Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Animation and Control of Breaking Waves

Viorel Mihalef[1], Dimitris Metaxas[1], Mark Sussman[2]

[1] CBIM Center, Dept. of Computer Science, Rutgers Univesity
[2] Dept. of Mathematics, Florida State University

Abstract

Controlling fluids is still an open and challenging problem in fluid animation. In this paper we develop a novel fluid animation control approach and we present its application to controlling breaking waves. In our Slice Method *framework an animator defines the shape of a breaking wave at a desired moment in its evolution based on a library of breaking waves. Our system computes then the subsequent dynamics with the aid of a 3D Navier-Stokes solver. The wave dynamics previous to the moment the animator exerts control can also be generated based on the wave library. The animator is thus enabled to obtain a full animation of a breaking wave while controlling the shape and the timing of the breaking. An additional advantage of the method is that it provides a significantly faster method for obtaining the full 3D breaking wave evolution compared to starting the simulation at an early stage and using solely the 3D Navier-Stokes equations. We present a series of 2D and 3D breaking wave animations to demonstrate the power of the method.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismAnimation

1. Introduction

One of the most complex natural phenomena in the world of liquids is that of the breaking wave. The famous painting "The Great Wave" (fig.1) by Hokusai, Japan's best known plastic artist, provides wonderful insight into the nature of the difficulties associated with simulating breaking waves. One can notice the presence of multiple scales of dynamics, ranging from the fine scale of the spray and the foam, passing through the small/intermediate scale of the ripples on the wave surface, and ending with the larger wavelength of the wave that overturns and breaks. While there are no satisfactory physical models for the foam/spray dynamics (see the Related Work section for more about this) we would like in this paper to address the simulation and control of the larger scale dynamics of the waves.

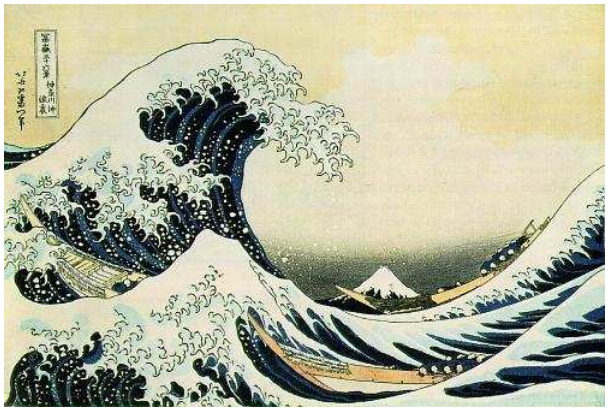


Figure 1: *"The Great Wave" by Hokusai is a beautiful showcase of the various scales involved in breaking.*

In the best of worlds, an animator will have the freedom to choose how the free surface of a liquid will look like at a specific moment in time, where it will be located and how its general subsequent evolution will look like. Our work is an attempt to get closer to this ideal type of control over the coarse behavior of a liquid while still maintaining realistic fluid behavior based on the laws of physics. We are focusing on specifying the shape and location of the free surface of a liquid at a given moment in time, that is of central importance for the animation. While our case study is that of

breaking sea waves, the method applies to more general fluid simulations.

We start from the reasonable assumption that each vertical slice of a three dimensional breaking wave (parallel to the wave direction of propagation) mimics the dynamics of a two-dimensional one. This leads to the idea of using two-dimensional simulations to help: 1. generate the *shape* of a 3D wave at a chosen moment in time and 2. generate the *velocities* of the 3D wave at that time instant. The usefulness of the idea resides not only in enabling us to start the 3D simulation using a physically correct velocity field but also in allowing us to recover some if not all of the previous dynamics of the wave. On top of control we also gain speedup of the calculation, given that we start it much later than in a standard CFD computation, where one begins from a height-field geometry (sinusoidal for example). Let's also emphasize that this method is usable in various other contexts, for example in controlling the geometry of a smoke plume or of the fiery ball in a nuclear explosion - like the one in [RNGF03], which shares the same spirit with the present work. One needs to only generate the 2D simulations and then integrate them in a 3D shape while controlling the 3D geometry. This is the essence of what we call the *Slice Method*. In its application to breaking wave simulation it can be summarized as follows:

1. the animator builds the 3D shape of the breaking wave by choosing each of its vertical 2D slices from a library of waves; the animator can also preview the short time expected behavior

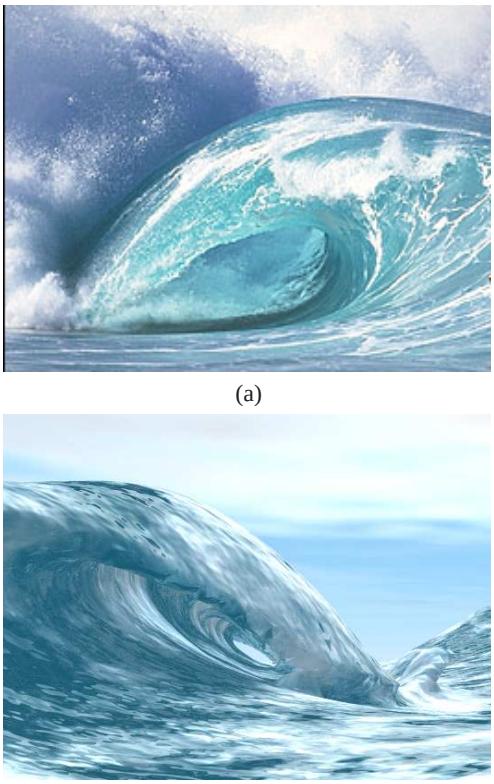
2. the program generates the subsequent evolution by flowing forward in time based on the 3D Navier-Stokes equations.

