Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2011)
J. Bender, K. Erleben, and E. Galin (Editors)

SPH Based Shallow Water Simulation

Barbara Solenthaler[1] Peter Bucher[1] Nuttapong Chentanez[2] Matthias Müller[2] Markus Gross[1]

[1]ETH Zurich [2]NVIDIA PhysX Research

Abstract

We present an efficient method that uses particles to solve the 2D shallow water equations. These equations describe the dynamics of a body of water represented by a height field. Instead of storing the surface heights using uniform grid cells, we discretize the fluid with 2D SPH particles and compute the height according to the density at each particle location. The particle discretization offers the benefits that it simplifies the use of sparsely filled domains and arbitrary boundary geometry. Our solver can handle terrain slopes and supports two-way coupling of the particle-based height field with rigid objects. An improved surface definition is presented that reduces visible bumps related to the underlying particle representation. It furthermore smoothes areas with separating particles to achieve better rendering results. Both the physics and the rendering are implemented on modern GPUs resulting in interactive performances in all our presented examples.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation and Virtual Reality

1. Introduction

Physically-based simulations have become an important element of real-time applications like computer games. This is because they increase the plausibility of the simulated materials and give the user the ability to realistically interact with the environment. Until now, physical simulations have largely been limited to solid bodies; other materials like fluids have been used with reservation since they pose high computational and methodological challenges to a real-time application. The reason for this is that high-resolution simulations require millions of grid cells or particles with a full 3D Eulerian or Lagrangian fluid simulation, respectively. Real-time applications typically reduce the simulated domain from a 3D volume to a 2D surface to reduce the computational costs. A popular approach are height field fluids where the domain is discretized by 2D grid cells that store the height of the surface at the respective location. A simple technique to simulate surface phenomena like puddles and waves is to solve the wave equation. This approach, however, does not handle bulk flow in the horizontal direction, and therefore effects like swirling rivers and floating objects cannot be simulated correctly. The shallow water equations (SWE) overcome this problem by incorporating a 2D velocity field normal to the water columns.

The Eulerian domain decomposition has been widely used in height field methods as well as in full 3D simulations. The grid discretization allows for efficient simulations, but handling irregular domain boundaries that are not aligned with the grid accurately and robustly is difficult. Moreover, the grid structure is not well suited to simulate sparse domains and water flowing into empty regions. Such effects, however, increase the realism of a virtual world and offer the ability to flexibly interact with the surrounding. In contrast to the domain decomposition of Eulerian models, the Lagrangian methods discretize the fluid volume by particles that can move to arbitrary locations. Recently, particles have been successfully used in SWE simulations in [LH10]. Lee and Han applied the SPH model to solve the SWE equations, including a one-way coupling of fluids with simple geometries represented by virtual particles.

In this paper, we extend the particle-based SWE model of Lee and Han [LH10] in order to fully benefit from the advantages of the Lagrangian nature. The main contributions of this paper are:

- Extensions of the basic particle-based SWE method that allow the simulation of arbitrary domain boundaries and terrain slopes.
- A two-way coupling of the particle-based fluid with rigid bodies, including buoyancy, drag and lift forces.
- A surface representation based on the underlying density

field that improves the rendering of low density particle regions.

We further present a detailed derivation of the particle-based SWE equations using 2D SPH particles.

2. Related Work

3D simulations of water are well studied in the literature and are used to model complex flow patterns like vortex structures and breaking waves. These simulations come at high costs because they typically require a high-resolution discretization, i.e., millions of cells or particles, to reproduce small-scale effects. There are many works on full 3D simulation of water, a detailed introduction can be found in [Bri08] for Eulerian simulations and [Mon05] for Lagrangian solvers.

A common practice to reduce the simulation costs of full 3D simulations while still being able to resolve highly detailed surface features is to reduce the simulation domain from 3D to 2D. Procedural methods have been used in e.g. [FR86, TDG00, HNC02] to describe ocean waves. These methods do not solve physical models but rather use parametric functions and mathematical descriptions to compute the velocity field locally. Procedural models can handle large-scale animations efficiently, however, the simulation of vortices and object interaction is challenging. The first works in computer graphics that simulated the water surface by solving the wave equation on a 2D height field are [KM90, Tes99]. The model has been extended to simulate rigid body interaction [OH95], bubbles and droplets [MY97], waterfalls [HW04], and terrain erosion [vBBK08]. Recently, [YHK07] introduced a method where 2D particles carrying the velocity and wavefront information are dynamically generated and removed at the surface. The main limitation of the wave equation model is that effects from vortices are not included, preventing the simulation of whirlpool effects and correct advection of floating objects.

