FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

Animation of Large Scale Open Water Phenomena (submitted to ACM SIGGRAPH 2006 for review)

Nils Thürey and Ulrich Rüde and M. Stamminger

Lehrstuhlbericht 06-3

Animation of Large Scale Open Water Phenomena (submitted to ACM SIGGRAPH 2006 for review)

Nils Thürey [*]
Institute for System Simulation
University of Erlangen-Nuremberg

Ulrich Rüde
Institute for System Simulation
University of Erlangen-Nuremberg

Marc Stamminger
Institute for Computer Graphics
University of Erlangen-Nuremberg

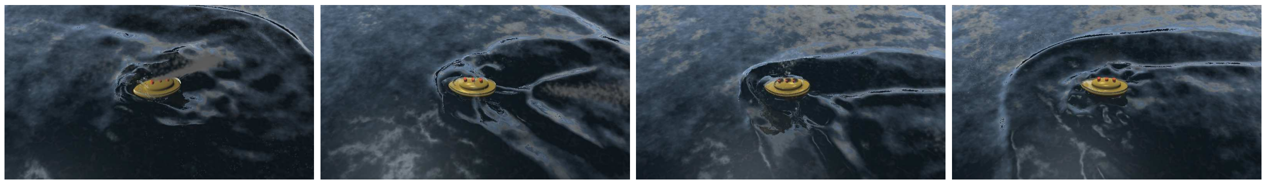

Figure 1: While the object moves along the fluid surface, it is always at the center of the 3D computation region. The outer waves are computed by a shallow water simulation.

Abstract

The goal of this paper is to perform large scale fluid simulations that capture the effects of all necessary scales – from small drops of fluid up to the propagation of large waves. To achieve this, we present a novel hybrid simulation method, that couples a two-dimensional shallow water simulation with a full three-dimensional free surface fluid simulation. We explain how to parametrize the shallow water solver according to the parameters of a 3D simulation. Each simulation is used to initialize double layered boundary conditions for the other one. The area covered by the 2D region can be an order of magnitude larger than the 3D region without significantly effecting the overall computation time. The 3D region can furthermore be easily moved within the 2D region during the course of the simulation. To achieve realistic results we combine our simulation method with a physically based model to generate and animate drops. For their generation we make use of the fluid turbulence model, and animate them with a simplified drag calculation. This allows simulations with relatively low resolutions.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Animation; I.6.3 [Computing Methodologies]: Simulation and Modeling—Applications;

Keywords: physically based animation, free surface fluids, lattice Boltzmann methods, shallow water simulation, dispersed flow simulation

1 Introduction and Related Work

In previous years, the physically based animation of liquids has seen significant progress. Especially the work of Ron Fedkiw [Fedkiw et al. 1999] , his group [Enright et al. 2002] [Guendelman et al. 2005], and Jos Stam [Stam 1999] have enabled the realistic and efficient simulation of liquids with a free surface. A typical animation to show the quality of a simulation is pouring liquid into a glass. Nowadays the simulations recreating these effects are hard to distuingish from reality. In this paper we will, however, focus on scenes that have a significantly larger scale, e.g. a ship traveling through the ocean. The challenge is to capture both the large scale movement of the water around the ship, as well as the splashing of the waves around it, including the drops and spray. The overall flow pattern has a characteristic size of several meters, while a typical drop of water is only several millimeters in size. As all phenomena from small drops up to large waves contribute to the visual appearance, they have to be captured to achieve a realistic representation of such a scene.

For all classes of algorithms that are used to simulate free surface flows, the problem is that the amount of computational work and the required resources grow significantly when the resolution of the simulation is increased. The full simulation of the above mentioned scene with a volume-of-fluid Navier-Stokes solver would hardly be possible even on large supercomputers. Adaptive techniques can be used to alleviate this problem to some extent [Losasso et al. 2004], but usually increase the complexity of a solver and have limits in their ability to speed up the computational time. In the following, we will describe a different approach that computes the full fluid flow only in a bounded region of interest, and uses a fast two-dimensional fluid simulation to compute the fluid surface around it. We also choose not to simulate the whole depth of the fluid, from the free surface to the bottom (e.g. the ocean floor), but only an upper layer of fluid. The small scale details such as drops are simulated as particles with a simplified, yet physically based, algorithm.

The contributions of this paper are:

- a new hybrid simulation method that couples a two-dimensional shallow water simultion with a full three-dimensional free surface simulation, and
- a physically based model for the creation and simulation of drops.

Although we use the lattice Boltzmann method to solve the two- and three-dimensional fluid flows, variants of both our hybrid method and the drop model could also be applied to other types of fluid solvers.

[*]e-mail: nils.thuerey@cs.fau.de

In its different forms the *Navier-Stokes* (NS) equations have long been used for physically based animation. [Kass and Miller 1990] were the first to use *shallow water simulations* in computer graphics. As of today, this simplified model, which assumes depth averaged fluid properties, is still a research topic e.g. for computations performed on the GPU [Hagen et al. 2005]. The use of a three-dimensional NS discretization with free surface boundary conditions was demonstrated by [Foster and Metaxas 1996]. By today, various methods exist to directly solve or approximate the NS equations. Apart from the level set based solvers mentioned above, *volume-of-fluid* (VOF) methods [Hirt and Nichols 1981] are commonly used [Mihalef et al. 2004], or, from the background of astro-physics, the *smoothed particle hydrodynamics* (SPH) solvers [Keiser et al. 2005].

