SPH with Small Scale Details and Improved Surface Reconstruction

Juraj Onderik* Comenius University †
Michal Chládek‡ Comenius University †
Roman Ďurikovič§ Comenius University †

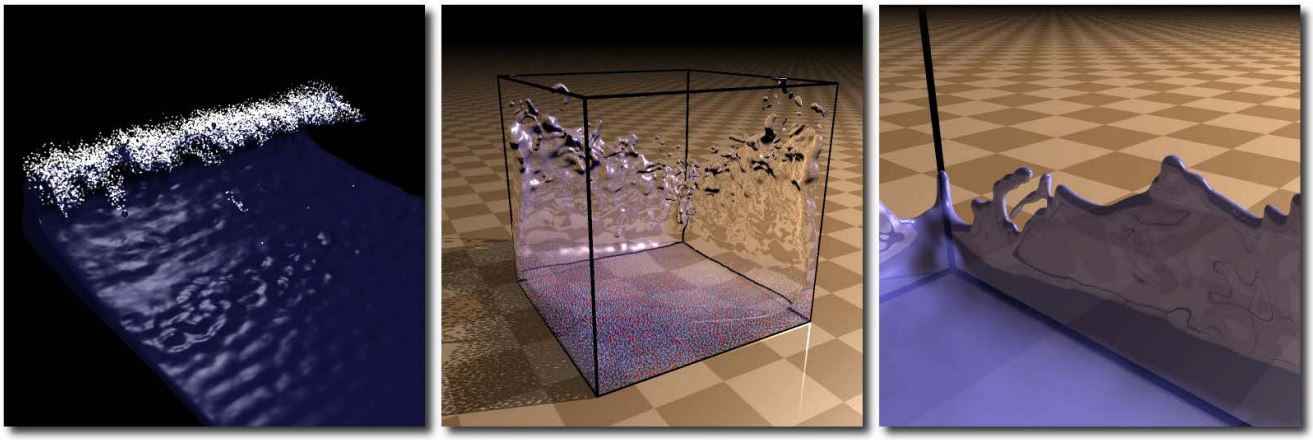

Figure 1: Left: *Breaking wave simulation (only 20k particles) with 15k splash particles emitted using our technique.* Middle: *Dam break test in water tank (50k particles). Only red particles are sorted during coherent neighbor search.* Right: *Our improved surface reconstruction creates better thin regions (14k particles).*

Abstract

We present a novel method for creating small scale details as splashes and foam for SPH simulations. In our technique, each fluid particle can become a source emitter of splash particles. The probability of emission is controlled by density decay and velocity of fluid particles. Splash particles are uncoupled, collide only with obstacles and follow only ballistic motion.

Due to coherency of fluid motion, spatial sorting of particles usually performed in neighbor search can be done faster than using regular sorting algorithm. We provide a simple 3 step (split-sort-merge) approach to reorder particles in approximately 2/3 time of a regular sort. In the first step we split particles into a sorted and an unsorted group, then we reorder the small unsorted group with a regular sort and finally we merge both sorted groups into a new particle list.

We have improved fluid surface reconstruction techniques [Zhu and Bridson 2005; Solenthaler et al. 2007] to better handle uneven distributions of particles. Our method first computes density distribution of particles which is then used for weighting the particle average similarly to previous methods. We also propose a different approach to reduce artifacts within isolated particles, without the need of eigen analysis, giving us a clean analytic expression of surface normals.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism;

Keywords: Small Scale Details, Splash, Neighbor Search, Coherency, Surface Reconstruction, Uneven Distribution

1 Introduction and Related Work

Modeling small scale details can greatly improve visual realism of large fluid simulations. Although splashes and foam can be successfully achieved with both Eulerian and Lagrangian techniques, the simulation domain must be enormously large. This is prohibitive for artists when prototyping such phenomena. Simplified techniques as shallow water simulation are still superior when modeling large bodies of water as oceans and rivers [Chentanez and Müller 2010]. Since they are unable to directly model small scale details, a number of works has focused on extending these techniques with uncoupled spray, bubble and foam particles[O'Brien and Hodgins 1995; Takahashi et al. 2003; Cleary et al. 2007]. When large, highly turbulent, splashing scenes (e.g. dam breaks, floods) are desired, these simplified methods fail. As a solution we provide in section 2 a combination of full 3D SPH fluid simulation with uncoupled splash particles to generate small scale details.

When simulating fluids with SPH or any of its variants WCSPH [Müller et al. 2005] or PCISPH [Solenthaler and Pajarola 2009] searching for neighbor particles is an essential part of the overall algorithm. The quadratic naive approach is exaggerative, because SPH particles have a short bounded influence. Since neighbor search is basically a simplified version of geometric range queries a number of techniques has been already presented. With respect to the fluid simulation majority of works use regular spatial decom-

*e-mail: juraj.onderik@fmph.uniba.sk
†Faculty of Mathematics, Physics and Informatics, Comenius University, 842 48 Bratislava, Slovakia http://www.fmph.uniba.sk
‡e-mail: michal.chladek@fmph.uniba.sk
§e-mail: roman.durikovic@fmph.uniba.sk

SCCG 2011, Vinicné, Slovak Republic, April 28 – 30, 2011.
© 2013 ACM 978-1-4503-1978-2/11/0004 $15.00

position of the simulation space. Spatial hashing has been used to model unbound domains [Teschner et al. 2003]. To prevent cache misses particles can be ordered using spatial indices along some space-filling curve. Multi-core CPU [Ihmsen et al. 2011] and parallel GPU [Goswami et al. 2010] approaches using a z-curve implementation exist. Although motion coherency has been exploited to maintain locality list over several time steps[Verlet 1967], using a coherent sort of spatial indices has not been yet provided, see Section 3.

