedings of Graphics Interface*, London, Ontario (June 1989).
 - [19] Hall, R., *Illumination and Color in Computer Generated Imagery*, Springer Verlag, Berlin (1988).



(a)



(b)

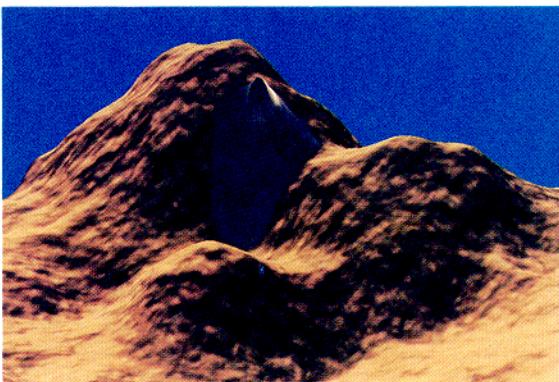


(c)



(d)

Fig. 7: Rain fall on a concave surface. After the drops hit the surface, they begin rolling towards the low point in the center. In (a), the first drops have hit the previously dry surface. In (b) most of the surface is wet and the first drops have started rolling towards the center. In (c) and (d) the rain continues to fall and enough water accumulates to create interesting wave patterns.

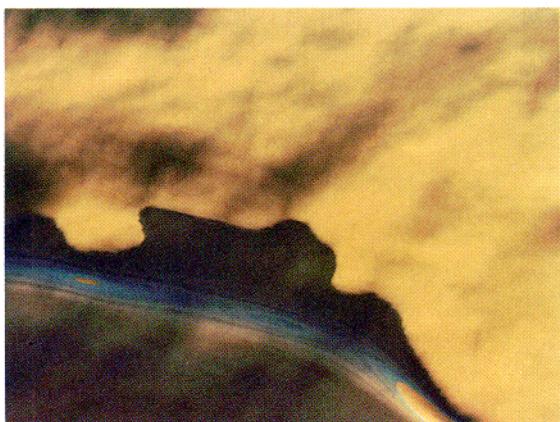


(a)



(b)

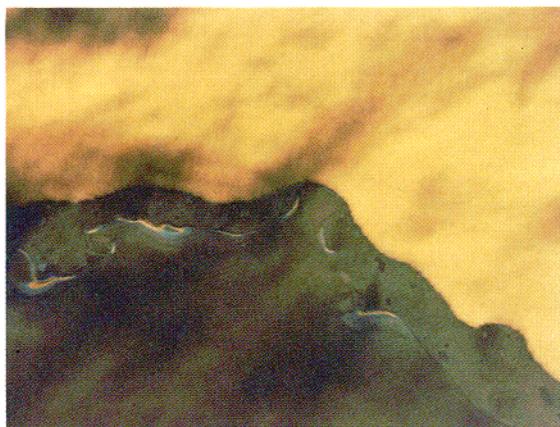
Fig. 8: A source of fluid near the top of one of the bumps is turned on, creating a gentle spring. The damping term in eq. 17 is used to make the flow very viscous. The fluid begins to flow downward in (a) and rolls gently down the hill in (b) going around bumps in the terrain.



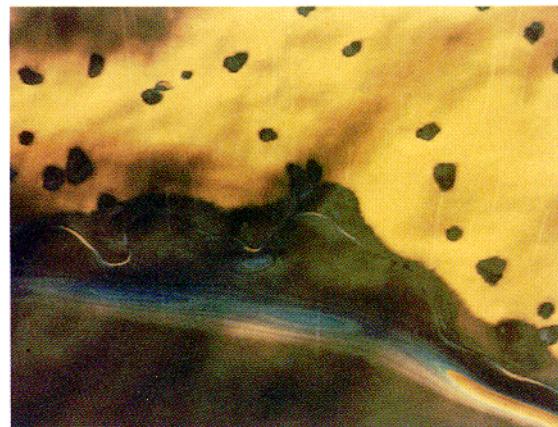
(a)



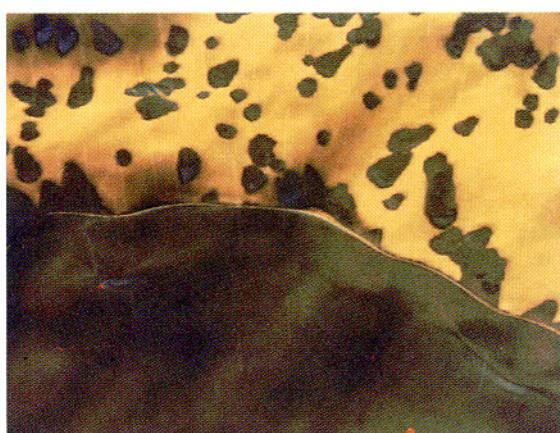
(b)



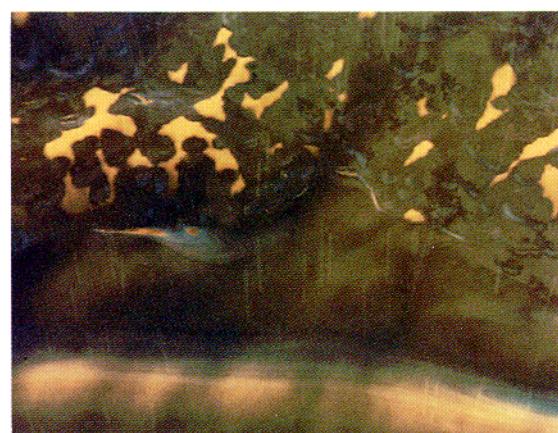
(c)



(d)



(e)



(f)

Fig. 9: In (a) and (b) a wave approaches the shore. Note that the wave bends markedly between (a) and (b). This is due to refraction caused by the dependence of wave speed on depth. In (c) the wave recedes, leaving wet sand behind. In (d) through (f) rain begins to fall and flow downward over the sand.