In the following we will use the *lattice Boltzmann method* (LBM), which originates from discrete compressible gas simulations [Frisch et al. 1987]. The LBM approximates the NS equations without the need for an iterative solver by relaxing the incompressibility constraint [He and Luo 1997]. It has been used in various areas, e.g. for single phase fluid computations [Wei et al. 2003], or in combination with a VOF model to simulate free surface flows [Thürey and Rüde 2004]. They also explain how typical limitations of VOF methods can be overcome within the LBM algorithm. As the shallow water equations represent an advection-diffusion problem similar to the full NS equations, they can likewise be solved with the LBM. A derivation of the appropriate changes to the basic algorithm can be found in [Dellar 2001].

A different approach to detailed and accurate fluid simulation solvers, that we will make use of in Section 4, can be found in the area of chemical engineering. For cases such as bubble column reactors, Eulerian-Lagrangian simulations of these dispersed multiphase flows, e.g. large numbers of bubbles in a relatively coarse fluid flow simulation [Delnoij et al. 1999], are used to understand and optimize the physical processes [Buwa et al. 2005]. These methods simulate bubbles with a spherical shape, and model the forces caused by the turbulent fluid around them. Apart from level set methods, where particles are used to accurately track the free surface, [Takahashi et al. 2003] also use particles to add small scale details. But in contrast to our approach they generate drops based of the surface curvature, and apply linear damping to model air resistance. Recently, simulations of large scale open water scenes were used in TV and feature film productions, e.g. by [Scanline-Production and Trojanski 2005], however, without giving detailed information about the models used.

2 Animation of the Water Surface

Within a certain region of interest, e.g. a moving ship, we perform a full three-dimensional simulation of the free surface fluid. We will briefly describe the algorithms used to perform the three- and two-dimensional fluid simulations, and then describe the coupling of both in more detail.

Free Surface Simulation: For solving the fluid phase in the three dimensional region we use the *D3Q19* lattice Boltzmann model shown in Figure 2. This model requires an equidistant grid (with a cell size of Δx) and 19 floating point values, one for each velocity vector, that are stored in each cell. These *distribution functions* DFs represent a small amount of fluid moving with the corresponding velocity. The basic algorithm proceeds in two steps - first the advection of the fluid molecules is handled by copying the DFs to the neighboring cell along their velocity direction. Once this is done for all cells in the grid, each again has a full set of DFs, which is

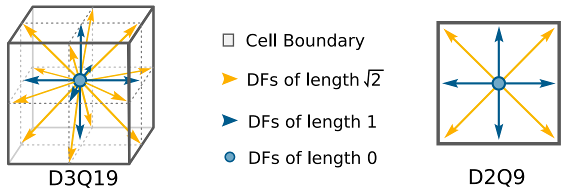

Figure 2: The lattice Boltzmann grid models used for the free surface (D3Q19) and the shallow water simulations (D2Q9).

used to calculate the macroscopic properties of the fluid – the density ρ and the velocity $\mathbf{u}$. In the following, f_i will denote one of the nineteen DFs, with $\mathbf{e}_i$ being the corresponding velocity vector. Its inverse direction is $\mathbf{e}_{\tilde{i}}$, thus $\mathbf{e}_{\tilde{i}} = -\mathbf{e}_i$.

$$\rho = \sum f_i \qquad \mathbf{u} = \sum \mathbf{e}_i f_i \ . \tag{1}$$

Density and velocity are needed to calculate the equilibrium DFs with

$$f_i^{eq}(\rho,\mathbf{u}) = w_i \left[\rho + 3\mathbf{e_i}\cdot\mathbf{u} - \frac{3}{2}\mathbf{u}^2 + \frac{9}{2}(\mathbf{e_i}\cdot\mathbf{u})^2 \right] . \tag{2}$$

Here w is a weight that depends on the length of the velocity vector: $w_i = 1/3$ for $i = 1$, $w_i = 1/18$ for $i = 2,...,7$, and $w_i = 1/36$ for $i = 8,...,19$. The equilibrium DFs from Eq. 2 are necessary to compute the collisions that occur between the molecules in a real fluid. With the LBM these are calculated by relaxing the DFs towards the equilibrium DFs depending on the fluid viscosity with:

$$f_i(\mathbf{x},t+\Delta t) = \ (1-\omega) f_i'(\mathbf{x},t+\Delta t) \ + \ \omega f_i^{eq} . \tag{3}$$

Here the relaxation parameter ω is calculated from the time step of the simulation Δt, the physical visocity ν' and the lattice viscosity ν with

$$\omega = \frac{1}{3\nu} + \frac{1}{2} \quad , \nu = \nu' \frac{\Delta t}{\Delta x^2} \ . \tag{4}$$

The free surface model we apply only simulates the fluid phase. Hence, DFs that would be streamed from the gas phase have to be reconstructed in cells at the interface with

$$f_{\tilde{i}}'(\mathbf{x},t+\Delta t) = f_i^{eq}(\rho_A,\mathbf{u}) + f_{\tilde{i}}^{eq}(\rho_A,\mathbf{u}) - f_i(\mathbf{x},t) \ , \tag{5}$$

where ρ_A is the reference density of the air, hence, in our case $\rho_A = 1$. These free surface boundary conditions do not include surface tension, but ensure an undisturbed movement of the fluid in the gas phase. The mass flux between cells is computed directly from the DFs that are streamed from and to interface cells. This mass value m is additionally stored for each cell, and can be used to determine the fluid fraction $\varepsilon = m/\rho$ of a cell. Thus, this value determines how much of a cell is filled with fluid. An example for a free surface implementation with LBM can e.g. be found in [Körner et al. 2005]. At the boundary of the 3D domain we use boundary conditions with a height dependent pressure and a velocity given by the shallow water simulation. In Section 3 this will be described in more detail.