Many applications including fluid animation require realistic visualization of smooth surfaces covering some point cloud. Usually an iso-surface is defined with respect to the particle data. This can be either converted to a polygonal mesh using Marching Cubes algorithm [Rosenberg and Birdwell 2008] or directly visualized using recent high quality volume rendering [Fraedrich et al. 2010]. Due to modern GPUs, point splatting is also a comparable alternative, mainly after a screen space post-processing [van der Laan et al. 2009]. Accurate iso-function definition is crucial when naturally looking surfaces are desired. Within the SPH context several methods based on density [Müller et al. 2003], center of mass [Zhu and Bridson 2005; Solenthaler et al. 2007], particle redistancing [Adams et al. 2007] and kernel anisotropy [Yu and Turk 2010] were presented. In the Section 4 we provide an improved surface reconstruction method which removes known artifacts and handles non-uniform particle distributions.

2 Small Scale Details for SPH

When modeling turbulent fluids with Eulerian methods, small scale details as splashes are usually considered as problematic since excessively large grid size is required. Although adaptive subdivision has been successfully employed[Losasso et al. 2004] a majority of previous works focused on exploiting simplified models of splashes and foam. Particle systems are a natural choice for modeling such phenomena in combination with almost any fluid simulation technique.

Figure 2: Left: *SPH fluid only.* Middle: *Splash Particles.* Right: *SPH with splash particles. The scene consists of 20k fluid particles and 14k splash particles.*

A simple extension to shallow water simulation using particle based splashes has been proposed in [O'Brien and Hodgins 1995]. In [Takahashi et al. 2003] spray particles improved the visual impact of breaking waves. Advanced bubbling and frothing effects have been achieved in [Cleary et al. 2007]. Recently [Chentanez and Müller 2010] presented a complex real-time simulation with several small scale details.

Particles have been used in numerous works (e.g. [Müller et al. 2005; Clavet et al. 2005]) to simulate full dynamics of fluids. Since particle based fluids naturally model splashing effects, less attention has been taken to extend even these methods with small scale details. Indeed, by combining an uncoupled particle system with a coupled SPH simulation one can achieve better results for large scale animations using only moderate number of fluid particles, see Figure 2.

2.1 Emitting Splash Particles

Our simulation technique consist of bulk water modeled with regular SPH *fluid particles* governed either by WCSPH [Müller et al. 2003] or PCISPH [Solenthaler and Pajarola 2009] and uncoupled spray particles which represent splashes or foam. During the SPH simulation each fluid particle can become a source for emitting spray particles based on the following observations.

A compact fluid volume is a subject for splashing mainly due to a collision with some obstacle or when it is free-falling and the air friction causes it to split into drops. Within the SPH framework this can be loosely expressed with density decay near isolated particles or the free surface. During compression, when particles approach each other, their estimated density increases causing pressure forces to push them away. During splashing or collisions with obstacles, these forces can be strong enough to remove some surface particles from the bulk mass. While they get isolated, density strongly decreases giving us a good criterion for marking them source emitters for spray particles. However, measuring derivative of density is not a good idea here since relative change of density can be large due to pressure oscillations inside fluid volume as well.

We propose a simple mechanism based on the concept of *splashing probability* (from 0 to 1). Suppose some fluid particle has a full (100%) probability of splashing. We define the *maximal emit rate* as the total number of splash particles the fluid particle can emit per second. This allows us to calculate how many particles can be maximaly emitted within one simulation step. Instead of emitting all of them, we first proceed a simple stochastic test. For each splash particle we generate a uniform random number (from 0 to 1) and emit the particle only if the number is greater than the current splashing probability of the fluid particle. This gives us a consistent way to control the splash generation using only splashing probability.

We compute splashing probability using a criterion based on density decay. For each particle we first calculate its density ratio $\frac{\rho_i}{\rho_0}$. If it is under a *critical density ratio* α_d we say the fluid particle has a nonzero splashing probability. Formally the splashing probability of i-th particle based on density decay is

$$P_i^d = \beta_d \left(\alpha_d - \frac{\rho_i}{\rho_0} \right), \tag{1}$$

where β_d is *splash density factor* which controls how strong the change in density affects the splashing probability.

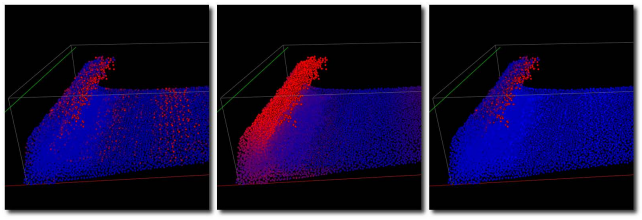

Figure 3: *Parameters used to control splashing probability. Red particles represent larger probabilities.* Left: *Density based splashing probability.* Middle: *Velocity of particles.* Right: *Final splashing probability composed from density decay and velocity.*

As shown in the Figure 3 (left) this approach greatly captures particles on the tip of the wave. Some particles on the resting surface

have low density as well, giving them higher splashing probability as well. This is, however, not desired since such particles are slowly floating on the free surface without making any splashes. We control this by constraining the splashing probability with the velocity of the particle, see Figure 3 (middle). Simple modulation with velocity length did not provide plausible results since we want to completely cut off particles with slow motion. We define splashing probability based purely on velocity as

$$P_i^v = \beta_v \left(|\mathbf{v}_i| - \alpha_v \right), \tag{2}$$

where again α_v is a *minimal splash velocity* which serves as a threshold and β_v is the *splash velocity factor* which controls how much the velocity changes the splashing probability.

Simple modulation of P_i^d and P_i^v can give positive probability also when both P_i^d and P_i^v are negative. This is the case of slow particles inside bulk fluid, which should definitely not make any splashes. Therefore we define the final splashing probability P_i of i-th particle as

$$P_i = \begin{cases} P_i^d \cdot P_i^v & P_i^d > 0 \ \wedge \ P_i^v > 0 \\ 0 & \text{otherwise.} \end{cases} . \tag{3}$$

Each emitted splash particle is first initialized with the position and velocity of its source particle adding small jitter to make it more realistic. Its life time is set to the *initial life time*. In every simulation step the its life time is decremented by the delta time until it becomes negative, making the particle dead, i.e. ready for another emission.

