

# Solving General Shallow Wave Equations on Surfaces

Huamin Wang<sup>†</sup> and Gavin Miller<sup>‡</sup> and Greg Turk<sup>†</sup>

<sup>†</sup>Georgia Institute of Technology

<sup>‡</sup>Adobe Systems Incorporated

---

## Abstract

We propose a new framework for solving General Shallow Wave Equations (GSWE) in order to efficiently simulate water flows on solid surfaces under shallow wave assumptions. Within this framework, we develop implicit schemes for solving the external forces applied to water, including gravity and surface tension. We also present a two-way coupling method to model interactions between fluid and floating rigid objects. Water flows in this system can be simulated not only on planar surfaces by using regular grids, but also on curved surfaces directly without surface parametrization. The experiments show that our system is fast, stable, physically sound, and straightforward to implement on both CPUs and GPUs. It is capable of simulating a variety of water effects including: shallow waves, water drops, rivulets, capillary events and fluid/floating rigid body coupling. Because the system is fast, we can also achieve real-time water drop control and shape design.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

---

## 1. Introduction

Water is crucial to human life. People see, use and drink water everyday, thus it is not surprising that water simulation plays an important role in computer graphics. Incompressible fluid dynamics in general can be described by the Navier-Stokes equations. Due to the complexity of the 3D Navier-Stokes equations, water behavior varies significantly under different circumstances. Researchers have invented and adopted many numerical methods to generate various realistic water effects by solving the Navier-Stokes equations. While most previous research has been focused on simulating the high-speed dynamics of large bodies of water, our main interest in this paper is to provide a fast and accurate technique for graphics applications to simulate small-scale fluid dynamics and its interaction with solid surfaces, such as toy boats floating in the bathtub and water drops streaming down a glass.

Compared with other representations such as particles, tetrahedral meshes or grid structures, we believe that a height

field is more suitable to represent small-scale water in low-speed fluid animations. A height-field-based system is easier to implement and its computational space increases only quadratically with spatial resolution. Also, restricting the Navier-Stokes equations to 2D makes it possible to apply implicit numerical schemes, which usually means more stability and higher efficiency. However, previous height-field-based techniques only supported a limited range of effects. In particular, surface tension was neglected, which diminished the accuracy of small-scale liquid simulations.

In order to remove these limitations, our first contribution is a general height-field-based system (Section 3) that solves the new General Shallow Wave Equations (GSWE) that are extensions of the traditional shallow wave equations. Our system builds height columns along surface normals rather than in the absolute gravity direction, as the left picture in Fig. 1 shows. External forces in our system include gravity (Section 4) and surface tension (Section 5), and they are solved by implicit schemes which are significantly more stable and efficient than explicit schemes. We further incorporate a two-way liquid/rigid body coupling method in Section 6 for floating and drifting effects. Finally in Section 7, we demonstrate how our system can be optimized to run in

---

<sup>†</sup> Email: {whmin, turk} @ cc.gatech.edu

<sup>‡</sup> Email: gmiller @ adobe.com

real time on both CPUs and GPUs, which allows users to control and design fluid shapes interactively.

This system is most suitable for simulating low-speed fluid dynamics of small-scale 3D water flows on surfaces, which is expensive to generate with particle systems or grid systems. It provides an efficient way to predict the behavior of high speed dynamics before applying more accurate but time-consuming techniques. While each different system is capable of efficiently generating some specific effects, a comprehensive system that combines height fields with particles or grids as in [OH95, IGLF06] should be able to capture more aspects of water animations in the future. We believe that the work presented in this paper will be an important sub-component of such a system.

## 2. Related Work

Water animation is notoriously difficult and time-consuming to produce because solving the 3D Navier-Stokes equations requires a huge computational domain. The computational space increases cubically in 3D when water is represented either by particle systems [Mon92, MCG03] or 3D grid systems [FM96, Sta99, FF01, EMF02], not to mention the extra computational cost required by the CFL condition. As expected, a natural solution to improve simulation performance is to reduce the computational space. Instead of using completely uniform grids in 3D, Losasso et al [LGF04], Houston et al. [HNB\*06] and Nielsen and Museth [NM06] proposed to use non-uniform grids such as octree structures or cells with various lengths. Feldman et al. [FOK05], Klingner et al. [KFCO06] and Elcott et al. [ETK\*07] considered solving fluid dynamics on unstructured tetrahedral meshes rather than grid structures. Treuille et al. [TLP06] used Principal Component Analysis (PCA) to further reduce a static computational space. However, it is difficult to extend this method to free surface water animations.

Since the computational space of a height-field-based system increases only quadratically with the spatial resolution, height fields were introduced to the graphics community by the work of Kass and Miller [KM90] for rapid fluid simulations. This technique was later augmented with semi-Lagrangian velocity advection by Layton and van de Panne [LvdP02]. Instead of using shallow wave equations, Chen and da Vitoria Lobo [CdVLHM97] proposed to only solve pressure projection in 2D while keeping other simulation steps in 3D. Recently, Irving et al. [IGLF06] demonstrated how to combine a height field representation with a grid structure in order to simulate some non-height-field behaviors such as overturning and splashing.

In order to simulate 2D water flows on curved surfaces, Stam [Sta03] proposed the use of quadrilateral meshes from Catmull-Clark subdivision. Flow simulation directly on manifold triangle meshes was demonstrated by Shi and Yu [SY04] and Elcott et al. [ETK\*07] using discrete dif-

ferential geometry (DDG) operators. Kim et al. [KLLR07] further adopted the BFECC method to reduce numerical dissipation in surface flows. Our method also simulates water flows on meshes directly, and we allow surfaces with arbitrary topology.