Turbulence Model: A problem of the basic LBM is, that it quickly becomes instable for lower viscosities and thus values of ω close to 2. To alleviate this, turbulence models are often applied. We make use of the commonly used *Smagorinski* turbulence model [Smagorinski 1963], that locally increases the fluid viscosity in regions where unresolved flow features are detected, to model the energy dissipation of small scale vortices [Hou et al. 1996]. The

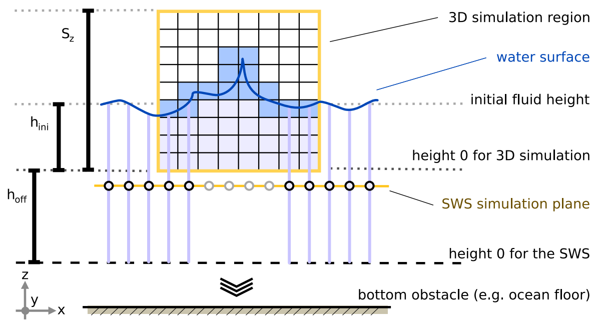

Figure 3: This picture gives an overview of our hybrid simulation method. The full three-dimensional fluid flow is solved in a given region of interest (illustrated by a 2D rectangle), and coupled to a two-dimensional shallow water simulation (shown as a 1D line in the picture).

amount of local viscosity increase is determined by the Reynolds stress tensor, that is computed locally for each cell as

$$\Pi_{i,j} = \sum_{\alpha=1}^{19} \mathbf{e}_{\alpha_i} \mathbf{e}_{\alpha_j} \left(f_i - f_i^{eq} \right) . \tag{6}$$

This value is used to compute the correction factor

$$S = \frac{1}{6C^2} \left(\sqrt{\nu^2 + 18C^2 \sqrt{\Pi_{i,j}\Pi_{i,j}}} - \nu \right) . \tag{7}$$

with a user defined constant C, that we have set to 0.04. The modified viscosity of the cell is then calculated with

$$\omega_s = \frac{1}{3(\nu + C^2 S) + 1/2} , \tag{8}$$

and used instead of the normal value for ω in Eq. 3.

Shallow Water Simulation: The shallow water, or St. Venant, equations can likewise be solved using the LBM. In this case, instead of considering the fluid pressure, a height value is computed for each cell. Overall, the algorithm is very similar to the basic algorithm described above – both the streaming step and Eq.3 for relaxation towards the equilibrium are still valid. The equilibrium DFs to be used with Eq.3, however, are calculated differently. Furthermore, as the fluid surface is only two-dimensional, we use the *D2Q9* LBM model with nine velocities. To distinguish the DFs of the shallow water simulation from those of the three dimensional free surface simulation, we refer to them as g_l in the following.

The fluid height h and the fluid velocity for the shallow water simulation (SWS) are calculated as:

$$h = \sum_{l=1}^{9} g_l \qquad \mathbf{v} = \frac{1}{h} \sum_{l=1}^{9} \mathbf{e}_l g_l . \tag{9}$$

In contrast to the 3D LBM model, the velocity computation of the SWS requires a division by the height, as shown in Eq.9. With height and velocity, the equilibrium DFs are computed as

$$g_0^{eq}(h, \mathbf{v}) = h \left[1 - \frac{5}{6} Gh - \frac{2}{3} \mathbf{v}^2 \right] , \tag{10}$$

and

$$g_l^{eq}(h, \mathbf{v}) = w_l h \left[\frac{1}{6} gh + \frac{1}{3} \mathbf{e_l} \cdot \mathbf{v} + \frac{1}{2} (\mathbf{e_l} \cdot \mathbf{v})^2 - \frac{1}{6} \mathbf{v}^2 \right] , \quad \text{for } l = 1..9 . \tag{11}$$

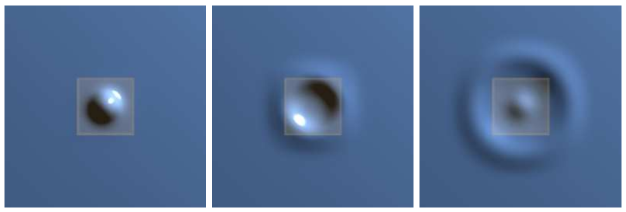

Figure 4: Wave propagation of a hemispherical drop on a flat surface using our algorithm. The 3D simulation region is highlighted in the middle.

Here G is the gravity force, normal to the two-dimensional plane of the SWS, and the weights w_l have the values $w_l = 1/18$ for $l = 2, ..., 5$, and $w_l = 1/36$ for $l = 6, ..., 9$. An in depth description of shallow water LBM can be found in e.g. [Zhou 2004]. To establish a fixed height of the SWS boundary, we set the cells there to have the equilibrium DFs for the initial height and a zero velocity.