2.2 Negative Pressure Correction

It is important to note here, that we have modified the SPH pressure calculation to solve compression only and almost freely allow expansion. In standard SPH pressure is usually related to density using state equation

$$p(\mathbf{p}) = \frac{k\rho_0}{\gamma} \left(\left(\frac{\rho(\mathbf{p})}{\rho_0} \right)^{\gamma} - 1 \right), \tag{4}$$

which can create negative pressures on isolated particles and near the boundary. Negative pressures push particles to each other creating artificial compressions. This is usually an unwanted effect which causes numerical instabilities. In case we want to model surface tension a small force pushing particles in the direction of negative pressure can be useful. Therefore we use the following modified pressure expression

$$p^*(\mathbf{p}) = \begin{cases} \lambda p(\mathbf{p}) & p(\mathbf{p}) < 0 \\ p(\mathbf{p}) & p(\mathbf{p}) \geq 0 \end{cases} , \tag{5}$$

where λ is a *negative pressure scaling factor*, which should by zero if we want to completely diminish artificial compression or a very small, positive number when modeling surface tension.

3 Coherent Neighbor Search

Vast majority of proposed SPH simulation projects use grids as the spatial acceleration data structure for an efficient neighbor search. Simulation domain is usually divided into a regular grid where each cell contains a list of particles being inside this cell. Assuming the particle interaction radius is less than or equal to the cell size, we need to process during neighbor search only particles within the current cell and all its neighboring cells (total 27 cells in 3D). This greatly simplifies the searching process, however we must be able to construct and update the grid data structure efficiently. This can be done easily in linear time using spatial hashing [Teschner et al. 2003]. When dealing with larger data sets, hash table must be large enough to keep the number of hash collisions low. Moreover when constructing an exact neighbor lists we have to access particles in the neighbor cells of the current particle using hash a table. This makes the memory access scattered therefore cache unfriendly.

We overcome this problem by reordering particles in memory to pertain better spatial locality. Assuming each cell of the grid can be indexed using some space-filling curve, we can assign each particle the spatial index and sort them. Recently z-curve has been successfully applied [Ihmsen et al. 2011; Goswami et al. 2010] using a hash function in the sorted particle list. Following plain cell index one can efficiently traverse all neighboring cells (27 cells in 3D) in a memory coherent fashion completely removing any binary search or hashing [Onderik and Ďurikovič 2008].

All proposed methods however rely on a fast and effective sorting of particles using their spatial index. Fast linear-time radix sort has been successfully adopted. However, sorting all of these indices from scratch each frame is not very efficient.

3.1 Coherent Sort of Spatial Indices

Assuming spatial and temporal coherence of the fluid motion we can do a simple observation of the spatial grid data structure update. During each simulation step particles change their location very subtly. Majority of them retain their spatial index unchanged from frame to frame. Specifically, in highly dynamic scenes only about 10% of all particles change their spatial index (i.e. leave one cell and enter another). In more resting scenes about 3% of particles change their cell. Excluding these 3%-5% of particles, indices of the resting 90%-97% particles will retain in a sorted order after the simulation step, see Figure 4.

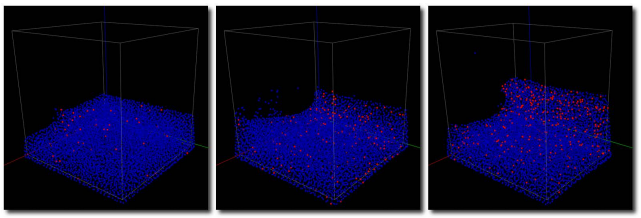

Figure 4: *Indices of red (blue) particles were changed (unchanged).* Left: *A resting scene where only 0.9% of indices changed.* Middle: *An average scene where 3.3% of indices changed.* Right: *A turbulent scene where 5.1% of indices changed.*

These observations lead us to a simple algorithm of coherently sorting spatial indices. In first pass we sweep particles (sorted in previous frame) and split them into 2 separate groups. The first group contains particles that did not change their index from the previous frame and thus all indices are already sorted. The second group contains particles with changed indices. These particles must be reordered using some efficient sorting algorithm e.g. radix sort. Now both groups of indices are sorted and we can merge them within a single sweep into the final sorted list.

Coherent sort can be summarizes as

1. *Split:* Split particles into a sorted and an unsorted group.
2. *Sort:* Sort the (smaller) unsorted group using e.g. radix sort.
3. *Merge:* Merge both sorted groups into final particle list.

See Section 5 for a detailed comparison with incoherent sort and the achieved speedup.

3.2 Coherent Sorting of Cell Indices

Coherence ensures that particles do not move extremely fast. Moreover we can assume that every particle either stays in its cell or moves *only* to one of its very neighboring cell.

When we use a plain cell index for ordering, sorted list can be constructed in a coherent fashion without using any explicit sorting algorithm. This is because the cell indexing preserves order with respect to the neighbor cells. Given two cells $C_1 = (i_1, j_1, k_1)$ and $C_2 = (i_2, j_2, k_2)$ having cell indices

$$\begin{array}{rcl} index(C_1) & = & i_1 + Ij_1 + IJk_1 \\ index(C_2) & = & i_2 + Ij_2 + IJk_2, \end{array} \tag{6}$$

where $index(C_1) < index(C_2)$, we can see that indices of any respective neighbor cell of C_1 and C_2 preserve the same order. Assume D_1 (D_2) are some respective neighbor cells of C_1 (C_2), where

$$\begin{array}{rcl} D_1 & = & C_1 + (-1 \cdots +1, -1 \cdots +1, -1 \cdots +1) \\ D_2 & = & C_2 + (-1 \cdots +1, -1 \cdots +1, -1 \cdots +1). \end{array} \tag{7}$$

Then index differences between C_1 and D_1 ($\Delta_1 = index(D_1) - index(C_1)$) and between C_2 and D_2 ($\Delta_2 = index(D_2) - index(C_2)$) are the same ($\Delta_1 = \Delta_2$):

$$\begin{array}{rcl} \Delta_1 & = & \{-1,0,+1\} + \{-I,0,+I\} + \{-IJ,0,+IJ\} \\ \Delta_2 & = & \{-1,0,+1\} + \{-I,0,+I\} + \{-IJ,0,+IJ\}. \end{array} \tag{8}$$