3. (optional) the program generates previous evolution by flowing the free-surface back in time using the pre-computed 2D dynamics.

The outcome in our specific case study is obtaining the dynamic behavior of overturning 3D waves from the onset up to post-breaking stages, having as initial information the shape at some chosen time step. The animator has control over the shape of the wave at that time step and its intended subsequent behavior without the need of specifying the velocity by hand (specification which would be essentially impossible to do in order to get realistic behavior). In order to populate the 2D wave library we had to do extensive 2D simulation work and we report here some of the results.

2. Related Work

In an attempt to present the roots of our approach, we give here a short overview of previous work on liquid simulation and fluid control reported in the graphics community. We briefly discuss ocean surface animation, then general simulation of liquids and end with a short account of control methods.



(a)

(b)

Figure 2: *(a) Snapshot of a real wave (courtesy Joseph Libby) (b) Simulation using the Slice Method*

2.1. Ocean Surface Animation

The simulation of ocean surface has been essentially a 2D story. Without going into details, one can distinguish two main animation approaches. The first one is represented by [FR86, Pea86, Sch80, TB87] who used parametric functions (a mix of Gerstner and Biesel swell models) to model the free-surface of water in order to simulate wave transport. The second one attempts to synthesize the ocean surface as a height field with a prescribed spectrum based on semi-empirical models, and it includes work by Mastin [MWM87] and Tessendorff [Tes99]. An interesting work by Thon et al. [TDG00] attempts to bridge the two approaches by using a synthesized spectrum to control a Gerstner model. They generate the small scale disturbances by adding a spectrum-driven Perlin noise model [Per85].

2.2. Simulation of Liquids

For a while the main thrust in simulating liquids was trying to augment the 2D models to produce 3D effects. Kass and Miller's model [KM90] was essentially a height field with dynamics governed by the 2D shallow water equations and it was extended in [CL94] by incorporating pressure effects,

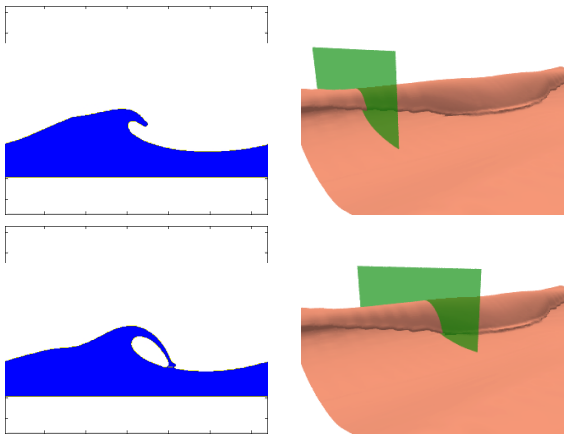


Figure 3: *From 2D to 3D: the Slice Method. The left pictures show two of the slices used to build the wave on the right; on the right side we show their respective position in the 3D wave, given by the green vertical cross-sections.*

obtained by solving a 2D Navier-Stokes equation.By further extending the height field model, [OH95] introduced a particle system that helped simulating splashing. [MP89] used a system of springs between particles to simulate viscosity effects and [TPF95] simulated melting using molecular dynamics in a deformable shape setup.

The advantage of using the full 3D Navier-Stokes became clear after Foster and Metaxas [FM96] modified the classic marker and cell (MAC) method [HW65] to obtain realistic fluids behavior for a decent computational price. They extended their method later [FM97] to include simple control mechanisms at the level of physical parameters. Stam [Sta99] departed from the finite-difference scheme used by Foster and Metaxas and introduced the "stable" semiLagrangian methods for computing the advection part of the Navier-Stokes equations. Foster and Fedkiw introduced in [FF01] a hybrid liquid model, combining implicit surfaces and particles, while Enright et al. [EFFM02] improved the hybrid model devising the particle level set method, which is presently one of the methods of choice for obtaining very realistic animations of complex water surfaces. Level set based solvers like the aforementioned one (level sets were introduced in [OS88]) are en-vogue lately due to their capabilities for capturing and rendering the interface as a smooth implicit surface and also for the theoretical ease with which they can deal with topology changes. The classical level set method cannot be used just by itself for fluid simulations due to poor mass conservation properties. While the particle level set method does a good job regarding keeping the mass constant for tests where there are no topological changes, the method does not explicitly enforce mass conservation and may lose mass for complex flows with lots of topological changes. Sussman [SP00, Sus03] addressed the loss of mass of the level set method by combining it with a volume of fluid method - and this is the numerical method used in the present work as well. Takahashi et al. [TFK*03] presented nice and fast simulations of water effects (including lots of splashes) and interaction with objects as a direct application of the CIP method (a semiLagrangian method which uses gradient information for better advection properties) discovered by Yabe (for a good review see [Yab97]). Their work integrates particle systems with the CIP Eulerian framework and, while the results for the spray simulation are a good step towards realism, the ones for foam simulation require improvement and probably the use of a better physical model. Our paper complements their work, in the sense that they were not specifically concerned with generating ocean breaking waves or controlling them.