In contrast to the wave equation model, the Shallow Water Equations (SWE) [LVDP02] include a 2D velocity field in addition to the surface height. The equations are derived from the Navier-Stokes equations and describe conservation of mass and momentum. The SWE method has been extended to resolve breaking waves in [TMFSG07] by generating and evolving triangle mesh patches, as well as in [CM10] where particles are dynamically added on top of steep and fast moving wavefronts. The latter work has additionally included spray and foam particles to simulate waterfalls and object interaction more realistically. The difficulty of modeling complex boundaries with regular grid approaches has been addressed in [HHL*05] where a finite volume method is used to solve the SWE.

A particle representation, on the other hand, overcomes these limitations since particles can move to arbitrary locations and can separate from the main body of fluid. In computational fluid dynamics, the particle model Smoothed Particle Hydrodynamics (SPH) has been used to solve the SWE equations to analyze dam breaks and flood hazards. In [AS05], SPH for shallow water simulation has been used on a flat terrain, and stability has been improved by introducing an additional diffusive term. [RPB05] augmented SPH to also store the water depth to reduce the number of particles required in deep water, at the expense of non-constant kernel radii. Anisotropic SPH kernels with variable smoothing length are used in [dLTA08] to obtain more numerical accuracy when water spreads out. The particle-based shallow water method has been introduced to computer graphics in [LH10]. Lee and Han have included a one-way coupling of particles with solid objects, where the objects are represented by ghost (virtual) particles. Our work is based on the basic concept presented in Lee and Han, but we additionally include arbitrary terrain boundaries and two-way fluid-solid interaction to make better use of the benefits of the Lagrangian nature.

3. Basic Models

Our method solves the shallow water equations on 2D SPH particles. In the following, we give a brief overview of the basic SWE and SPH equations. For detailed derivations and descriptions of both models we refer the reader to [Bri05] for SWE and [DC96, Mon05] for SPH.

3.1. Shallow Water Equations

The shallow water equations are derived from the Navier-Stokes equations to describe the evolution of the liquid surface given by a 2D height field. They represent conservation of volume and momentum and can be written as

$$\frac{Dh}{Dt} = -h\,\nabla\cdot\mathbf{u} \tag{1}$$

$$\frac{D\mathbf{u}}{Dt} = -g\nabla(h+H)+\mathbf{a}_{\text{ext}}, \tag{2}$$

where h is the height of the water above ground, $\mathbf{u}$ is the 2D horizontal water velocity and g is gravity. In addition to [LH10], we include the height of the terrain H and external accelerations $\mathbf{a}_{\text{ext}}$. For particles moving with the fluid, the material derivative D/Dt turns into the simple derivative d/dt.

3.2. 2D SPH

According to SPH, a scalar quantity A of a particle i is interpolated by a weighted sum of contributions from all neighboring particles j:

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{r}_i - \mathbf{r}_j, l), \tag{3}$$

where $\mathbf{r}$ is the particle position, ρ_j the density of j and $W(\mathbf{r},l)$ the smoothing kernel with finite support l. Differ-

ential operators act on the kernels only and are

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W(\mathbf{r}-\mathbf{r}_j, l) \tag{4}$$

$$\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W(\mathbf{r}-\mathbf{r}_j, l) \tag{5}$$

$$\nabla \cdot \mathbf{A}_i = \sum_j \frac{m_j}{\rho_j} \mathbf{A}_j \cdot \nabla W(\mathbf{r}-\mathbf{r}_j, l). \tag{6}$$

We use the same kernels as presented for the 3D case in [MCG03] but adapted to 2D:

$$W_{\text{poly6}}(\mathbf{r}, l) = \frac{4}{\pi l^8} \begin{cases} (l^2 - r^2)^3 & 0 \le r \le l \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$W_{\text{spiky}}(\mathbf{r}, l) = \frac{10}{\pi l^5} \begin{cases} (l-r)^3 & 0 \le r \le l \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

$$W_{\text{viscosity}}(\mathbf{r}, l) = \frac{10}{9\pi l^5} \tag{9}$$
$$\begin{cases} -4r^3 + 9r^2 l - 5l^3 + 6l^3(\ln l - \ln r) & 0 \le r \le l \\ 0 & \text{otherwise.} \end{cases}$$