This property of cell indices enables us to perform coherent reordering without explicitly sorting. Similarly, we first sweep the previously sorted particles and separate them into 27 (in 3D) groups. Each group collects particles moved to a respective neighbor cell. For example, if some particle moves to the bottom-right neighbor cell, i.e. its change of cell coordinates is (+1,-1,0), we assign it to the group with id 12 ($= 13 + 1 - 1 \cdot 3 + 0 \cdot 9$). Generally given a cell coordinate difference $\Delta = (\Delta_i, \Delta_j, \Delta_k)$ we assign particle to the group with id

$$\text{group}(\Delta) = 13 + \Delta_i + 3\Delta_j + 9\Delta_k. \tag{9}$$

Notice that group 13 will contain particles which did not change their cell index.

Due to the order preservation property of cell index, particles within each group will be in a sorted order. Finally, we simply merge all these 27 (in 3D) groups into the resulting sorted particle list. The overall algorithm needs to sweep the particles only twice (first split then merge).

4 Improved Surface Reconstruction

Generating a smooth surface from particles is crucial for high quality rendering of fluids. Due to major improvements possible with recent graphics hardware, real-time, screen-space based techniques become popular [van der Laan et al. 2009]. However, for high-end results, ray tracing implicit surfaces still plays an important role.

Various approaches has been already proposed throughout the literature. In [Müller et al. 2003] a color function is interpolated using SPH summation over neighbor particles. This generates a considerably bloby surface even for resting scenes, where a flat fluid surface is desired. Great improvement has been achieved [Zhu and Bridson 2005] by measuring the distance to a weighted average of neighbor particles. This approach suffers from various artifacts in concave regions and between isolated particles. This has been successfully corrected by [Solenthaler et al. 2007] using a distance decay function based on measuring the rate of change of the calculated center of mass. Based on weighted average of close particles, [Adams et al. 2007] introduced additional redistancing to achieve even flatter surfaces. Recently [Yu and Turk 2010] proposed different approach based on anisotropic scaling of kernel functions. They use PCA to estimate variance of neighboring particles and thus determine principal dimensions of scaled kernel function.

During our experiments we found that all three methods ([Müller et al. 2003], [Zhu and Bridson 2005], [Solenthaler et al. 2007]) generate an unnatural looking surface when particle distribution gets uneven. As shown in Figure 5, surface gets either bloby (row 1) or has artifacts within isolated particles (row 2) or provides unnatural shapes (row 3).

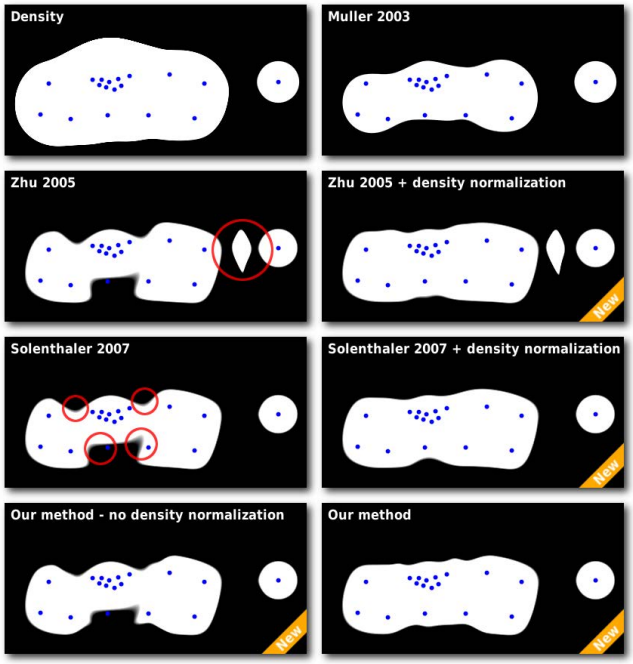

Figure 5: *Comparison of various iso-surface construction methods. First row shows SPH density distribution (left) and color function of [Müller et al. 2003] (right). Second row depicts distance based approach of [Zhu and Bridson 2005] (left) and our density normalized modification (right). Artifacts are shown in red circles. Third row contains corrected surface by [Solenthaler et al. 2007] (left) and our density normalization for uneven distributions (right). Finally fourth row provides our new approach (left) and its density normalized version (right).*

Similar to [Solenthaler et al. 2007] we define fluid surface as the zero level of the following *point-to-center* distance based implicit function

$$\phi(\mathbf{p}) = \|\mathbf{p} - \mathbf{C}(\mathbf{p})\| - Rf(\mathbf{p}). \tag{10}$$

Here $f(\mathbf{p})$ is a distance decay function (described later), R is the user defined distance of the surface from boundary particles and $C(\mathbf{p})$ is the *center* function which calculates for a given point $\mathbf{p}$ the weighted position average of neighboring particles.

As shown in figure 5 (first column) this approach can lead to unnatural surface deformations or even "ghost" regions lying outside of the fluid. As a solution we modify the center function by normalizing the weighted particle average using iso-density distribution

$(1/w_j)$ (second column)

$$\mathbf{C}(\mathbf{p}) = \frac{\mathbf{C}_1(\mathbf{p})}{C_2(\mathbf{p})} = \frac{\sum_j \frac{1}{w_j} W(\|\mathbf{p}-\mathbf{p}_j\|,h)\mathbf{p}_j}{\sum_j \frac{1}{w_j} W(\|\mathbf{p}-\mathbf{p}_j\|,h)}. \tag{11}$$

The *iso-density* w_j of j-th particle is calculated using standard SPH interpolation of unit mass of neighboring particles

$$w_i = \sum_j W(\|\mathbf{p}_i - \mathbf{p}_j\|,h), \tag{12}$$

where $W(\|\mathbf{r}\|,h)$ is a smooth kernel function. We use a simple polynomial kernel $W(r,h) = (1-\frac{r^2}{h^2})^3$ for this application. Notice the kernel support h used in surface reconstruction can be different to the physical interaction distance used in SPH. In our application we set the surface kernel support h to be 1.5x to 2x the length of the physical interaction radius.