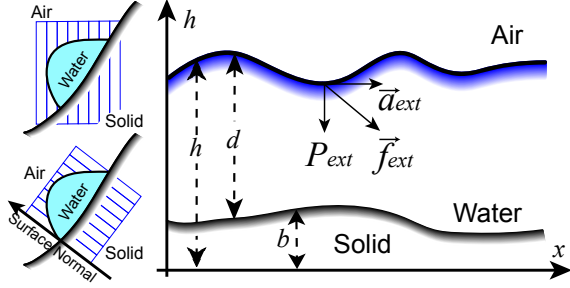
Considerable research has been done recently on grid systems and particle systems to model coupling between water and other objects, including rigid bodies, cloth and thin shells and other fluids. For height-field-based systems, fluid/rigid body coupling methods were proposed by O'Brien and Hodgins [OH95] and Chen and da Vitoria Lobo [CdVLHM97] to simulate waves caused by moving objects.

Other topics related to our work include water drop simulation [FHP98, NHS02], water wave simulation [HNC02, MMS04], fluid surface tension [HK03, CM04, SSK05, WMT05], and fluid control [TMPS03, MTPS04, FL04, SY05].

## 3. General Shallow Wave Equations

Our goal in this section is to extend traditional Shallow Wave Equations (SWE) [KM90] to our new General Shallow Wave Equations (GSWE). As their names imply, SWE and GSWE are both based on the shallow wave assumptions: the wave velocity is low and the wave height variation is small. The original shallow water equations can be derived from the Navier-Stokes equations according to the method of Saint-Venant [UJ04]. However, the height field in SWE is built in the absolute gravity direction and the only external force is the gravity force in the horizontal direction. While a SWE system can be greatly simplified because of these, it suffers from the following limitations. Firstly, solid surfaces could not be too steep otherwise water drops will not be properly represented, as shown by the upper left picture in Fig. 1. Secondly, it is not clear how to incorporate arbitrary external forces, such as surface tension forces and user control forces, nor how to develop implicit schemes for arbitrary external forces. Interaction between height-field-based water and the environment is also difficult to model in a physically-based manner.

The height field in GSWE is constructed along surface normals rather than in the absolute gravity direction as Fig. 1 shows. In order to avoid self-intersection of water columns when the surface is fully detailed with small bumps, we use averaged surface normals from a low-resolution surface and represent the difference between the original surface and the low-resolution surface as a terrain height field  $b(x)$  in the local background. The water height field is then defined as a function  $h(x, t)$  of the surface position  $x$  and time  $t$ . Horizontal water velocity is  $\vec{u}(x, t)$ , and vertical velocity is implicitly given as  $\partial h / \partial t$ . Since our system allows arbitrary external forces, we first separate a 3D force  $\vec{f}_{ext}$  into a 1D pressure component  $P_{ext}$  and a 2D acceleration component  $\vec{a}_{ext}$  as the



**Figure 1:** Left: The height field is built along surface normals rather than in the absolute gravity direction. Right: External forces are separated into pressure and acceleration components.

right picture in Fig. 1 shows. Although these two components act on the water based on different mechanisms, they both take effect by changing the water's horizontal velocity  $\vec{u}$ . The pressure component, including air pressure, surface tension pressure and vertical gravity pressure, squeezes water and causes horizontal movement due to pressure difference. On the other hand, the acceleration component, including user control force and horizontal gravity acceleration, acts on the horizontal velocity immediately. By restricting the 3D Navier-Stokes equations to 2D surfaces, we can formulate the non-viscid General Shallow Wave Equations as follows:

$$\vec{u}_t = -(\vec{u} \cdot \nabla)\vec{u} - \nabla P_{ext}/\rho + \vec{a}_{ext} \quad (1)$$

$$h_t + \nabla \cdot (h - b)\vec{u} = 0 \quad (2)$$

in which  $\rho$  is the water density. Eq. 1 updates the horizontal velocity due to both  $P_{ext}$  and  $\vec{a}_{ext}$ . Eq. 2 updates the height field and maintains the incompressibility implicitly. Eq. 2 can be reorganized into:

$$h_t + (h - b)\nabla \cdot \vec{u} + \vec{u} \cdot \nabla h = 0 \quad (3)$$

Since we are targeting at slow water, we ignore the velocity term  $(\vec{u} \cdot \nabla)\vec{u}$  in Eq. 1. According to the method of characteristics, we first solve the height field advection due to the velocity using an explicit solver:

$$h_t + \vec{u} \cdot \nabla h = 0 \quad (4)$$

The rest of Eq. 1 and 3 are then simplified by differentiating Eq. 3 with respect to  $t$  and Eq. 1 with respect to the spatial dimensions, and then eliminating cross-derivatives:

$$\frac{\partial^2 h}{\partial t^2} = \frac{d\Delta P_{ext}}{\rho} - d\nabla \cdot (\vec{a}_{ext}) \quad (5)$$

When external forces vary slowly through time such as user control forces, we treat them as if they are temporally invariant so that Eq. 5 can be safely solved by explicit methods at each time step. However, two common natural forces, the

gravity force and the surface tension force, are quite sensitive to height field changes as time evolves. If we still use explicit solvers in this case, the system would require significantly smaller time steps according to the CFL condition in order to avoid instability. Therefore, we develop implicit schemes for both of these forces in Sections 4 and 5 respectively. They are combined and solved together in a single matrix system:

$$(\mathbf{A}_g + \mathbf{A}_s + \mathbf{I} - \mathbf{I}_c)h^t = b - b_c \quad (6)$$

in which  $h^t$  is the unknown height field at time  $t$ , and the previous height fields  $h^{t-1}, h^{t-2}, \dots$  at time  $t-1, t-2, \dots$  are already known. Matrices  $\mathbf{A}_g$  and  $\mathbf{A}_s$  are formulated from implicit schemes for gravity and surface tension respectively.  $\mathbf{I}_c$  and  $b_c$  are the coupling matrix and vector, which will be discussed in Section 6.  $b$  is the prospective height field augmented with artificial viscosity effects by a factor  $\tau$ :

$$b = h^{t-1} + (1 - \tau)(h^{t-1} - h^{t-2}) \quad (7)$$

After the height field has been solved, we update the velocity field using Eq. 1 and apply the surface friction by a damping factor  $\frac{h(x)}{h(x)+d}$ . When  $d$  is zero, there is no friction and when  $d$  goes larger, the friction effect becomes more obvious.