The Laplacian of the first two kernels is not positive everywhere which can cause stability issues, thus we use the third kernel for computing the viscosity term. We designed the viscosity kernel such that

$$\nabla^2 W(\mathbf{r}, l) = \frac{40}{\pi l^5}(l-r) \tag{10}$$

$$W(|\mathbf{r}| = l, l) = 0 \tag{11}$$

$$\nabla W(|\mathbf{r}| = l, l) = \mathbf{0}. \tag{12}$$

We used Equation (9) to determine the correct scaling factor in Equation (10). During the simulation, only the Laplacian of Equation (9) is evaluated. The derivation of the kernel properties is given in Appendix A.

4. Particle-Based SWE Model

Our SWE solver uses 2D particles to discretize the surface of a fluid, as shown in Figure 1. In Section 4.1, we first derive the Lagrangian shallow water equations in detail and show how they can be solved with SPH. We then explain in Section 4.2 how arbitrary terrains and water heights can be integrated, and show our two-way coupling of the particle-based fluid with rigid objects in Section 4.3. Section 4.4 then discusses how the surface definition and rendering can be improved by addressing the problems coming with the particle representation.

4.1. Particle-Based SWE Equations

Our aim is to solve Equations (1) and (2) using 2D SPH particles attached to the ground. In this case, the horizontal 2D velocity field is defined by the velocities of the particles as

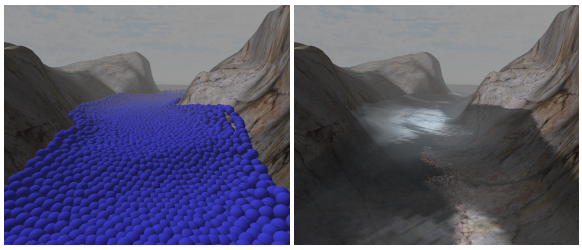

Figure 1: *The particle representation facilitates the interaction with complex and unbounded geometries. The fluid surface is defined by the particles (left) and is rendered in real-time (right).*

in regular 2D SPH simulations. A straightforward approach would be to store the scalar water height h as an additional attribute on the particles. In this case, one would evaluate the divergence and gradient of the water height appearing in the SWE with the corresponding SPH approximations. The main problem with this simple approach is the right hand side of Equation (2). Imagine two or more particles which store the same height approaching each other. There is no repulsive force between them because the gradient of the height is zero so they can get arbitrarily close to each other. To get good sampling, additional forces would have to be added or the particles would have to be repositioned directly.

We solve that problem in a simple way as proposed in [RPB05]. The idea is to define the height implicitly via the 2D density of the fluid:

$$h = \frac{\rho^{2D}}{\rho^{3D}}. \tag{13}$$

Assuming the 3D density of the water to be constant, the two quantities are proportional via the constant of proportionality $\frac{1}{\rho^{3D}}$. In the remainder of the paper we will use $\rho = \rho^{2D}$ and $\rho_0 = \rho^{3D}$. With constant particle masses and normalized kernels, volume conservation is free in this formulation because the integral of the density (height) field over the 2D domain is constant:

$$\begin{aligned} V &= \int_\Omega h \, d\mathbf{r} = \int_\Omega \frac{1}{\rho_0} \rho \, d\mathbf{r} \\ &= \frac{1}{\rho_0} \int_\Omega \sum_j \left(m_j W(\mathbf{r}-\mathbf{r}_j) \right) d\mathbf{r} \\ &= \frac{1}{\rho_0} \sum_j m_j \int_\Omega W(\mathbf{r}-\mathbf{r}_j) \, d\mathbf{r} \\ &= \frac{1}{\rho_0} \sum_j m_j = const. \end{aligned}$$

This means that the weak form of Equation (1) holds automatically and we do not lose water globally. To guarantee volume conservation locally, the strong form must also hold.