This approach nicely overcomes problems with clustered particles, however artifacts near isolated particles still remain. As proposed in [Solenthaler et al. 2007] we can reduce these artifacts by modulating the surface distance R in equation 10 with a decay function $f(\mathbf{p})$, which is actually a factor that controls how close/far is the iso-surface from particles. Assuming two isolated particles, function $f(\mathbf{p})$ should be 1 for points close to the particles and should decay to zero in the middle between particles, scaling down artifact. This can be achieved by calculating the largest eigenvalue of the 3x3 Jacobian $\nabla C(\mathbf{p})$ and using it as a parameter that controls the amount of decay. Apart from the complexity of solving cubic equations to get eigenvalue of a 3x3 matrix, deriving an analytical expression of the surface normal (gradient of the surface) is even more complex.

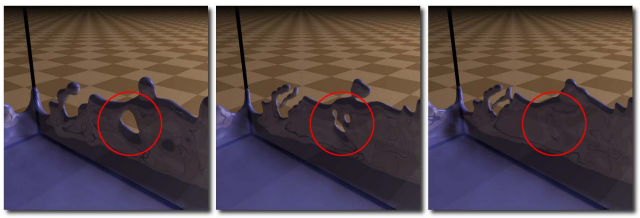

Figure 6: Left: *Thick and bloby approach by [Müller et al. 2003].* Middle: *Distance based method of [Zhu and Bridson 2005; Solenthaler et al. 2007]. Notice some holes in the thin regions.* Right: *Our density normalized method making thin regions more compact.*

Therefore we provide an alternative definition of the distance decay function based solely on position. Formally $f(p)$ is a composition of center function $\mathbf{C}(\mathbf{c})$, normalized iso-density $w(\mathbf{c})$ and transfer function $g(w)$

$$\begin{aligned} f(\mathbf{p}) &= g(w(\mathbf{C}(\mathbf{p}))) \\ w(\mathbf{c}) &= \sum_j \frac{1}{w_j} W(\|\mathbf{c}-\mathbf{p}_j\|,h) \\ g(w) &= \left(1 - \frac{(w-w_{\max})^2}{(w_{\max}-w_{\min})^2}\right)^2 \end{aligned} \tag{13}$$

For an arbitrary point $\mathbf{p}$ we compute $f(\mathbf{p})$ in three steps. First, we find the weighted center $\mathbf{c} = \mathbf{C}(\mathbf{c})$ using the center function $C(\mathbf{p})$. Next, we evaluate the normalized iso-density $w = w(\mathbf{c})$ at the center point $\mathbf{c}$. Finally, we smoothly transfer the normalized density into (0,1) interval using the transfer function $g(w)$ (see Figure 7 for more details).

The proposed decay function might seem complicated, however an analytical derivative $\nabla f(\mathbf{p})$ can be expressed using standard calculus. See Appendix 8 for more details about surface normal (gradient) derivation $\nabla\phi(\mathbf{p})$.

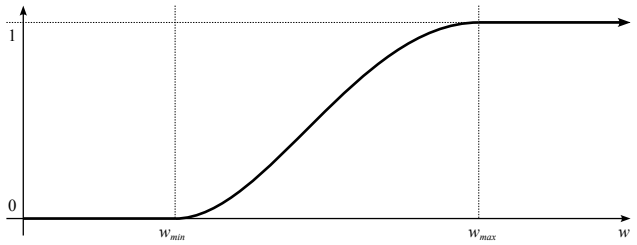

Figure 7: *The given density input w is smoothly modulated from zero to one based on user editable minimal $w_{\min}$ and maximal $w_{\max}$ density parameters, which control the final shape of the constructed iso-surface.*

5 Results

We have verified proposed methods with several testing simulations within our WCSPH/PCISPH framework and rendered using our direct implicit surface raytracer. Both applications were implemented in C++ targeting Intel x86 platform. Experiments have been performed several times with various parameters. Best results were achieved by setting particle support radius to $0.01m$, mass to $0.0002kg$, rest density to $1000kg/m^3$, viscosity to $0.4Ns/m^2$ and stiffness to $100Nm/kg$. Clamping negative pressures almost to zero together with a leap-frog integration highly increased the stability allowing us to perform even large scenes with a time-step up to $2ms$.

All simulations and rendering were performed on a mobile Core2 Duo T9600 2.8GHz with 4GB RAM. In the largest scenes (512k particles) physics calculations took around 1.5 second per frame, and raytracing (1024x1024) took about 20 seconds per frame. Smaller scenes (7.5k particles) together with raytracing (256x256) were processed at interactive rates (about 10 frames per second), see Table 1

Scene	Simulation	Raytracing	Total
WCSPH 27k	0.09s	1.1s	1.19s
WCSPH 64k	0.22s	2.5s	2.72s
WCSPH 125k	0.47s	4.3s	4.77s
WCSPH 512k	1.88s	19.3s	21.18s

Table 1: Execution times of several dam break simulations.

5.1 Index Sorting

We measured efficiency of the coherent neighbor search by executing several dam break simulations with various number of particles (10k to 500k). Each test has been performed multiple times with different restrictions on the density fluctuation (1% to 10%). Neither of these parameters however significantly affected the execution time of the particle sorting. Since coherent sorting splits particles into a sorted and an unsorted group based on the index change, we have measured the performance with respect to the new index ratio (the quocient of particle count with new index w.r.t. the total number of particles), see Figure 9.