### 3.1. Spatial Discretization

Because the height field is built along solid surface normals, spatial discretization in our system depends on solid surfaces rather than absolute 3D space. We discretize flat surfaces into regular grids and then use finite differencing schemes to formulate the GSWE matrix system.

For curved surfaces, we uniformly discretize them by a particle repulsion algorithm, instead of relying on parameterizing them into quadrilateral grids. We then build a height column at each particle as Fig. 2 shows. During each repulsion step, we search for the six closest particles and these particles become the neighborhood  $U_i = \{X_{i_1}, X_{i_2}, \dots, X_{i_6}\}$  of each particle  $X_i$ . We use six neighbors because this is the average number of neighbors once such a relaxation system converges. We find this approach produces similar results to other methods such as repulsion of all points within a fixed radius. Next, we sort the neighbors in counter-clockwise order. The repulsion force applied on  $X_i$  is then calculated as:

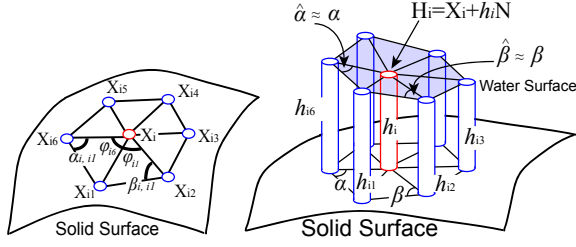
$$f(X_i) = a \sum_j w_{i,j} (X_j - X_i) \quad (8)$$

$$w_{i,j} = \cot \alpha_{i,j} + \cot \beta_{i,j}$$

The force magnitude factor  $a$  is usually between 0.01 and 0.2 in our experiments.  $w_{i,j}$  is a non-zero weight factor only when  $X_j \in U_i$  as in [GSD05], and  $\alpha_{i,j}$  and  $\beta_{i,j}$  are two angles facing toward the same edge  $(X_i, X_j)$  as Fig. 2 shows. The advantage of calculating repulsion forces using Eq. 8 instead of polynomials in [Tur91] or Gaussian falloffs in [Hec97] is to facilitate the surface tension scheme, as will be discussed later in Section 5. In order to make  $w_{i,j}$  symmetric,



**Figure 3:** BUNNY: The left picture illustrates spatial discretization and the cell connectivity on the bunny model. Colored cells are water cells and black cells are boundary cells. The right picture shows a water drop changes its path after the bunny model was rotated 90°.



**Figure 2:** Particles and their height columns are constructed on a solid surface.

we add  $X_i$  into  $X_j$ 's neighborhood once we know  $X_j \in U_i$ , even though  $X_i$  may not be necessarily in  $U_j$ . After the repulsion process has converged, we construct the particle connectivity from the nearest neighbors and then we calculate the differential geometry operators directly from the neighborhood according to [GSD05]. In this way, we do not need to reconstruct manifold triangle meshes as required in [SY04, ETK\*07].

### 3.2. Boundary Conditions

When simulating large bodies of water with no dry areas, we use either Dirichlet boundary conditions or Neumann boundary conditions for different wave reflection effects on the boundary.

When simulating water drops that are sparsely distributed on surfaces, we do not define the height field in dry regions in order to save memory and computational cost. But unlike undefined regions for large bodies of water, these dry regions are dynamic and need to be updated as long as the water flows. Naively, we can recognize dry regions as cells with non-positive heights. However, in order to avoid grid artifacts, we also define boundary cells with negative heights immediately next to water cells so that a more accurate boundary can be estimated through interpolation. The left picture in Fig. 3 illustrates boundary cells in black on the bunny model. Boundary cells only exchange water with neighboring water cells when we solve GSWE in defined regions, including both boundary cells and water cells. We ap-

ply the boundary conditions between boundary cells and water cells, where the actual boundary is specified: Neumann conditions prevent water from entering that boundary, while Dirichlet conditions allow water to move freely.

After the height field has been updated, we remove boundary cells from defined regions if they are no longer next to water cells. When a boundary cell becomes a water cell, its neighborhood may contain new boundary cells as well. Boundary oscillation may be caused by a surface discontinuity if new boundary cells are not initialized with proper height values. Similar to the fast marching algorithm [Set99, OF02], our method initializes new boundary cells with  $C^1$  continuity, which is sufficient in most cases.

### 4. Implicit Gravity Scheme

Since the height field is built along surface normals rather than in the gravity direction, we separate the gravity into two components (vertical and horizontal) as in Section 3:

$$\begin{aligned} P_g(X_i) &= \rho g_i h_i \\ \vec{a}_g(X_i) &= \vec{g} - g_i \mathbf{N}_i \end{aligned} \quad (9)$$

in which  $g_i$  is the vertical gravity acceleration at  $X_i$ ;  $g_i = -\vec{g} \cdot \mathbf{N}_i$ . We can calculate  $\vec{a}_g$  directly after mesh resampling because each  $g_i$  at  $X_i$  is a temporal invariant. On the other hand,  $P_g$  depends on the height field and requires an implicit scheme for greater stability.