As this shallow water solver is similar to a basic LBM solver, the Smagorinski turbulence model given by Eq.7 and Eq.8 can likewise be used to increase stability. The only difference is that $\Pi_{i,j}$ is now computed as a sum over the nine DFs of a SWS cell with Eq. 6. To ensure stability for varying velocities, we furthermore apply the adaptive time stepping as described in [Thürey et al. 2005] to both simulations.

3 Hybrid 2D/3D Simulation

An overview of our hybrid simulation approach is given in Fig. 3. Both algorithms have been parametrized to solve the same fluid simulation problem, and are then coupled at a interface region. In the following we will assume that the SWS is performed in the xy-plane, and the gravitational force acts in the direction of the negative z-axis.

There is an inherent difference between the two simulation approaches that has to be overcome: the derivation of the SWS assumes a depth averaged velocity and has a coupling between fluid height and velocity. The 3D simulation, on the other hand, can have a velocity varying along the z-axis, and has boundary conditions (see below) that makes it independent of the initial height of the fluid surface h_{ini}. In order to be able to couple both simulations, we have developed the following parametrization procedure for the SWS. It ensures that the SWS and the 3D simulation have the same wave propagation speed.

A first step towards this goal is to offset the SWS by a constant factor h_{off}, which we choose as $h_{\text{off}} = S_z - h_{\text{ini}}$, as shown in Fig. 3. Here S_z is the height of the 3D domain in cells. Thus, the SWS height has an initial value of S_z independent of h_{ini} in the 3D simulation. Now the gravity force has to be scaled to synchronize the wave propagation speed in both simulations. This is done by examining the behaviour of the SWS properties. Given an arbitrary simulation setup, the properties of the fluid change by a factor given in Table 3, when the value of the parameter in the first column is multiplied by 2. Thus, given the initial SWS fluid height and $n = log_2(h_{\text{off}} + h_{\text{ini}})$ we set the SWS gravity G to match the given offset for the simulation resolution. It is computed from the z-component of the 3D gravity $\mathbf{g}_z$ as:

$$G = \mathbf{g}_z \cdot \left(\frac{e}{2} \right)^{-n} . \tag{12}$$

The 3D gravity is given by the physical value, usually $G' = 9.81\ [m/s^2]$, as $\mathbf{g} = (0, 0, G' \cdot \Delta t^2 / \Delta x)$. The initial time step size

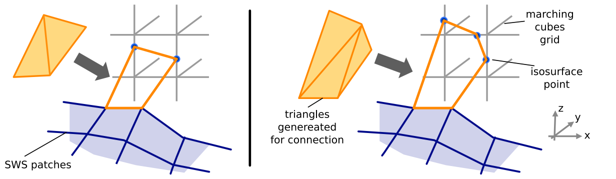

Figure 5: The two cases that need to be distinguished to generate a closed surface mesh for the 3D and shallow water simulations.

Δt is set according to the maximum fluid or moving obstacle speed in the simulation setup. Now, when transferring velocities in the xy-plane between the simulations, the influence of the offset and gravity scale have to be removed. According to Table 3 this is accomplished by

$$\mathbf{v}_{x,y} = s_u \, \mathbf{u}_{x,y} \text{ , with } s_u = \frac{\sqrt{2}^{\,n}}{\sqrt{s_g}^{\,1/n}} \; . \tag{13}$$

3D to 2D Coupling: Here, we determine the height of the fluid at a position within the 3D simulation region by searching for the first interface cell. We start at the cell with grid position $(i, j, 0)$, and assume a planar fluid surface. Hence, the fluid height is computed for the first interface cell at (i, j, k_{H}) with

$$H(i,j) = k_{\mathrm{H}} + \varepsilon(i,j,k_{\mathrm{H}}) + h_{\mathrm{off}} \; . \tag{14}$$

To determine the velocity of the fluid at the water surface $\mathbf{u}_{\mathrm{avg}}$, we average three cells at the 3D fluid surface:

$$\mathbf{u}_{\mathrm{avg}} = \sum_{l=0}^{2} \frac{\mathbf{u}(i,j,k_H - l)}{3} \; . \tag{15}$$

This removes small temporal deviations at the surface, when cells are reinitialized during the free surface handling. To transfer the information from the 3D simulation to the SWS a cell at (i, j), marked with X in Fig. 6, is initialized with the equilibrium DFs:

$$g_l = g_l^{eq}\Big(H(i,j), s_u \, \mathbf{u}_{\mathrm{avg}}\Big) \; . \tag{16}$$

The cells where height and velocity are set with Eq. 16 represent the inner boundary for the SWS. Further inwards we perform the full 3D simulation, thus the cells of the SWS do not have to be updated in this region, as their values are never used. To ensure a transfer with as few disturbances as possible, we use a double layered transfer. Thus, we use a second type of boundary condition for the region of SWS cells directly outwards of the boundary cells described above, marked with O in Fig. 6. For the cells that are updated according to Eq. 16 all DFs are reset each time step, while for the second boundary layer we only rescale the existing DFs to match the required fluid height:

$$g_l^* = g_l \cdot \frac{H(i,j)}{h(i,j)} \; . \tag{17}$$

The combination of these two boundary conditions ensures a correct transfer of both fluid surface height and velocity.

2D to 3D Coupling: The transfer from the SWS to the 3D simulation, is done by initializing the 3D cells to represent the SWS height and velocity. For a 3D cell at (i, j) marked with O in Fig. 6 we thus either remove cells from the simulation, if $H(i, j) > h(i, j)$, or otherwise add new ones. To correctly initialize the new cells we

	G	N	Δt	$\mathbf{v}$
G	2	1	$\sqrt{2}$	$\sqrt{2}$
N	1	2	e	$\sqrt{2}$

Table 1: Behavouir of the SWS upon parameter change. Here N is the number of SWS cells along the x- or y-axis.