As expected non-coherent radix sort is independent of index change. Coherent search using insert sort for a small unsorted

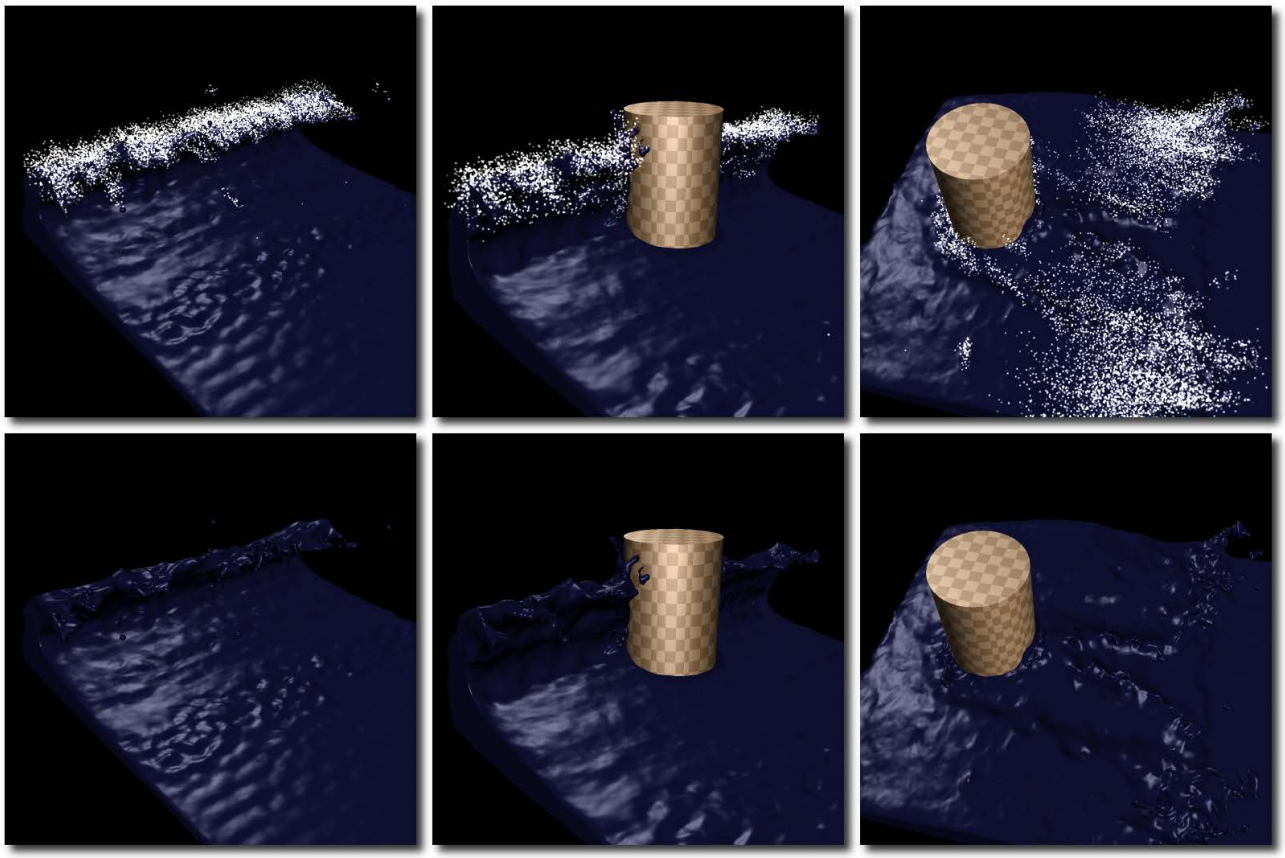

Figure 8: *Side by side comparison of a wave breaking and a splashing simulation with (top row) or without (bottom row) slash particles. Only 20k fluid particles together with 14k uncoupled splash particles were used. Desired surface blobyness was achieved by setting iso interaction radius to only 1.3x of SPH interaction radius.*

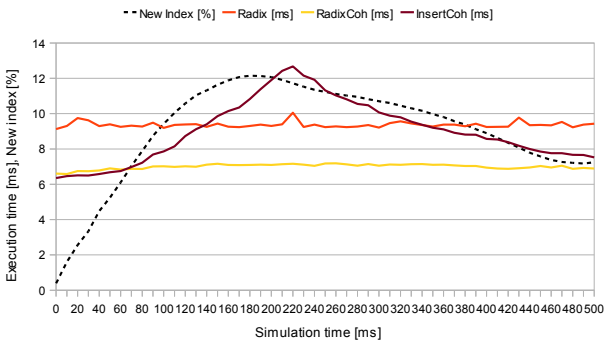

Figure 9: *Comparison of execution time (y-axis in ms) of non-coherent and coherent version of particle sorting. Unlike using insert sort, the coherent version of radix sort did not show significant slow down due to the increasing index change ratio, providing stable 1.3x to 1.5x speed up with respect to the full sorting.*

group slows down with the increase of index change ratio significantly. Coherent version of radix sort gave us the best results with only a subtle reaction to the index change achieving a 1.3x to 1.5x speedup.

5.2 Splash Generation

We have successfully demonstrated a large scale phenomena as wave breaking of ocean using SPH with our splashing technique. Only 20k fluid particles and about 15k splash particles were necessary to get plausible results, see Figure 8. This is an order of magnitude less than using only SPH with comparable results. Since splash particles are uncoupled (searching for neighbors in not necessary), they can be simulated in parallel gaining additional speedup.

In real units the whole scene is only about $0.6m$ wide and $0.7m$ long. Simulation duration until the first wave crash is $0.7s$. Maximal emit rate of every fluid particle was set to 700 splash particles per second, each having life time about $0.05s$. The probability of emission was calculated with Equation 3, where splashing density ratio α_d was set to 0.95 (95% of rest density), splashing density factor β_d to 2, minimal splashing velocity α_v to $0.7m/s$ and splashing velocity factor α_v to 2.

We rendered splash particles as additive semitransparent spherical point sprites. Contribution C_i of i-th particle with position p_i is accumulated along the ray based on the ray-particle distance d_i and particle life time L_i as

$$C_i = \left(1 - \frac{d_i}{d_0}\right)\left(1 - \left(2\frac{L_i}{L_0} - 1\right)^2\right), \tag{14}$$

where L_0 is the initial life time of particle and d_0 is the radius of spray particle. We set d_0 to $(0.001m)$ 10% of the fluid particle radius.