Our implicit gravity scheme is extended from [KM90] by applying a different gravity acceleration  $g_i$  at each position  $X_i$ . Given a 1D height field  $h = \{h_1, h_2, \dots, h_n\}$  from  $X_1$  to  $X_n$ , by treating  $P_g$  as  $P_{ext}$  in Eq. 5 and differentiating  $h'_i$  using finite differencing in the spatial domain, we obtain the implicit matrix  $\mathbf{A}_g$  as the coefficients for  $h'$  in the system Eq. 6:

$$\begin{aligned} a_{i,i} &= \frac{1}{4\Delta x^2} [(d_i + d_{i+1})(g_i + g_{i+1}) + (d_i + d_{i-1})(g_i + g_{i-1})] \\ a_{i,i+1} &= a_{i+1,i} = -\frac{1}{4\Delta x^2} (d_i + d_{i+1})(g_i + g_{i+1}) \end{aligned} \quad (10)$$

Similarly, the matrix for curved surfaces by the differential geometry scheme is:

$$\begin{aligned} a_{i,i} &= \frac{1}{8A} \sum_j w_{i,j} (d_i + d_j) (g_i + g_j) \\ a_{i,j} &= a_{j,i} = -\frac{1}{8A} w_{i,j} (d_i + d_j) (g_i + g_j) \end{aligned} \quad (11)$$

The particle surface area  $A$  is assumed to be uniform for all particles, after the solid surface has been uniformly sampled by the particle repulsion algorithm.

### 5. Implicit Surface Tension Scheme

According to Laplace's law, surface tension pressure  $P_{surf}$  is related to surface mean curvature  $K$ , which can be estimated using the Laplace-Beltrami operator in [GSD05]:

$$P_{surf}(\mathbf{H}_i) = \gamma \cdot K_i$$

$$K_i = \frac{1}{2A} \left| \sum_j \hat{w}_{i,j} (\mathbf{H}_j - \mathbf{H}_i) \right| \quad (12)$$

$$\mathbf{H}_i = \mathbf{X}_i + h_i \mathbf{N}_i$$

in which  $\gamma$  is the surface tension coefficient, and  $\mathbf{H}_i$  is the water surface extended along surface normal  $\mathbf{N}_i$  at  $\mathbf{X}_i$  as in Fig. 2. We assume that  $\mathbf{N}_i$  is locally constant after the solid surface has been densely sampled. We also assume that the weight factor  $\hat{w}_{i,j}$  on water surfaces is close to the weight factor  $w_{i,j}$  on solid surfaces under the shallow wave assumptions. Since the repulsion force in Eq. 8 diminishes to zero after sampling, Eq. 12 is simplified to:

$$K_i = \frac{1}{2A} \left| \sum_j w_{i,j} (\mathbf{X}_j - \mathbf{X}_i) + \sum_j w_{i,j} (h_j \mathbf{N}_j - h_i \mathbf{N}_i) \right| \quad (13)$$

$$= \frac{|\mathbf{N}_i|}{2A} \sum_j w_{i,j} (h_j - h_i) = \frac{1}{2A} \sum_j w_{i,j} (h_j - h_i)$$

The Laplace-Beltrami operator on a curved surface is a generalized version of the Laplacian on a flat surface. This operator is naturally normalized since the sum of all coefficients for a particular node  $\mathbf{X}_i$  is zero from Eq.13.

One way to formulate an implicit scheme for surface tension from Eq. 12 and 13 is to treat all  $h_i$  and  $h_j$  as unknowns completely at the next time step. Let us just consider the flat surface as an example. The Laplacian of the height field gives surface tension pressure, which needs to be processed by the Laplacian again according to Eq. 5. By central differencing, the matrix  $\mathbf{A}_s$  for flat surfaces will have cell kernels that take the form:

$$\begin{pmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{pmatrix} \quad (14)$$

as coefficients. We omit the more complicated matrix formula for curved surfaces here. Although this complete implicit scheme is numerically stable, its matrix is not straightforward, symmetric or sparse like that of the gravity scheme, because the kernel spans more than the 1-ring neighborhood. Fortunately we notice from Eq. 14 that cells beyond the 1-ring neighborhood have much less influence than those in the 1-ring neighborhood. Therefore, we propose another incomplete implicit scheme by only treating height columns within the 1-ring neighborhood as unknowns after applying  $P_{surf}$  in

Eq. 12 into the system Eq. 5. This gives us a symmetric and sparse matrix  $\mathbf{A}_s$  as:

$$a_{i,i} = \frac{\gamma}{4\rho A^2} \sum_j w_{i,j}^2 (d_i + d_j)$$

$$a_{i,j} = -\frac{\gamma}{4\rho A^2} w_{i,j}^2 (d_i + d_j) \quad (15)$$

Height columns beyond the 1-ring neighborhood will be considered as knowns by taking values from the current time step. They formulate a vector  $b_s$ , which is added to vector  $b$ :

$$b_s(i) = -\frac{\gamma}{8\rho A^2} \left\{ B_i^{t-1} + 2 \sum_j w_{i,j}^2 (d_i + d_j) (h_j^{t-1} - h_i^{t-1}) \right\}$$

$$B_i^{t-1} = \sum_j w_{i,j} (d_i + d_j) (K_j^{t-1} - K_i^{t-1})$$

$$K_i^{t-1} = \sum_j w_{i,j} (h_j^{t-1} - h_i^{t-1}) \quad (16)$$

We implemented and tested both implicit schemes, and we did not notice any significant benefits from using the complete scheme. Since the incomplete scheme is more efficient, we choose the incomplete surface tension scheme for all of the following experiments if we do not explicitly say otherwise.