2.3. Control of Fluids

We can classify control techniques for fluids into (1) control of physical parameters and (2) control of geometric parameters.

The first type allows a user to control physical parameters such as density, viscosity and surface tension, or dynamic variables, such as velocity, pressure and temperature, and modify them to satisfy various animation goals. This type of control was initiated by Foster and Metaxas [FM97] who used for example variable viscosity to control the CFL condition and variable surface tension to control the dynamics of a fountain. [FF01] controlled the motion of water flow by prescribing exact values for the velocity field at specific locations while [EMF02] used variable surface tension to speed-up the settling of water poured in a glass.

The second type of control is closer to the keyframing procedure familiar to animators. Essentially, one tries to obtain a simulation such that the fluid shape matches one or more predefined keyframes. The first successful simulation of this type was done by [TMPS03] for smoke control. Concurrently with our work, [MTPS04] demonstrate for the first time a type of geometric control simulation applied to liquids. However, their method cannot be applied to a natural phenomenon like breaking waves for two reasons: first, the wave breaking phenomenon has an intrinsic natural dynamics that would require, besides geometric tuning, the use of physics-based velocities; second, it would be extremely tedious to obtain the 3D breaking wave keyframes that would be needed for such an animation.

Our paper offers a possible solution to the two challenges mentioned above. In the following we present our animation and control methodology for breaking waves.

3. Simulation and Control

Using an Eulerian framework, all our simulations are based on a rectangular grid implementation of the Navier-Stokes

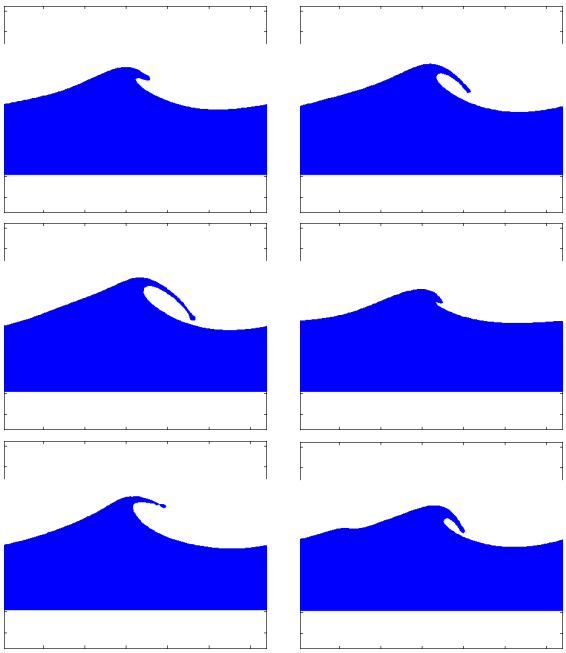


Figure 4: *Various waves from the library. From left to right and top to bottom, the control parameters* (A,d,ω) *(see section 3.2) are:* $(.5,.3,3)$, $(.5,.3,3.5)$, $(.5,.4,3)$, $(.6,.3,2)$, $(.7,.3,1)$, $(.7,.3,2)$

equations. As we have mentioned in the introduction, the starting point in building the breaking wave simulations is the *generation of a library* of 2D breaking waves. This enables the animator in the next step to *generate the 3D shape* by using the 2D library to choose the desired slice geometry (fig. 3). The 2D wave library comes with a very useful piece of information, that is the velocity vector field. Either by reading directly the slice vector field or, if necessary, by linearly interpolating it between slices, one obtains the 3D velocity vector field, necessary for starting the three dimensional Navier-Stokes simulation.

3.1. The Fluid Solver

When we simulate fluids there are two ingredients we need to take care of: (1) fluid surface detection, where one either tracks or captures the liquid surface, and (2) Navier-Stokes solver, where one solves the momentum equations governing the fluid dynamics.

3.1.1. Fluid surface detection

Fluid detection at any time step is done by tracking or capturing the fluid surface. Tracking the surface is an example of the Lagrangian approach, and consists of advecting a set of markers that approximate the surface at any time step. In 3D one needs to perform rather complicated surgery to handle the splitting or merging of the surface, hence the preference of some researchers for Eulerian methods. In an Eulerian method one uses a grid to localize the surface; one can use massless markers to track the surface as in the MAC method used by Foster and Metaxas or one can "capture" the surface, in the sense that, instead of explicitly tracking it, one updates a scalar field on the Eulerian grid which defines (captures) the surface. For example, the volume of fluid (VOF) method uses a "color" function, equal to one if a computational cell contains only liquid, and equal to zero if a computational cell contains no liquid. The level set method defines the surface as the zero level set of a scalar field, which is, say, positive inside the liquid, and negative outside it. The surface is reconstructed at any time by means of an implicit surface reconstruction algorithm like marching cubes, for example. We should note that merging or splitting of the surface is "built-in" for such methods and one doesn't need to do anything fancy in order to deal with them. The main advantage of the volume of fluid method is the explicit enforcement of conservation of volume, while the level set method offers simplicity of implementation and ensures surface smoothness. The combined level set and volume of fluid (CLSVOF) method inherits the advantages of both these methods. Due to our specific coupling of the volume fraction function with the level set function, the presence of spurious volumes that mars the classical VOF method is not observed.