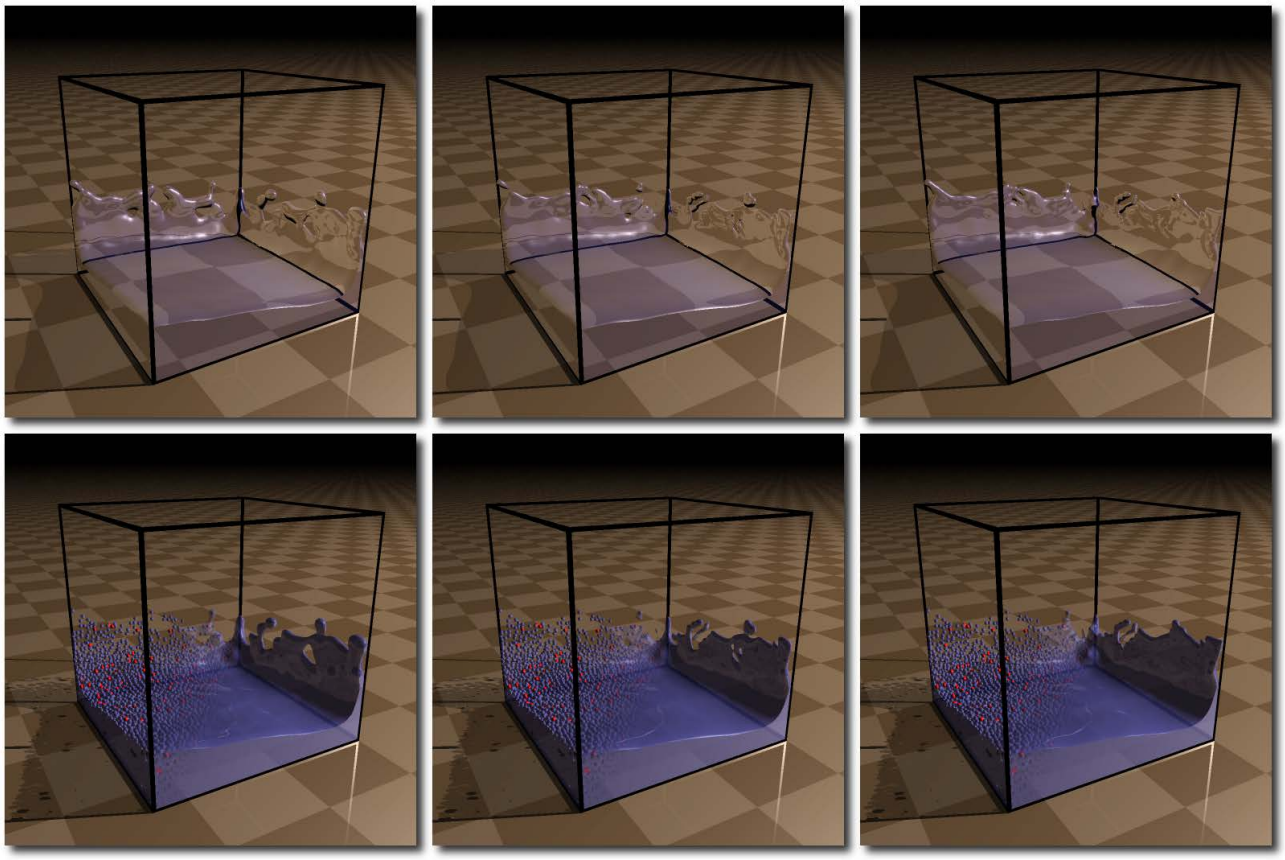

Figure 10: *Comparison of surface generated by* Left: *[Müller et al. 2003],* Middle: *[Zhu and Bridson 2005; Solenthaler et al. 2007] and* Right: *our density normalized method. Red particles in the bottom row have new indices used in the coherent neighbor search. Only 14k particles were used.*

5.3 Surface Reconstruction

Surface of small or moderate scenes usually looks too bloby, lacking thin regions of splashing fluid. We have thus setup an aquarium scene (0.4m x 0.4m x 0.4m) with only 14k particles for dam break test. As shown in Figure 10 and 6 the [Müller et al. 2003] method a nice smooth surface, but splashes are too bloby. Approach of [Zhu and Bridson 2005; Solenthaler et al. 2007] is less bloby, but shows several holes (no fluid) in thin regions. Due to density normalization our approach fills these holes making thin regions more compact.

The implicit surface is reconstructed with the iso level set to 0.8 (80% of iso particle radius). For the modulation function we set w_{min} to 0 and w_{max} to 1.05. To get smooth and flat surface, we use the density kernel function with support radius 2 times larger as used in SPH computation. Instead of using the physical density, we had to compute the new *iso* density with the larger kernel for each particle. This double interaction radius also has a significant impact on the performance of implicit surface raytracing since during iso funcion evaluation we had to visit 125 neighbor cells (including center cell) instead of only 27 as in SPH.

6 Conclusion and Future Work

We have proposed novel methods to speed up simulation and improve visual quality of particle based fluid animation. New approach for faster sorting of particle indices exploiting the coherency of physical motion, has been developed. In every frame only a fraction of particles change their spatial index, the rest stays sorted from the previous frame. We sort the small set of unordered particles using a regular radix sort and merge both sorted sets to construct a new ordered particle list.

Small scale details has been added to SPH simulation by emitting spray and foam particles during splashes. Each fluid particle serves as a potential source of spray particles. We propose simple conditions based on the concept of splashing probability to control particle emission. Visual enhancements are clearly beneficial mainly when prototyping large scale simulations with moderate number of SPH particles.

Commonly adopted surface generation method [Zhu and Bridson 2005] has been improved for uneven particle distributions. We normalize the center of mass by weighting particles with their estimated iso-density. Moreover, a new approach has been proposed to prevent artifact near isolated particles. We modulate surface distance based on the decay of normalized iso-density on center of mass. In contrast to [Solenthaler et al. 2007] we do not need to perform eigenvalue analysis giving us a simple analytic expression of surface normals.