### 5.1. Drops: Surface Tension and Contact Angles

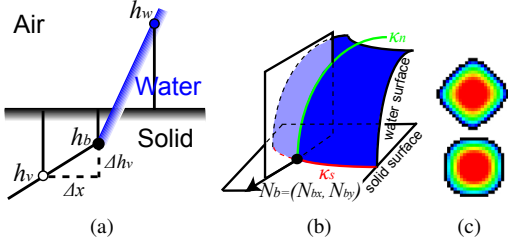
There are two important factors that contribute to the shape and motion of water drops: surface tension forces at the air/water interface, and the effect of the hydrophilicity of the surface on which drops are forming. To account for surface tension forces at the air/water interface, we apply surface tension pressures to both the water cells and the boundary cells as discussed in Section 3.2. To account for hydrophilicity, we use the virtual surface method proposed in [WMT05] to produce water drops with various contact conditions. Such contact conditions allow us to simulate hydrophobic surfaces on which drops will bead up, and hydrophilic surfaces on which drops are flattened. Although the actual boundary (also called the contact front) is between boundary cells and water cells, our experiments showed that it is safe to simply assume boundary cells as exact contact fronts and apply boundary surface tension pressures only to them. The 1D case is shown in Fig. 4a. Since surface tension is estimated as weighted difference as in Eq.13, the virtual surface is interpreted as height difference between boundary cell  $h_b$  and virtual surface cell  $h_v$ :

$$\Delta h_v = h_b - h_v = \Delta x \tan \theta_c \quad (17)$$

in which  $\theta_c$  is the contact angle. For the 2D regular grid, we incorporate an angular factor  $\alpha$  to count for the fact that the boundary normal is not aligned with the grid axis:

$$\Delta h_v = \alpha \Delta x \tan \theta_c$$

$$\alpha = \begin{cases} N_{bx} + N_{by}, & w_l + w_r + w_u + w_d = 1 \\ (w_l + w_r)N_{bx} + (w_u + w_d)N_{by}, & \text{otherwise} \end{cases} \quad (18)$$



**Figure 4:** Illustrations of the virtual surface method. Part (a) gives the virtual surface in the 1D case. Part (b) shows the contact line normal  $N_b$  and two assumed principal components  $\kappa_n$  and  $\kappa_s$ . Part (c) shows water drop shapes without (top) and with (bottom) the angular factor  $\alpha$ .

in which  $w_l$ ,  $w_r$ ,  $w_u$  and  $w_d$  are 1 if left, right, up or down neighbor cells are in defined regions, otherwise 0.  $N_{bx}$  and  $N_{by}$  are the boundary's surface normal coordinates as shown in Fig. 4b. Fig. 4c shows that a water drop can correctly bead up into a circle by using the angular factor, otherwise it might become a diamond shape. On curved surfaces, since the connectivity between height columns could be arbitrary, instead of finding a proper angular factor, we sum the boundary mean curvature from two assumed principal components  $\kappa_n$  and  $\kappa_s$  as Fig. 4b shows. The depth component  $\kappa_n$  in the normal plane modulates how steep the surface is, which can be estimated similarly to the 1D case. The angular component  $\kappa_s$  in the horizontal plane measures the contact line curvature as:

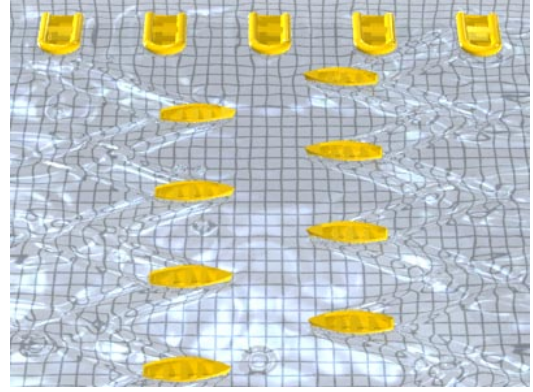
$$\kappa_s = \sigma \sum_i \begin{cases} \varphi_i, & h_i - h_b > 0 \text{ and } h_{i+1} - h_b > 0 \\ \frac{(h_i - h_b)\varphi_i}{h_i - h_{i+1}}, & h_i - h_b > 0 \text{ and } h_{i+1} - h_b < 0 \\ \frac{(h_{i+1} - h_b)\varphi_i}{h_{i+1} - h_i}, & h_i - h_b < 0 \text{ and } h_{i+1} - h_b > 0 \end{cases} \quad (19)$$

in which  $\sigma$  is a scalar factor to modulate  $\kappa_s$ 's magnitude, and  $\varphi_i$  is the angle between two neighboring edges as in Fig. 2. We maintain a wetness map from water paths and choose the contact angle based on the surface wetness.

## 6. Two-Way Fluid/Rigid Body Coupling

In order to make our system complete for wave effects, we also incorporate a simple two-way fluid/rigid body coupling algorithm based on the method proposed by O'Brien and Hodgins [OH95]. This algorithm is separated into three steps. In Step 1, we recognize those height columns in which solid objects contact with water. For planar cases, this can be accelerated by using the depth buffer in graphics hardware. In Step 2, we solve the dynamics system (Eq. 5) as before except for those constrained columns recognized in Step 1. They are supplied with the coupling matrix  $\mathbf{I}_c$  and vector  $\mathbf{b}_c$  as in [OH95]:

$$\begin{aligned} \mathbf{I}_c(i, i) &= 1 \\ \mathbf{b}_c(i) &= s_i^t \end{aligned} \quad (20)$$



**Figure 5:** BOATS: Floating boats interact with water waves.

to count for the impact effects of the rigid object. Finally, in Step 3, the state of the rigid object is updated according to standard rigid body dynamics. The water pressure to the rigid object is calculated when calculating the rigid object impact during Step 2.

This coupling approach assumes the rigid object to be small and floating on the water, so it cannot handle completely immersed objects so far. We also assume that the contact region between the rigid object and water is nearly convex due to the height field representation limit.

## 7. Interactive Fluid Control

Since our system has little restriction on external forces, we can interactively control fluid shapes by specifying various external control forces. We provide two techniques to implement such external control forces  $\vec{f}_{ext}$ .