We give now a brief overview of the CLSVOF method. For a more detailed description please refer to [Sus03] and [SP00]. The position of the free surface (the wave surface in our case) is updated using the level set equation

$$\phi_t + \mathbf{u}\cdot\nabla\phi = 0 \tag{1}$$

where $\mathbf{u}$ is the velocity of the liquid and ϕ is the level set function which is positive in the liquid and negative in the air. Along with the level set equation we also solve the volume fraction advection equation:

$$f_t + \mathbf{u}\cdot\nabla f = 0 \tag{2}$$

where the volume fraction is initialized in each computational cell Ω at $t = 0$ as:

$$f_\Omega = \frac{1}{vol(\Omega)}\int_\Omega H(\phi(x,0))dx \tag{3}$$

where H is the Heaviside function:

$$H(\phi) = \begin{cases} 1, & \phi \geq 0; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The essential coupling of the level set function ϕ and the volume of fluid function f is the following:

- the level set function is used to truncate the volume fractions, thus removing the "flotsam" usually associated with VOF methods; the truncation is done by removing any volume fractions outside a cell-width wide narrow band about the zero level-set.
- the volume fraction function is used to construct the level set as a volume-preserving distance function

The discrete volume fraction and level set function are defined on a staggered grid (MAC) and are located at the cell centers. They are advected using a "coupled" second order conservative operator split advection scheme. The velocity vector field $\mathbf{u}$ used to update ϕ and f in (1) and (2) is obtained from the momentum equations which we describe below.

3.1.2. Momentum equations

The Navier-Stokes equations for incompressible flow are:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \nu \Delta \mathbf{u} + f \quad (5)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (6)$$

The first equation is the fluid equivalent of Newton's law ($\frac{D\mathbf{u}}{Dt}$ is the acceleration, here written as the "material derivative" of the velocity, i.e. $\frac{D\mathbf{u}}{Dt} = \mathbf{u}_t + \mathbf{u} \cdot \nabla u$) while the second one enforces mass conservation. The fluid has constant density ρ, has velocity given by the vector field $\mathbf{u}$, p is the pressure inside the body of fluid, ν is its dynamic viscosity coefficient (we used $\nu = 0$ in our computations) and any other forces acting on it are denoted by f (in our case f was due to gravity and body forces). These equations must be supplemented by appropriate boundary conditions, for example $\mathbf{u} = 0$ on solid walls at rest.

Equation (5) states essentially that the force acting on a fluid particle is the combined effect of convection forces (attempting to carry it along with the rest of the fluid), pressure gradient forces directed towards lower pressure regions (see the minus sign), dampening forces which scale along with the viscosity coefficient and other (external) forces, for example surface tension, gravity or body forces.

The numerical scheme that we use to compute solutions to (5) and (6) is based on the projection approach [BCG87] and is described in detail in [Sus03]. The second order Runge-Kutta method is used to advance the solution in time. A second order, slope limited, upwind finite difference scheme is used to discretize the nonlinear advective terms.

3.2. 2D Wave Generation

The generation of our 2D wave library consists of two pieces: (1) we prescribe initial conditions derived from linear wave theory for our 2D fluid solver and (2) we advance the prescribed initial conditions using the 2D Navier-Stokes solver in order to construct 2D wave slices ranging from simple geometries to 2D breaking wave profiles. The waves in the 2D wave library are henceforth used to prescribe initial conditions for the 3D Navier-Stokes solver (usually right before the 3D wave breaks).

For prescribing initial conditions for the 2D Navier-Stokes solver, we characterize waves by their period, T, wavelength, λ, and amplitude, A. The wavelength λ measures the distance between any two points of the same phase.

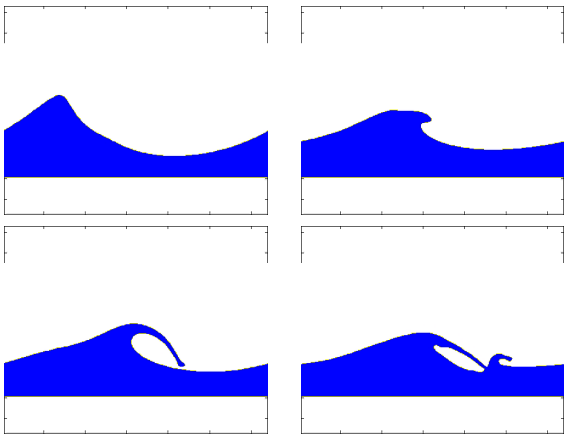


Figure 5: *Snapshots of the time sequence of a plunging breaking wave (at times 1.2, 3.6, 4.7, 5.6) . Camera moving with the wave speed (more details in the text). Such waves form the 2D wave library.*

If we denote $k = \frac{2\pi}{\lambda}$ and $\omega = \frac{2\pi}{T}$, the surface displacement for a wave is given by

$$\eta(x,t) = A\cos(kx - \omega t) \quad (7)$$

and, if we consider any fluid particle, its velocity components are given by

$$u(x,t) = A\omega e^{-kz}\cos(kx - \omega t) \quad (8)$$

$$v(x,t) = A\omega e^{-kz}\sin(kx - \omega t) \quad (9)$$

where z is the depth of the particle with respect to the mean surface level. Note that the velocity decreases exponentially with depth. In fluid mechanics literature it is agreed that wave breaking may occur due to direct wind forcing, wave-wave interaction or wave instability. We have concentrated on the last option and used a steep Stokes wave of third order of high slope $\varepsilon = kA = 2\pi A/\lambda$ (as in [CKZL99]) which led to breaking due to instability. The initial - let's call it "standard" - 2D profile was (one can view it as a superposition of waves of type (7)):

$$\eta(x,0) = d + \frac{1}{2\pi}(\varepsilon\cos(2\pi x) + \frac{1}{2}\varepsilon^2\cos(4\pi x) + \frac{3}{8}\varepsilon^3\cos(6\pi x))$$