Coherent radix sort is a natural candidate for parallelization. As a future work we will investigate benefits of a GPU based implementation. Currently our bubble and foam particles are governed by simple buoyant force without reaction to bulk fluid flow. We plan to extend our model to handle such scenarios.

7 Acknowledgements

This research was partially supported by a VEGA 1/0662/09 2009-2011 project a Scientific grant from Ministry of Education of Slovak Republic and Slovak Academy of Science.

8 Appendix

Assuming $\mathbf{r}_j = \mathbf{p} - \mathbf{p}_j \in \mathbb{R}^{3\times 1}$ and $\nabla\|\mathbf{r}_j\| = \frac{\mathbf{r}_j^{\mathbf{T}}}{\|\mathbf{r}_j\|} \in \mathbb{R}^{1\times 3}$ surface normal (iso-function gradient) can be expressed as

$$\begin{aligned}
\nabla\phi(\mathbf{p}) &= \tfrac{(\mathbf{p}-\mathbf{C}(\mathbf{p}))^{\mathbf{T}}}{\|\mathbf{p}-\mathbf{C}(\mathbf{p})\|}(\mathbf{1}-\nabla\mathbf{C}(\mathbf{p})) - R\nabla f(\mathbf{p}) && \in \mathbb{R}^{1\times 3}\\
\nabla f(\mathbf{p}) &= g'(w(\mathbf{C}(\mathbf{p})))\nabla w(\mathbf{C}(\mathbf{p}))\nabla\mathbf{C}(\mathbf{p}) && \in \mathbb{R}^{1\times 3}\\
\nabla w(\mathbf{c}) &= \textstyle\sum_j \frac{1}{w_j\|\mathbf{c}-\mathbf{p}_j\|} W'(\|\mathbf{c}-\mathbf{p}_j\|,h)(\mathbf{c}-\mathbf{p}_j)^{\mathbf{T}} && \in \mathbb{R}^{1\times 3}\\
g'(w) &= -4\left(1-\tfrac{(w-w_{\max})^2}{(w_{\max}-w_{\min})^2}\right)\tfrac{(w-w_{\max})}{(w_{\max}-w_{\min})^2} && \in \mathbb{R}\\
\nabla\mathbf{C}(\mathbf{p}) &= \tfrac{\nabla\mathbf{C}_1(\mathbf{p})C_2(\mathbf{p})-\mathbf{C}_1(\mathbf{p})\nabla C_2(\mathbf{p})}{C_2(\mathbf{p})C_2(\mathbf{p})} && \in \mathbb{R}^{3\times 3}\\
\nabla\mathbf{C}_1(\mathbf{p}) &= \textstyle\sum_j \frac{1}{\rho_j\|\mathbf{r}_j\|} W'(\|\mathbf{r}_j\|,h)\mathbf{p}_j\mathbf{r}_j^{\mathbf{T}} && \in \mathbb{R}^{3\times 3}\\
\nabla C_2(\mathbf{p}) &= \textstyle\sum_j \frac{1}{\rho_j\|\mathbf{r}_j\|} W'(\|\mathbf{r}_j\|,h)\mathbf{r}_j^{\mathbf{T}} && \in \mathbb{R}^{1\times 3}
\end{aligned} \tag{15}$$

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. In *ACM Transactions on Graphics (SIGGRAPH '07 papers)*, ACM Press, New York, NY, USA, vol. 26, 48–48*. 2, 4

CHENTANEZ, N., AND MÜLLER, M. 2010. Real-time simulation of large bodies of water with small scale details. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '10, 197–206. 2

CLAVET, S., BEAUDOIN, P., AND POULIN, P. 2005. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation 2005*, 219–228. 2

CLEARY, P. W., PYO, S. H., PRAKASH, M., AND KOO, B. K. 2007. Bubbling and frothing liquids. *ACM Trans. Graph. 26*, 3, 97. 2

FRAEDRICH, R., AUER, S., AND WESTERMANN, R. 2010. Efficient high-quality volume rendering of sph data. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2010) 16*, 6 (November-December), to appear. 2

GOSWAMI, P., SCHLEGEL, P., SOLENTHALER, B., AND PAJAROLA, R. 2010. Interactive SPH simulation and rendering on the GPU. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, M. Otaduy and Z. Popovic, Eds., 1–10. 2, 3

IHMSEN, M., AKINCI, N., BECKER, M., AND TESCHNER, M. 2011. A parallel sph implementation on multi-core cpus. *Computer Graphics Forum 30* (March), 99–112. 2, 3

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA, 457–462. 2

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154–159. 2, 4, 5, 7, 8

MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. 2005. Particle-based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 237–244. 2

O'BRIEN, J. F., AND HODGINS, J. K. 1995. Dynamic simulation of splashing fluids. In *CA '95: Proceedings of the Computer Animation*, IEEE Computer Society, Washington, DC, USA, 198. 2

ONDERIK, J., AND ĎURIKOVIČ, R. 2008. Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics, Statistics and Informatics 4*. 3

ROSENBERG, I. D., AND BIRDWELL, K. 2008. Real-time particle isosurface extraction. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '08, 35–43. 2

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible sph. *ACM Trans. Graph. 28* (July), 40:1–40:6. 2

SOLENTHALER, B., SCHLÄFLI, J., AND PAJAROLA, R. 2007. A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds 18*, 69–82. 2, 4, 5, 7, 8

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, 391–400. 2

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANTES, D., AND GROSS, M. H. 2003. Optimized spatial hashing for collision detection of deformable objects. In *VMV*, 47–54. 2, 3

VAN DER LAAN, W. J., GREEN, S., AND SAINZ, M. 2009. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '09, 91–98. 2, 4

VERLET, L. 1967. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev. 159*, 1 (Jul), 98. 2

YU, J., AND TURK, G. 2010. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '10, 217–225. 2, 4

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. ACM Press, New York, NY, USA, vol. 24, 965–972. 2, 4, 5, 7, 8