The first method uses external pressures calculated from distance maps to a target fluid shape. For example, the pressure field in the SCA example (Fig. 10) uses the Euclidean distance. To put it another way, if we express the control pressure field as a terrain height field, control shapes are terrain valleys and the water flows from peaks to valleys as expected. Because distance maps are  $C^1$  continuous in most places, those control forces are smooth and continuous in most places as well. As expected, this method is comparatively stable and robust. It is faster when the target shape is static, since reconstructing the distance map (by a fast marching algorithm in our practice) can be expensive.

Instead of constructing a pressure field, the second method modifies the fluid velocity directly by external accelerations. We create these accelerations from cursor motions, similar to some digital painting tools. We also clamp the magnitude of these synthetic accelerations in order to prevent instability caused by arbitrary user inputs. Fig. 6 is a screenshot captured from our interactive shape design system. This method is straightforward and efficient.

Name	Type	Resolution	Gravity $g$ ( $m/s^2$ )	Surface Tension $\gamma$ ( $N/m$ )	Viscosity $\tau$	Simulation Speed (frames per sec)
BUNNY	Curved Surface	160,000	9.8	0.073	0.5	CPU: 11.0
GLASS	Curved Surface	160,000	9.8	0.073	0.5	CPU: 4.1
POOL	Regular Grid	$400 \times 400$	9.8	0.073	0.3	CPU: 4.3
WINDOW	Regular Grid	$400 \times 400$	9.8	0.073	0.3	GPU: 9.2, CPU: 11.0
SPHERES	Coupling	$400 \times 400$	9.8	0	0	GPU: 1.2, CPU: 0.4
DRIFT	Coupling	$1,200 \times 250$	9.8	0	0	GPU: 2.0, CPU: 0.8
BOATS	Coupling	$400 \times 400$	9.8	0	0	GPU: 3.3, CPU: 1.3
INTERACTIVE	Accel. Control	$400 \times 400$	0	0.073	0.3	CPU: 20
SCA	Pressure Control	$400 \times 400$	9.8	0.073	0.3	GPU: 4.2, CPU: 3.0

**Table 1:** Simulation statistics. When available, we list simulation speeds for both CPU and GPU solvers (rightmost column).



**Figure 6:** INTERACTIVE: Fluid control by external accelerations due to the motion of a user's cursor.

## 8. Matrix Solver and Data Structures

We take advantage of the surface mesh connectivity to store the sparse GSWE matrices. Diagonal elements are stored at vertices, and off-diagonal elements are stored at mesh edges between each vertex and its 1-ring neighbors. We use directional edges when the matrix is not symmetric. We also create *hyper* edges that form connections between vertices and their 2-ring neighborhoods in order to store the matrix required by the complete surface tension scheme.

To solve the symmetric positive definite matrix system defined by the implicit gravity scheme and the incomplete surface tension scheme, we use the preconditioned conjugate gradient method with either a Jacobi preconditioner or a modified incomplete Cholesky decomposition preconditioner. We also implemented the preconditioned Bi-Conjugate Gradient Stabilized method (Bi-CGSTAB) with modified incomplete LU decomposition to test the complete surface tension scheme.

The 2D regular grid matrix solver is implemented on both CPUs and GPUs (with a Jacobi preconditioner), in order to

improve the performance as Table 1 shows. Details about GPU implementation can be found in [BFGS03]. The extension to a GPU matrix solver for curved surfaces should be straightforward, but remains as future work.

## 9. Results and Discussion

(Please check the accompanying video for more results.) Our system was implemented using all the techniques described in the previous sections. We tested a variety of applications using this system on a Dell<sup>TM</sup> XPS 700 PC with Intel Core2Extreme 2.92Ghz CPU and Dual Geforce 7950 graphics card. Test statistics are listed in Table 1. We use real-world units and parameters, including the water density  $\rho$ , which is  $1.0 \times 10^3 kg/m^3$  at  $3.98^\circ C$ . The frame rate for each example is measured in the worst case with maximum simulation burden. Table 1 shows that the GPU matrix solver runs faster on large bodies of water, such as the coupling examples. On the other hand, the CPU solver can better take advantage of sparse water, like the examples with water drops.

Compared with particle systems or grid systems that usually require several minutes to generate a single frame under similar resolution, most of our applications run at interactive rates and some of them run in real time. Compared with previous height field systems, the general system has a slightly higher computational cost for several reasons. Firstly, the overhead to handle a general system is small but not negligible. Secondly, more iterations are needed to solve the matrix system with a surface tension matrix  $\mathbf{A}_s$ , especially when the surface tension coefficient  $\gamma$  is large. The two-way coupling algorithm introduces more iterations too, since constrained cells cause the whole matrix to be less diagonally dominant. This is clearly shown in Table 1 from the SPHERES example and the DRIFT example. Although SPHERES uses lower resolution, the DRIFT example runs twice as fast since it involves less constrained cells than SPHERES.

Interestingly, when gravity is in the same direction as the surface normal, the gravity pressure component  $g_i$  in Eq. 9

becomes negative, which causes a less diagonally dominant matrix. Eventually with sufficiently strong gravity, the matrix is failed to be positive definite and its solution becomes unpredictable. Since the influence of  $\mathbf{A}_g$  on the matrix system depends on the water depth, ideally we can remove redundant water from the height field in order to keep the system solvable. This scenario corresponds to water drops dripping from the bottom of a solid surface, after the amount of accumulated water goes above some threshold.

The WAVE example shown in Fig. 10 illustrates water waves due to a raised column of water. Our result matches with dispersion relations in that waves under both gravity and surface tension will disperse faster than waver under gravity only. It should be noted that shallow wave assumption implies that the wave length should be much larger than the water depth, which means the system theoretically cannot exactly represent Kelvin waves in deep ocean or capillary waves with wave lengths  $< 1.7cm$ . Fortunately, we noticed from our examples that when the water depth is close to the wave length, shallow waves still provides a reasonable and fast approximation to gravity-capillary waves for graphics applications, like ripples on a small pool.