(d denotes the water depth) and the velocity was obtained from the corresponding superposition using (8) and (9). We present in fig. 5 some results obtained for $\varepsilon = 0.55$, g = 0.1, $\omega = 3$, density equal to 1 and viscosity equal to 0. Our results are quite similar to the ones obtained by [CKZL99]. The computational domain has size 128×128 and the physical domain is $[0\ 1] \times [0\ 1]$.

In order to generate the library we modified the following parameters: two geometric parameters, the amplitude A and the depth d and the kinematic parameter ω. Other parameters which one may potentially vary were kept constant in order

to keep the dimension of the parameter space small enough (specifically, we always used $\nu = 0$, $g = 0.1$ and $k = 2\pi$). For example, the slope $\varepsilon = kA$ for a plunging breaker varied between 0.5 and 0.7, values below 0.5 generating just the so-called "spilling breakers" which don't feature any plunging phenomenon while for even smaller values we obtained simple progressive-waves. The kinematic parameter ω was kept between 2 and 4, again with the lower values corresponding to progressive-waves and spilling breakers and the larger values to plunging breakers. Each run was limited to about 7 seconds of physical time, which ensured that breaking took place. One important issue was that, more often than not, one needs to move the camera along with the wave, in order to keep the slices "aligned". We implemented the camera movement by simply adding a horizontal component to the velocity usually equal to -0.1 (we are allowed to do this as our computations are periodic in the x-direction). A run similar to the "standard" one takes 3-4 minutes for a 128×64 grid. To build the wave library one could then simply sample the cube $[0.5/2\pi\ 0.7/2\pi] \times [0.25\ 0.35] \times [2\ 4]$ in the space (A, d, ω) (the parameters are based on our experiments).

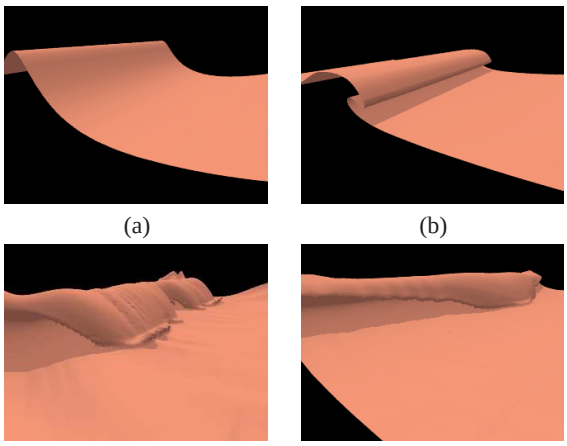


Figure 6: *One can easily build variable initial geometries. (a) Axisymmetric wave, early stage type, (b) intermediate stage type, (c) and (d) are surf wave type with variable geometries*

3.3. 3D Wave Generation and User Control. The Slice Method.

The three-dimensional wave generation starts traditionally from a very early stage of the water surface given as a height field, for example as a sinusoidal ruled-surface. We are not constrained by this though: the Slice Method allows us to start at a later moment and, most importantly, the geometry of the surface does not need to be a height field, it does not even need to have genus zero! In fig. 2 we show a picture of a real wave with non-zero genus and a similar wave generated with the Slice Method. One can appreciate the level of geometric control that the method offers, allowing us to build 3D surf waves which would otherwise not be obtainable with standard methods.

The slices used in building this particular 3D geometry were obtained from the 2D simulation described in the previous section, varying between slice-time 2.6 and 5.0. This was the initial configuration used to start the Navier-Stokes solver that generated the results in the surf wave sequence from fig. 10. In fig. 6 we indicate that the animator can easily build the desired initial shape: axisymmetric, starting at an earlier (a) or later stage (b), or variable geometry (c and d). In particular, the geometry in (a) was used to start the axisymmetric wave in fig. 7, while the geometry in (c) was used to generate the waves in fig. 2 and 10 and the one in (d) the wave in fig. 8. The process of building the 3D geometry is the following (we will refer to figures 5 and 6): the initial 3D configuration is an axisymmetric wave, usually a few moments before the overturning (for example the first image in fig. 6). Arguably the most important geometric features belong to the moment of breaking (third frame in fig. 5), and usually we build this region first. The user previews in real-time the evolution of the 2D wave. If the user is not satisfied with that specific breaking dynamics or geometry he or she can switch to previewing another 2D evolution by changing the available geometric and kinematic parameters (A, d or ω). Accordingly, the program would then switch to another library item, such as the ones in fig. 4. If even more flexibility is needed, one may choose to start a new 2D run to build a new library item. Once satisfied with the slice geometry the user "accepts" this slice and the program updates the 3D level set to reflect the change by simply rewriting the level set values in the vertical plane of the slice. We implemented a "hard update" which changes the level set values for the 3D position of that slice only and also a more viewer-friendly "soft update", which updates along with the chosen slice all the neighboring slices so that no two neighboring slices are more than one time step apart; indeed, for most of the domain, this provides the right "look". The general user control algorithm can be written as:

```
for each 3D slice
    if (!satisfactory)
        browse library and choose another 2D dynamics
        if (!satisfactory)
            run new 2D simulation
    update the 3D level set by rewriting slice level set
    run marching cubes
```