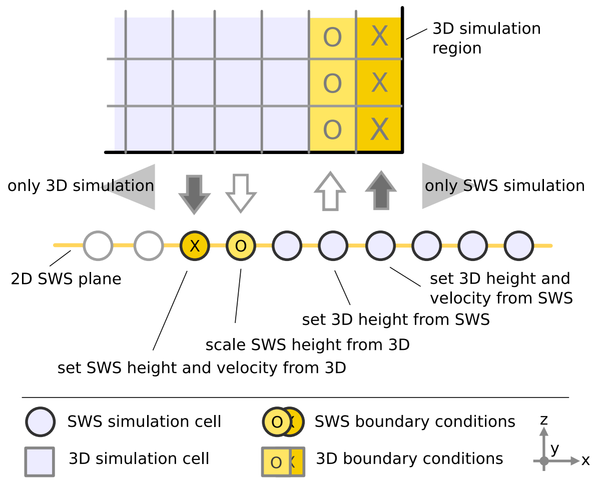

Figure 6: Detail of the double layer boundary conditions in the overlapping interface region.

directly use the velocity of the SWS. The z coordinate of the velocity is calculated from the SWS fluid heights of the current and previous timestep.

For the outer layer of the 3D simulation (cells marked with X in Fig. 6) we use velocity boundary conditions with a fixed pressure. For the LBM the pressure of a cell with height k in the domain is given by:

$$\rho_k = 1.0 + (h_{\mathrm{ini}} - k) \cdot g_z \cdot \; -3\omega \; . \tag{18}$$

The pressure thus increases further down in the grid, with a gradient that depends on the relaxation time ω. The velocity can again be taken directly from the SWS, as described above. Note that it is in this case not necessary to scale the SWS velocities, as we set the whole height of the 3D simulation These boundary conditions, however, do not ensure the full propagation of arbitraty waves generated in the SWS region, as this would require a wave profile initialization ...? Although these boundary conditions do not enforce mass conservation for the transfer, this is not problematic, as the overall height of the fluid height is kept at the initial value by the SWS with its outer boundary conditions.

The depth of the overlapping region for the two simulations is variable, but we have found a distance of one eighth of the 3D domain size to yield good results. A validation run is shown in Fig. 4. The circular wave retains its shape while it is transferred from the highlighted 3D region to the SWS region. The in- and outflow at the 3D domain boundary furthermore causes no disturbances of the flow field. A pure SWS simulation would not have been able to resolve the drop forming in the middle of the 3D region, visible in the right picture of Fig. 4.

Surface Generation: To generate a mesh from the 3D simulation, we use the marching cubes algorithm [Lorensen and Cline 1987]. A triangulation of the SWS surface is easiliy computed by constructing patches between four adjacent SWS cells that have x and y coordinates according to their grid position, and a z coordinate

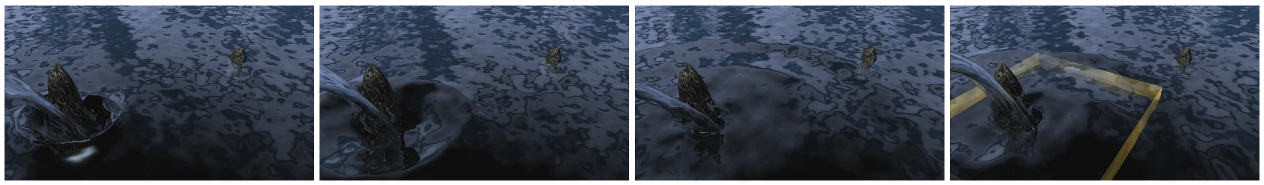

Figure 8: A stream of water hits a rock and a water surface In the rightmost picture the interface between 2D and 3D region is highlighted.

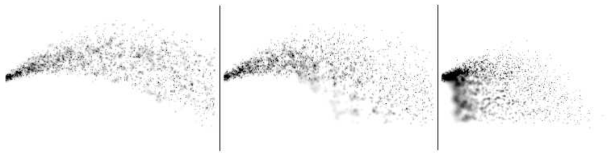

Figure 7: Effect of the drag model for different size scales. The simulations were parametrized to represent scales of $10cm$,$1m$ and $10m$, from left to right. The smaller particles, are slowed down, and cause mist below the outflow.

given by $h(x,y)$. At the 3D domain boundary we leave out the first row of SWS cells, and construct triangles to connect the 3D mesh to the SWS patches. If both points of a marching cubes cell lie on its z-edges, this is sufficient to ensure a closed mesh. For all other cases, we also have to connect triangles to the points above or below the cell at the surface, as shown in Fig. 5. In rare cases, e.g. when a drop directly hits the connection line, this technique will not result in a closed mesh. For our tests we have, however, not encountered any visible artifacts. In the interface region, where we have full information from both simulations, we linearly blend the fluid surface heights, to achieve a smooth transition from one type of simulation to the other. As the mesh generated from the fluid fractions already requires smoothing, we also perform a smoothing of the interface region to prevent any artifacts from misaligned normals.