One limitation of our GSWE approach is that water columns may intersect if the surface is concave. We eliminate high-frequency curved regions by using averaged normals and the background height field  $b(x)$ , but this cannot entirely remove all concavities. We have not come across problems from concavities in our examples, but this issue will arise for some geometric configurations.

We provided GPU rendering with single reflection and refraction during online simulation for all of our applications. Some examples were supported by offline GPU rendering with multi-sampling to remove aliasing artifacts. We also used an offline photorealistic ray tracer to produce high-quality videos for some examples.

## 10. Conclusion and Future Work

We have presented a new physically based framework to efficiently simulate small-scale 3D water flows on solid surfaces under shallow wave assumptions. This framework is governed by the General Shallow Wave Equations (GSWE) and it is based on a height field representation. We have developed several techniques within this framework including an implicit gravity scheme, an implicit surface tension scheme and two-way fluid/rigid body coupling. Our experiments show that our system is fast, stable, straightforward to implement on both CPUs and GPUs. The approach is flexible enough to produce multiple effects such as water waves, rivulets, water drop effects, fluid/rigid body coupling and interactive control and shape modeling.

Looking into the future, our short-term plan includes developing a GPU matrix solver for curved surfaces and incorporating our algorithm into an interactive media painting sys-

tem. In the long term, we would like to study how to combine height field techniques with particle systems and grid systems so that non-height-field effects can be efficiently produced as well. Motion reduction of the height field provides another possibility to explore in the future.

## Acknowledgments

This research was funded by Adobe Systems Inc., an NVIDIA fellowship, and by NSF grant CCF-0625264.

## References

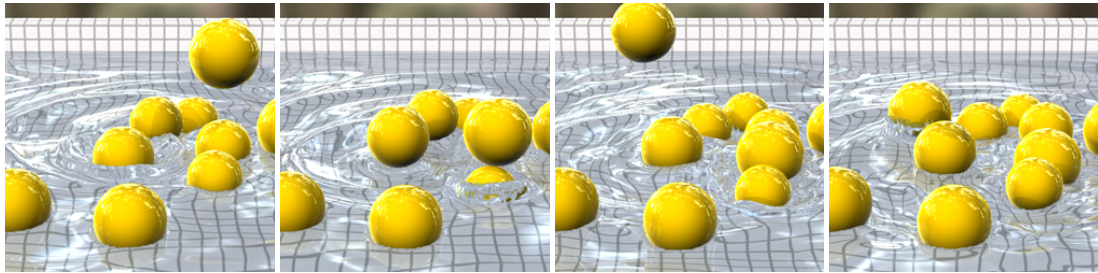
- [BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖDER P.: Sparse matrix solvers on the GPU: conjugate gradients and multi-grid. *ACM Trans. Graph.* 22, 3 (2003), 917–924.
- [CdVLHM97] CHEN J. X., DA VITORIA LOBO N., HUGHES C. E., MOSHELL J. M.: Real-time fluid simulation in a dynamic virtual environment. *IEEE Comput. Graph. Appl.* 17, 3 (1997), 52–61.
- [CM04] COHEN J. M., MOLEMAKER M. J.: Practical simulation of surface tension flows. In *SIGGRAPH Sketches* (2004), p. 70.
- [EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. In *Proc. of ACM SIGGRAPH '02* (2002), pp. 736–744.
- [ETK\*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.* (2007).
- [FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. of SIGGRAPH '01* (2001), pp. 23–30.
- [FHP98] FOURNIER P., HABIBI A., POULIN P.: Simulating the flow of liquid droplets. In *Graphics Interface* (1998), pp. 133–142.
- [FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. In *Proc. of SIGGRAPH '04* (2004), p. 441.
- [FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process.* 58, 5 (1996).
- [FOK05] FELDMAN B. E., OAR'BRIEN J. F., KLINGNER B. M.: Animating gases with hybrid meshes. In *Proc. of ACM SIGGRAPH '05* (2005).
- [GSD05] GRINSPUN E., SCHRÖDER P., DESBRUN M.: Discrete differential geometry: An applied introduction. SIGGRAPH Course #14, 2005.
- [Hec97] HECKBERT P.: Fast surface particle repulsion. SIGGRAPH '97: New Frontiers in Modeling and Texturing Course, pages 95–114, 1997.
- [HK03] HONG J.-M., KIM C.-H.: Animation of bubbles in liquid. *Comp. Graph. Forum (Eurographics Proc.)* 22, 3 (2003), 253–262.
- [HNB\*06] HOUSTON B., NIELSEN M. B., BATTY C., NILSSON O., MUSETH K.: Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Trans. Graph.* 25, 1 (2006), 151–175.