After one slice is chosen one proceeds to modifying another one and so on. Using the "soft update" (which essentially updates several slices at a time) one can efficiently build the 3D geometry. For example, the geometries in fig. 6, c and d, were built in about 5 minutes each, with lots of the time spent performing marching cubes on the $128 \times 128 \times 64$ grid. One can imagine other methods of improving the user

control, for example applying a smoothing filter to the 3D level set after building the 3D breaking wave. One can also implement a free horizontal translation of a 2D breaking wave sequence by resampling the level set (as our waves are periodic in the x direction) - this could help in aligning the 2D waves in the library to a standard model. Such extensions would be easy to implement and definitely useful in a production setting.

A few remarks are in order:

- from the 3D configuration obtained one can naturally flow backwards in time by uniformly decreasing the slice-time
- the Slice Method is applicable in general to any phenomenon which exhibits some type of weak-symmetry. In our case one can think of such a symmetry being provided by the group of translations along the horizontal axis normal to the direction of the wave propagation, modulo time evolution of each slice. For the simulation of a nuclear explosion, for example, like the one in [RNGF03], the group of symmetry is, of course, the group of rotations about the vertical axis.
- one could think of using only the two-dimensional library instead of solving the 3D Navier-Stokes for the simulation subsequent to the moment of control, in a similar way it was done in [RNGF03]. While we are currently investigating this option we should point out that there are several problems with it: mass won't necessarily be conserved, and this is bound to create visual artifacts in a simulation of liquids - while when simulating smoke/fire/explosions this is not a problem. Also, one won't be able to handle easily the interaction with objects - unless they are shape-invariant to the group of symmetry mentioned above. Moreover, using a fully 3D Navier-Stokes solver is preferable as, if needed, it allows the user to use the standard physical parameter control methods (as discussed in section 2.3). Finally, the 3D simulation yields the necessary and rich longitudinal instability that would be absent from a 2D simulation (but is visible in fig. 7, for example).

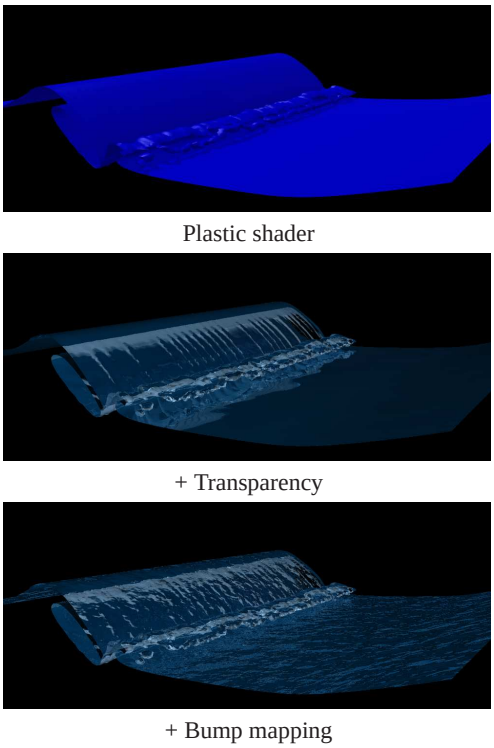


Figure 7: *Rendering methods applied to an axisymmetric breaker*

4. Animations

All the three-dimensional results are based on the two-dimensional wave library generated as explained previously. The elements of the library include a full range of profiles situated between progressive-waves and plunging breakers (like the one in fig. 5). The physical parameters (density, viscosity, etc.) for each 3D simulation were the same ones used in the 2D simulation that was used to define the slices.

The simplest 3D geometry is that of an axisymmetric wave, for which all the vertical slices are at the same time (fig. 7). For this particular one we worked on a $96 \times 96 \times 48$ grid and started the simulation at time t=1.0 seconds, when the wave is still a height field (much like in the standard CFD simulations). One can observe a few important effects in this simulation, namely the formation of the air tube under the wave front, immediately after break-up, together with the formation of the secondary jet. Also notice the fingering of this secondary jet (that would be absent from a 2D simulation), due to performing a full 3D Navier-Stokes simulation.

Perhaps the most impressive examples are given by the surf waves like the ones in figures 8 or 10 (the grid size was $128 \times 128 \times 64$ and the slice time varied between 3.2 and 6.0 seconds). Immediately after the breakup one can notice a lot of turbulence - which is one of the main factors influencing foam and spray generation. The motion tames down afterwards, as expected. Here is a good place to point out again that in the traditional CFD simulations of 3D breaking waves (and of most of the other standard types of simulation) one has to start in the early stage of a symmetric configuration (like the early stage of the axisymmetric wave). Obtaining surf waves with controlled geometry like the ones we present here is simply out of bounds for the traditional methods.

As an example of a 3D interaction with objects we present in fig. 9 a "Milky Wave" crashing against a rocky structure (the grid size was $96 \times 96 \times 48$ and the slice times varied between 3.2 and 4.8 seconds). By using the Slice Method we acquired control on the shape of the breaker right before breaking.