4 Animation of Drops

For the animation of drops in our simulations we make use of methods developed for dispersed gas-liquid flows. In the following we will describe our model for the animation and generation of water drops. Each drop is described by its position $\mathbf{x}$, velocity $\mathbf{w}$ and radius r. We assume that the drops are small enough to remain spherical due to surface tension. Thus using the density of water ρ_W the mass of a drop is given by its volume:

$$m_P = \rho_W \frac{4}{3}\pi r^3 \ . \qquad (19)$$

Generation: To generate particles in our simulation we make use of the turbulence model explained in Section 2. As it already determines how many unresolved flow features a given cell has, we use it to compute a particle generation probability for cells at the fluid interface. We compute this probability from the absolute value of the Reynolds stress tensor given by Eq. 6 and the physical speed $\mathbf{u}' = \mathbf{u}\ \Delta x/\Delta t$. The stress tensor usually takes values of ca. $P_m = 10^{-2}$ for regions with significant unresolved detail independent of the actual grid resolution. We assume that the range of velocities, where the pressure of the surrounding air causes instabilities that lead to drops at the surface, is similar to that of the drop terminal velocities, which motivates the following probability function:

$$p_D = P_{ij} * (\mathbf{u}')^2 \quad \text{with} \quad P_{ij} = \sqrt{\Pi_{ij}\Pi_{ij}} \ . \qquad (20)$$

Thus for a high physical speed of $10[m/s]$ and significant unresolved flow details this function will result in a drop generation probablity close to one. Note that the calculation of the Reynolds stress tensor is especially easy for LBM, as it is computed locally from the derivative information contained in the non-equilibrium parts of the DFs (see Eq. 6). For other types of solvers, this computation will require access to neighboring grid cells to compute the derivatives. Upon creation we initialize the drop velocity by the fluid velocity and a randomized normal offset to avoid immediate collision with the fluid surface.

Animation: For each LBM step, we simply update the particle position using their velocities and the LBM time step length:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t \mathbf{w} \ . \qquad (21)$$

To update the particle velocities, we compute the balance of the forces acting upon it:

$$m_P \frac{d\mathbf{w}}{dt} = F_G + F_D \ . \qquad (22)$$

where F_G is the force due to gravity and F_D is the drag force caused by the drop of water moving through the air. In contrast to the dispersed flow simulations mentioned above, we thus ignore any lift the drops might experience as well as other forces that would e.g. be caused by the density gradient in the air. The lift is proportional to the ratio between the involved fluids, which is close to zero for air and water. Likewise, we assume a density gradient in the air very close to zero.

F_G is computed from gravity and particle mass:

$$F_G = m_P \ \mathbf{g} \ , \qquad (23)$$

while the computation of the drag force requires more effort. The movement of water drops through the air has been studied in depth for meteorological purposes, see e.g. [Pruppacher and Klett 1997]. From these studies it is known, that rain drops usually have a size less than $4.5mm$. Above this size they will start to deform during their movement and eventually break apart due to the high forces from the air in comparison to the surface tension. It was furthermore measured, that these large drops have a terminal velocity of up to $w_{t1} = 9m/s$, while smaller drops of with e.g. $r = 0.5mm$ only accelerate to ca. $w_{t2} = 2m/s$.

Given a coefficient of drag C_D, the drag force acting upon a particle is calculated as:

$$F_D = \frac{C_D}{2}\ \rho_L\ \pi\ r^2\ \mathbf{w}_{\text{rel}}|\mathbf{w}_{\text{rel}}| \ , \qquad (24)$$

where $\mathbf{w}_{\text{rel}}$ is the relative velocity of the particle. It is computed from the velocity of the air $\mathbf{w}_A$ by $\mathbf{w}_{\text{rel}} = \mathbf{w}_A - \mathbf{w}$. As we do not

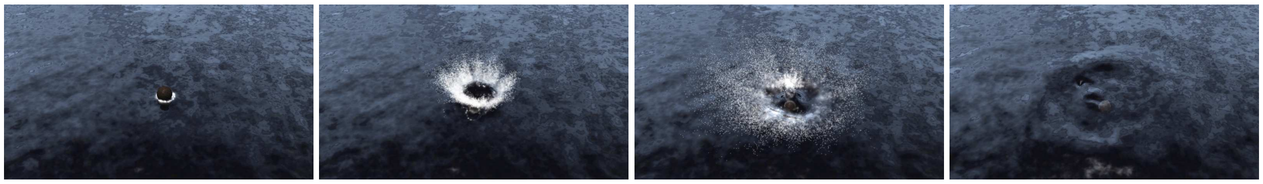

Figure 9: Animation of a spherical object hitting a water surface.

explicitly simulate the gas phase, we usually set $\mathbf{w}_A = 0$. Other values could be used to simulate the effect of wind. For the drag coefficient there are various approximations for different regimes of turbulence. As the case of a larger spherical rain drop at terminal velocity is already turbulent ($Re > 1000$), and the approximations are computationally very expensive, we fit the computation of the drag coefficient to yield values in the required range. We thus require that the drag force and gravity acceleration balance for the drops at their respective terminal velocities. Assuming a linear change for both parameters, we compute the drag force with a bilinear interpolation:

$$C_D = \frac{|\mathbf{w}_{rel}|}{w_{t2} + (w_{t1} - w_{t2}) w_r} \left(\frac{1}{2} + \frac{1}{2} w_r\right) . \qquad (25)$$

with $w_r = (r_1 - r)/(r_1 - r_2)$. For our simulations we have limited the size of the drops to the range of $r_1 = 0.005m$ to $r_2 = 0.0005m$. After the computation of F_G and F_D we update the velocity according to Eq. 22 with an Euler-step:

$$\mathbf{w}(t + \Delta t) = \mathbf{w}(t) + (F_G + F_D) m_P \Delta t . \qquad (26)$$