This is only true in the limit:

$$\begin{aligned}\frac{\partial}{\partial t}\rho_i &= \frac{\partial}{\partial t}\left(\sum_j m_j W(\mathbf{r}_i(t)-\mathbf{r}_j(t))\right)\\ &= \sum_j m_j \frac{\partial}{\partial t} W(\mathbf{r}_i(t)-\mathbf{r}_j(t))\\ &= \sum_j m_j \nabla W(\mathbf{r}_i-\mathbf{r}_j)\cdot(\mathbf{u}_i-\mathbf{u}_j)\\ &= \mathbf{u}_i \cdot \sum_j m_j \frac{\rho_j}{\rho_j}\nabla W(\mathbf{r}_i-\mathbf{r}_j)\\ &\quad -\sum_j m_j \frac{\rho_j \mathbf{u}_j}{\rho_j}\cdot\nabla W(\mathbf{r}_i-\mathbf{r}_j).\end{aligned}$$

Using Equations (4) and (6) and with the kernel size approaching zero we get:

$$\begin{aligned}\frac{\partial}{\partial t}\rho_i &= \mathbf{u}\cdot\nabla\rho - \nabla\cdot(\rho\mathbf{u})\\ &= \mathbf{u}\cdot\nabla\rho - (\nabla\rho\cdot\mathbf{u}+\rho\nabla\cdot\mathbf{u})\\ &= \rho\nabla\cdot\mathbf{u}\end{aligned}$$

and with $h=\frac{1}{\rho_0}\rho$

$$\frac{\partial}{\partial t}h_i = h\nabla\cdot\mathbf{u}$$

which is Equation (1). Since ρ_i is defined on the particle, the left hand side above corresponds to the material derivative. The momentum conservation Equation (2) becomes

$$\frac{D\mathbf{u}}{Dt} = -g\nabla(h+H)+\mathbf{a}_{\text{ext}} \tag{14}$$

$$= -\frac{g}{\rho_0}\nabla\rho - g\nabla H + \mathbf{a}_{\text{ext}}. \tag{15}$$

This equation is equal to the momentum conservation equation used in regular SPH with stiffness constant $k=\frac{g}{\rho_0}$ and with the additional external force $-g\nabla H$ which acts along the gradient of the terrain. In other words what we have to do is to *perform a regular 2D SPH fluid simulation and interpret the density as height*. The same technique has also been applied in [LH10].

4.2. Arbitrary Domain Boundaries

We define the terrain by a height map, i.e. a scalar field where each scalar denotes the height at discrete positions. The external terrain force

$$\mathbf{f}_i^{\text{terrain}} = -g\nabla H_i \tag{16}$$

acts along the gradient of the terrain. We compute ∇H at each discrete position of our height map with central differences. This has to be computed only once when initially loading the data. The gradient values are then bilinearly interpolated at the particle positions.

Very steep and vertical domain boundaries can introduce very large terrain forces. Thus, we use simple geometries to represent the boundary in these cases, and replace the terrain force by a boundary force repulsing a particle whose distance l_i to the boundary is less than b. We defined the force as

$$\mathbf{f}_i^{\text{boundary}} = \alpha_i(\mathbf{f}_i\cdot\mathbf{n})\mathbf{n}, \tag{17}$$

where $\mathbf{f}_i$ is the sum of all other forces acting on i and $\mathbf{n}$ is the normal of the closest point on the boundary. α_i is the repulsion strength defined as

$$\alpha_i = c(1-\frac{l_i}{b}), \tag{18}$$

which is higher the smaller the distance is to the boundary. c is a parameter to control the force intensity; in all our examples we used $c=0.15$.

4.3. Rigid Body Interaction

Our method considers two-way coupled interaction of particle-based height fields and rigid objects.