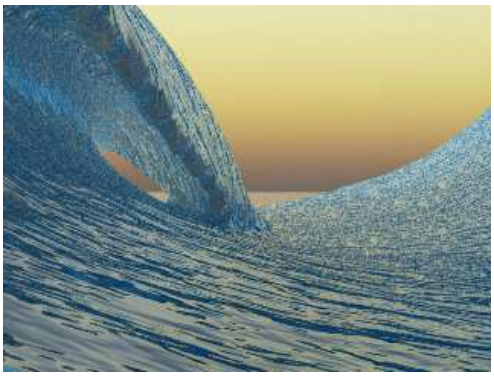


Figure 8: *Under a surf wave*

Figure 9: *The "Milky Wave": interaction with fixed objects*

The animations took about 10-12 min/frame to generate for a resolution of $128 \times 128 \times 64$ on a Pentium 3, 2.6 GHz machine with 2 Gb of RAM. The rendering has been mainly done in Vue d'Esprit [Vue 4] at the cost of about 4-6 min for a 640×480 picture. The water ripples are generated using three dimensional bump mapping (Perlin fractal noise), similarly to [Per85]. The illusion of ripples moving on the surface of the wave is due to the movement of the wave through the fixed 3D bump map. This is a non-physically based approach, which requires further investigation, and we do intend to implement the spectral control used in [TDG00] to make it physically-based; nevertheless, we believe that the good looking results fully justify reporting it. The wave in fig. 2 was texture-mapped and ray-traced in Vue d'Esprit.

5. Conclusions and Future Work

In this paper we presented a method for generating two and three dimensional breaking wave simulations and also for exerting control on the shape and behavior of 3D breaking waves at a certain (desired) moment in time. The Slice Method enables us to obtain results that cannot be obtained using standard computational fluid dynamics simulation techniques. We believe this work is a good starting point that opens several avenues of research, such as creating user-friendly software for the generation of breaking waves and the application of the Slice Method to controlling other types of fluid simulations.

6. Acknowledgements

The first author would like to thank Prof. Norman Zabuski for offering useful insight into various fluid dynamics related issues, and also Dr. Sukmoon Chang and Dr. Kyung-Ha Min for their help and advice.

References

[BCG87] BELL J. B., COLELLA P., GLAZ H. M.: A second-order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics 85* (1987), 257–283. 5

[CKZL99] CHEN G., KHARIF C., ZALESKI S., LI J.: Two-dimensional Navier-Stokes simulation of breaking waves. *Physics of Fluids 11*, 1 (1999), 121–133. 5

[CL94] CHEN J., LOBO N.: Toward interactive-rate simulation of fluids with moving obstacles using the Navier-Stokes equations. *Computer Graphics and Imagage Processing 57* (1994), 107–116. 2

[Vue 4] www.e-onsoftware.com. 8

[EFFM02] ENRIGHT D., FEDKIW R., FERZIGER J., MITCHELL I.: A hybrid particle level set method for improved interface capturing. *J. of Computational Physics 183* (2002), 83–116. 3

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM TOG 21*, 3 (2002), 736–744. 3

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. *Proceedings of SIGGRAPH 2001*, 23–30 (2001). 3

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical Models and Image Processing 58* (1996), 471–483. 3

[FM97] FOSTER N., METAXAS D.: Controlling fluid animation. *Computer Graphics International* (1997), 178–188. 3

[FR86] FOURNIER A., REEVES W. T.: A simple model of ocean waves. *ACM SIGGRAPH 1986* (1986), 75–84. 2

[HW65] HARLOW F. H., WELCH J. E.: Numerical calculation of time-dependent viscous incompress-

ible flow of fluid with a free surface. *Physics of Fluids 8* (1965), 212–218. 3

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. *ACM SIGGRAPH 1990* (1990), 49–57. 2

[MP89] MILLER G., PEARCE A.: Globular dynamics: a connected particle system for animating viscous fluids. *Computers and Graphics 13*, 3 (1989), 305–309. 3

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIC Z., STAM J.: Fluid control using the adjoint method. *ACM Transactions on Graphics 23*, 3 (2004). 3

[MWM87] MASTIN G. A., WATTERGER P. A., MAREDA J. F.: Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications 3* (1987), 16–23. 2

[OH95] O'BRIEN J., HODGINS J.: Dynamic simulation of splashing fluids. *Computer Animation 95* (1995), 198–205. 3

[OS88] OSHER S., SETHIAN J. A.: Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79* (1988), 12–49. 3

[Pea86] PEACHY D.: Modeling waves and surf. *ACM SIGGRAPH 1986* (1986), 65–74. 2

[Per85] PERLIN K.: An image synthesizer. *Computer Graphics 3*, 19 (1985), 287–296. 2, 8

[RNGF03] RASMUSSEN N., NGUYEN D., GEIGER W., FEDKIW R.: Smoke simulation for large scale phenomena. *Siggraph 2003, ACM TOG 22* (2003), 703–707. 2, 7

[Sch80] SCHACHTER B.: Long crested wave models. *Computer Graphics and Imagage Processing 12* (1980), 187–201. 2