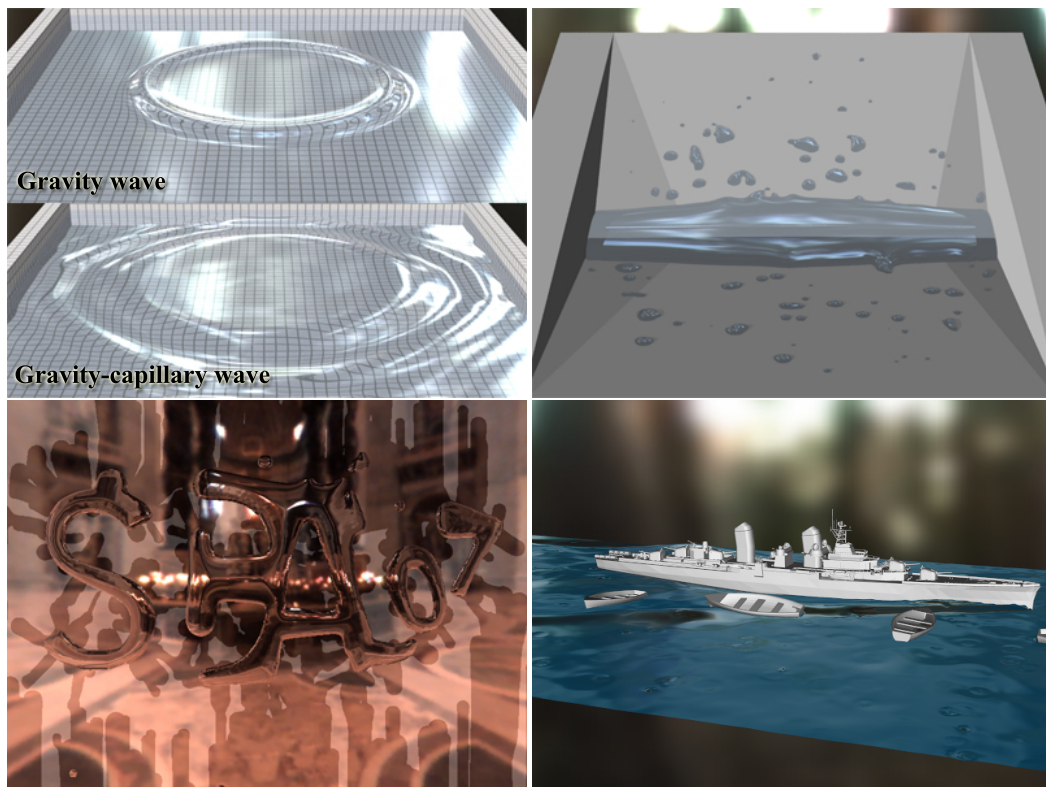
- [HNC02] HINSINGER D., NEYRET F., CANI M.-P.: Interactive animation of ocean waves. In *Proc. of SCA '02* (2002), pp. 161–166.
- [IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *Proc. of ACM SIGGRAPH '06* (2006), p. 38.
- [KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. In *Proc. of ACM SIGGRAPH 2006* (2006).
- [KLLR07] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Advections with significantly reduced dissipation and diffusion. *IEEE TVCG* (2007).
- [KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. In *Proc. of ACM SIGGRAPH '90* (1990), pp. 49–57.
- [LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *Proc. of SIGGRAPH '04* (2004), no. 3, pp. 457–462.
- [LvdP02] LAYTON A. T., VAN DE PANNE M.: A numerically efficient and stable algorithm for animating water waves. *The Visual Computer* 18, 1 (2002), 41–53.
- [MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. of SCA '03* (2003), pp. 154–159.
- [MMS04] MIHALEF V., METAXAS D., SUSSMAN M.: Animation and control of breaking waves. In *Proc. of SCA '04* (2004), pp. 315–324.
- [Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics* 30 (1992), 543–574.
- [MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *Proc. of SIGGRAPH '04* (2004), pp. 449–456.
- [NHS02] NEYRET F., HEISS R., SENEGAS F.: Realistic rendering of an organ surface in real-time for laparoscopic surgery simulation. *the Visual Computer* 18, 3 (may 2002), 135–149.
- [NM06] NIELSEN M. B., MUSETH K.: Dynamic tubular grid: An efficient data structure and algorithms for high resolution. *Journal of Scientific Computing* 26, 3 (2006).
- [OF02] OSHER S., FEDKIW R.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.
- [OH95] O'BRIEN J. F., HODGINS J. K.: Dynamic simulation of splashing fluids. In *Computer Animation '95* (1995), pp. 198–205.
- [Set99] SETHIAN J.: *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.
- [SSK05] SONG O.-Y., SHIN H., KO H.-S.: Stable but nondissipative water. *ACM Trans. Graph.* 24, 1 (2005).
- [Sta99] STAM J.: Stable fluids. In *Proc. of ACM SIGGRAPH '99* (1999), pp. 121–128.
- [Sta03] STAM J.: Flows on surfaces of arbitrary topology. In *Proc. of SIGGRAPH '02* (2003), vol. 22, pp. 724–731.
- [SY04] SHI L., YU Y.: Inviscid and incompressible fluid simulation on triangle meshes. *Comput. Animat. Virtual Worlds* 15, 3-4 (2004), 173–181.
- [SY05] SHI L., YU Y.: Controllable smoke animation with guiding objects. *ACM Trans. Graph.* 24, 1 (2005).
- [TLP06] TREUILLE A., LEWIS A., POPOVIĆ Z.: Model reduction for real-time fluids. *Proc. of ACM SIGGRAPH '06* 25, 3 (2006), 826–834.
- [TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. In *Proc. of SIGGRAPH '03* (2003), vol. 22, pp. 716–723.
- [Tur91] TURK G.: Generating textures on arbitrary surfaces using reaction-diffusion. In *Proceedings of ACM SIGGRAPH '91* (1991), pp. 289–298.
- [UJ04] UIJTTEWAAL W. S., JIRKA G. H.: *Shallow Flows: Research Presented at the International Symposium on Shallow Flows, Delft, Netherlands*. A.A. Balkema, 2004.
- [WMT05] WANG H., MUCHA P. J., TURK G.: Water drops on surfaces. In *Proc. of ACM SIGGRAPH '05* (2005), pp. 921–929.



**Figure 7:** WINDOW: Water drops are flowing on the window panel with a white waterproof boundary “water”.



**Figure 8:** SPHERES: Spheres cause waves and splashes in the height field. They have different volumes but identical mass.



**Figure 9:** Several effects generated by our system. Examples are: WAVE, POOL, SCA and DRIFT, which demonstrate wave dispersion, water drops on a glass, user-controlled fluid shapes, and fluid/rigid body interaction.