Additional effects: To actually cause a disintegration of a thin fluid sheet into drops, we randomly choose a size r_D in the given range, and subtract the mass of the drop from the interface cell where it was generated. For simulations representing a large scale, this could result in huge numbers of particles – for these cases we subtract a multiple of the drop mass from the cell, and display the drop as a correspondingly larger transparent particle, thus representing multiple drops of similar size. Once the drop hits the fluid surface again, we add the mass that was subtracted before. Here, similar to [Takahashi et al. 2003], foam particles could be generated and tracked with the surface velocities. Fig. 7 shows examples of the drag force influence for different scales. For the larger test cases the higher velocities result in higher drag forces, resulting in a noticeable slowdown of the smaller drops.

Another effect that cannot be directly simulated with the algorithm explained in Section 2, is that of instabilities caused purely by the relative physical velocity of the fluid $\mathbf{u}'_{\text{rel}}$, as the air is not simulated as a fluid itself. Thus, in order to cause these instabilities, we use a simple approximation, and manually add the following term

$$f_i = f_i + (P_m - P_{ij}) \cdot w_i \frac{\mathbf{u}'_{\text{rel}}}{50} \cdot \mathbf{e}_i , \qquad (27)$$

for cells, with $P_{ij} < P_m$ that do not generate particles.

5 Results

To demonstrate the capabilities of our approach, we have extended the fluid solver of the Open-Source 3D application Blender with our hybrid algorithm. *(Note to the reviewers - please try the appropriate version from the accompanying archive files, each containing a README file with further usage information.)* All results shown in the following were created with this implementation using a physical viscosity of water $\nu_W = 1 \cdot 10^-6$. To enhance the realism, we add a small scale bump map to the water surface, giving the impression of smaller chaotic waves. The accompanying animation gives examples without this effect.

	3D	SWS	Simulation	Size
Fig. 8	120^3	480^2	14.6 s	0.2 m
Fig. 9	150^3	600^2	24.6 s	10 m
Fig. 10	150^3	900^2	102.6 s	2 m

Table 2: Here the grid resolutions for the simulation test cases, together with average simulation times per frame can be seen (measured with an 2.2 GHz Opteron CPU). The size value is the physical length of a side of the 3D domain used for parametrization of the simulations.

A test case of our simulation method is shown in Fig. 8. A stream of water hits a rock and a water surface. The created waves spread outwards without a visible border between the SWS and the 3D simulation. Simulation resolutions and times can be found in Table 5. A test case that demonstrates the capabilites our drop model is shown in Fig. 9. A spherical object is dropped into a fluid surface. The drop model, with up to K drops at a single time step, enhances the impression of a large simulation scale. Here limitations of the raytracer become apparent – the drops and mist below a water surface are not shaded, and there is no self shadowing of the drops. However, these issues will hopefully be resolved in an upcoming release of the program.

Given a working hybrid simulator, only small changes are necessary to achieve animations such as shown in Fig. 10. Here we move the 3D domain according to the position of an object in the xy-plane. For each movement of the domain by Δx we copy the values stored in the grid by one in the desired direction. During the next step, the boundary regions of both simulations will again be correctly initialized for the boundary conditions.

For a simulation run with a relatively large SWS domain, such as shown in Fig. 10, we have measured the workload distribution between the different parts of the algorithm. In this case, ca. 68.8% are spent on the simulation of the 3D region. The 2D region, covering a 35 times larger area, requires 24.9% of the time, while ca. 2.6% are spent on the coupling of both simulations. The remaining 3.7% were spent on drop calculation, surface mesh generation and initialization. The update of an SWS cell is on average three times faster than the update of a 3D cell.

6 Conclusions and Outlook

We have presented an algorithm to efficiently simulate large scale open water fluids. This is achieved by coupling a shallow water

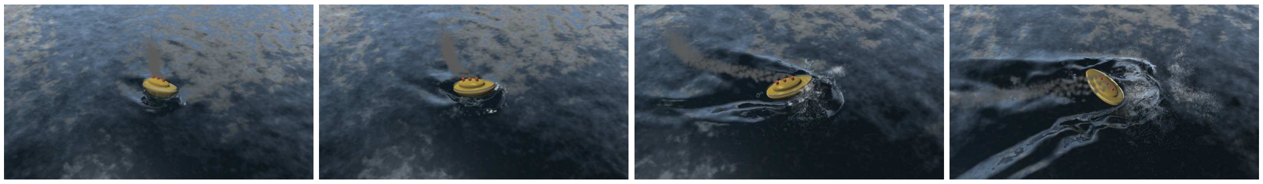
Figure 10: Pictures from an animation with a moving 3D domain.

simulator with a 3D free surface simulation. Our approach to model the formation and movement of drops allows us to add detail to relatively coarse fluid simulations. In conclusion, animations of large water surfaces, that would require huge amounts of memory and computational time with a conventional full 3D simulation, can be calculated within a matter of hours. The chance to arbitrarily move the 3D domain within the 2D region furthermore increases the flexibility of our approach.