[SP00] SUSSMAN M., PUCKETT E. G.: A coupled level set and volume of fluid method for computing 3D and axisymmetric incompressible two-phase flows. *Journal of Computational Physics 162* (2000), 301–337. 3, 4

[Sta99] STAM J.: Stable fluids. *ACM SIGGRAPH 1999* (1999), 121–128. 3

[Sus03] SUSSMAN M.: A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *Journal of Computational Physics 187* (2003), 110–136. 3, 4, 5

[TB87] TS'O P. Y., BARSKY B. A.: Modeling and rendering waves: wave tracing using beta-splines and reflective and refractive texture mapping. *ACM Transactions on Graphics 6*, 3 (1987), 191–214. 2

[TDG00] THON S., DISCHLER J. M., GHAZANFARPOUR D.: Ocean waves synthesis using a spectrum-based turbulence function. *Computer Graphics International Proceedings* (2000). 2, 8

[Tes99] TESSENDORF J.: Simulating ocean water. *Siggraph Course Notes* (1999). 2

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. *Eurographics* (2003). 3

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIC Z., STAM J.: Keyframe control of smoke simulations. *ACM Transactions on Graphics 22*, 3 (2003), 716–723. 3

[TPF95] TERZOPOULOS D., PLATT J., FLEISCHE K.: Heating and melting deformable models (from goop to glop). *Graphics Interface 89* (1995), 219–226. 3

[Yab97] YABE T.: Universal solver CIP for solid, liquid and gas. *CFD Review* (1997). 3

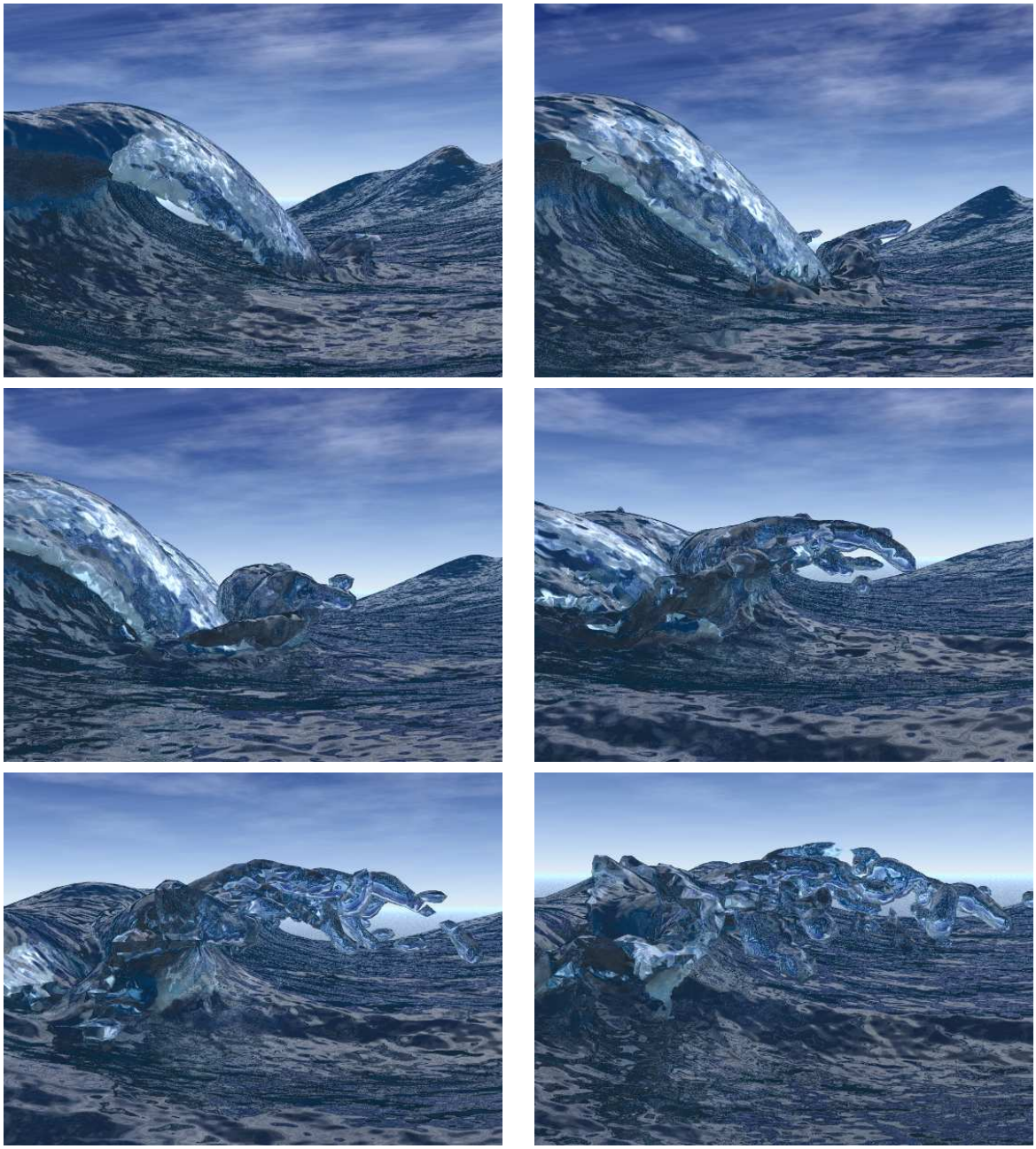


Figure 10: *Surf wave sequence*