We plan to increase the overall computational speed by making use of adaptive grids and parallelization for machines with multiple CPUs. For an intermediate scale, we are working on applying the methods from SPH to the generated drop particles, to accurately capture effects such as coalescence. It would furthermore be interesting to add a model for the generation of drops in the SWS region as well, or couple it with an FFT solver for ocean waves [Tessendorf 2004]. An easy way to further speed up the computations would be to reduce the SWS resolution by an integer factor, and interpolate the values at its boundary. Finally, the method could be used to couple multiple regions of three dimensional computation in one large water surface simulation. Given enough computational resources in combination with low grid resolutions, this could be used to simulate interactive environments with large water surfaces e.g. for virtual reality applications.

References

BUWA, V., GERLACH, D., AND DURST, F. 2005. Regimes of bubble formation on submerged orifices. *Phys. Rev. Letters*.

DELLAR, P. J. 2001. Non-hydrodynamic modes and a priori construction of shallow water lattice Boltzmann equations. *Phys. Rev. E 65*.

DELNOIJ, E., KUIPERS, J. A. M., AND VAN SWAAIJ, W. P. M. 1999. A three-dimensional CFG model for gas-liquid bubble columns. *Chemical Engineering Science 54*.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and Rendering of Complex Water Surfaces. *Proceedings of ACM SIGGRAPH*.

FEDKIW, R. P., ASLAM, T., MERRIMAN, B., AND OSHER, S. 1999. A non-oscillatory Eulerian approach to interfaces in multimaterial flows. *J. of Comp. Phys. 152*.

FOSTER, N., AND METAXAS, D. 1996. Realistic Animation of Liquids. *Graphical Models and Image Processing 58*.

FRISCH, U., D'HUMIÈRES, D., HASSLACHER, B., LALLEMAND, P., POMEAU, Y., AND RIVERT, J.-P. 1987. Lattice Gas Hydrodynamics in Two and Three Dimensions. *Complex Systems 1*, 649–707.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling Water and Smoke to Thin Deformable and Rigid Shells. *Proceedings of ACM SIGGRAPH 24(3)*.

HAGEN, T. R., HJELMERVIK, J. M., LIE, K.-A., NATVIG, J. R., AND HENRIKSEN, M. O. 2005. Visual simulation of shallow-water waves. *Simulation Modelling Practice and Theory 13*.

HE, X., AND LUO, L.-S. 1997. A Priori Derivation of Lattice Boltzmann Equation. *Phys. Rev. E 55*.

HIRT, C. W., AND NICHOLS, B. D. 1981. Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries. *J. of Comp. Phys. 39*.

HOU, S., STERLING, J. D., CHEN, S., AND DOOLEN, G. 1996. A Lattice Boltzmann Subgrid Model for High Reynolds Number Flow. *Fields Institute Communications 6*, 151–166.

KASS, M., AND MILLER, G. 1990. Rapid, Stable Fluid Dynamics for Computer Graphics. *Proceedings of ACM SIGGRAPH*.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRE, P., AND GROSS, M. 2005. A Unified Lagrangian Approach to Solid-Fluid Animation. *Proceedings of the Eurographics Symposium on Point-Based Graphics*.

KÖRNER, C., POHL, T., RÜDE, U., THÜREY, N., AND ZEISER, T. 2005. Parallel Lattice Boltzmann Methods for CFD Applications. In *Numerical Solution of Partial Differential Equations on Parallel Computers*, A. Bruaset and A. Tveito, Eds., vol. 51 of *LNCSE*. Springer, 439–465.

LORENSEN, W., AND CLINE, H. 1987. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. In *Computer Graphics Vol. 21, No. 4*, 163–169.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating Water and Smoke With an Octree Data Structure. *Proceedings of ACM SIGGRAPH 23*.

MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2004. Animation and control of breaking waves. *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation (SCA)*.

PRUPPACHER, H. R., AND KLETT, J. D. 1997. *Microphysics of Clouds and Precipitation*. Springer.

SCANLINE-PRODUCTION, AND TROJANSKI, S., 2005. Flowline - Hydro Fluid Dynamics for Visual Effects. `www.flowlines.info`.

SMAGORINSKI, J. 1963. General circulation experiments with the primitive equations. *Mon. Wea. Rev. 91*.

STAM, J. 1999. Stable Fluids. *Proceedings of ACM SIGGRAPH*.

TAKAHASHI, T., FUJII, H., KUNIMATSU, A., HIWADA, K., SAITO, T., TANAKA, K., AND UEKI, H. 2003. Realistic Animation of Fluid with Splash and Foam. *Computer Graphics Forum 22 (3)*.

TESSENDORF, J. 2004. Simulating Ocean Surfaces. *SIGGRAPH 2004 Course Notes 31*.

THÜREY, N., AND RÜDE, U. 2004. Free Surface Lattice-Boltzmann fluid simulations with and without level sets. 199–208. Workshop on Vision, Modelling, and Visualization VMV.

THÜREY, N., POHL, T., RÜDE, U., OECHSNER, M., AND KÖRNER, C. 2005. Optimization and Stabilization of LBM Free Surface Flow Simulations using Adaptive Parameterization. *Computers and Fluids* (December).

WEI, X., ZHAO, Y., FAN, Z., LI, W., YOAKUM-STOVER, S., AND KAUFMAN, A. 2003. Natural phenomena: Blowing in the wind. *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation SCA 2003* (July).

ZHOU, J. G. 2004. *Lattice Boltzmann Methods for Shallow Water Flows*. Springer.