4.3.1. Fluid to Solids

The height field exerts three forces on a solid, namely buoyancy, drag and lift forces. The buoyancy force $\mathbf{f}^{\text{buoyancy}} = -g\rho V$ is proportional to the displaced mass of the fluid ρV and is pointing upward. An object either floats on the surface or sinks depending on its relative weight as shown in Figure 2. In order to determine the mass of the displaced fluid, we have to compute the fraction of the water column above the particle that is replaced by the solid. For each particle, we cast a ray vertically upwards and compute its bottom- and topmost intersection with the object's surface $d_{i,min}$ and $d_{i,max}$. The immersion depth stored at particle i is defined as

$$d_i = min(d_{i,max},h) - d_{i,min}. \tag{19}$$

The minimum is used to distinguish whether the object is floating or is fully immersed. The displaced mass is then computed by $\hat{m}_i = m_i d_i/h_i$, leading to the buoyancy force per particle

$$\mathbf{f}_i^{\text{buoyancy}} = -g\hat{m}_i. \tag{20}$$

Drag and lift forces are computed analogously to [YHK07] with

$$\mathbf{f}_i^{\text{drag}} = -\frac{1}{2}C_D A_i^{\text{eff}}|\mathbf{u}_{rel}|\mathbf{u}_{rel}, \tag{21}$$

$$\mathbf{f}_i^{\text{lift}} = -\frac{1}{2}C_L A_i^{\text{eff}}|\mathbf{u}_{rel}|(\mathbf{u}_{rel}\times\frac{\mathbf{n}\times\mathbf{u}_{rel}}{|\mathbf{n}\times\mathbf{u}_{rel}|}) \tag{22}$$

$$\tag{23}$$

where C_D and C_L are drag and lift coefficients, $\mathbf{n}$ the surface normal, and $\mathbf{u}_{rel}=\mathbf{u}_o-\mathbf{u}_i$ is the relative velocity between the object and the fluid. A_i^{eff} is the effective area and can be computed with

$$A_i^{\text{eff}} = \begin{cases}(\frac{\mathbf{n}\cdot\mathbf{u}_{rel}}{\mathbf{u}_{rel}}\alpha^s+(1-\alpha^s))A_i & \mathbf{n}\cdot\mathbf{u}_{rel}>0\\ 0 & \text{otherwise.}\end{cases} \tag{24}$$

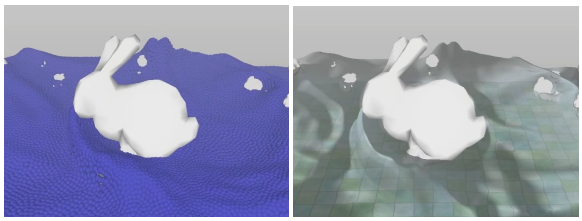

Figure 2: *Our two-way fluid-solid interaction produces realistic waves and ripples at the surface. The left image shows the particles, and the right image the resulting surface.*

where α^s depends on the geometry of the object [YHK07]. The area A_i represents the water column above a particle and is defined by

$$A_i = \frac{m_i}{\rho_0 h_i} = \frac{m_i}{\rho_i}. \tag{25}$$

4.3.2. Solids to Fluid

In contrast to grid-based SWE methods, SPH-based SWE does not allow to modify the height of the fluid due to solids in a simple way. Modifying the height means that particles have to be rearranged such that the resulting SPH densities change accordingly. Since the SPH density computed by

$$\rho_i = m_i \sum_j W(\mathbf{r}, l) \tag{26}$$

directly depends on the number of neighbors and their positions, i.e. kernel values, small position changes can already have a large effect on the computed quantity. Thus, we cannot modify particle locations easily and without introducing discontinuities. In the worst case, this can introduce large pressure forces and instabilities.

We solve this problem by defining a collision force $\mathbf{f}^{\text{collision}}$ based on an elastic collision between the particles and the object. In a fully elastic collision, the new velocity of a particle i is given by

$$\mathbf{u}_i' = \frac{m_i\mathbf{u}_i + m_o\mathbf{u}_o - m_o(\mathbf{u}_i - \mathbf{u}_o)}{m_i + m_o}, \tag{27}$$

where m_o is the mass of the immersed part of the object, and $\mathbf{u}_o$ is the velocity of the object at the collision point. The collision force applied on a particle is therefore given by

$$\mathbf{f}_i^{\text{collision}} = \beta m_i \frac{\mathbf{u}_i' - \mathbf{u}_i}{\Delta t}. \tag{28}$$

$0 \leq \beta \leq 1$ defines the influence of the object onto the particle. Note that a large value for β could push all particles below the object away very quickly, preventing that any buoyancy force acts on the object. Our experiments have shown that a value of $\beta = 0.15$ produces visually appealing water waves. The effect of the collision force is illustrated in Figure 3.

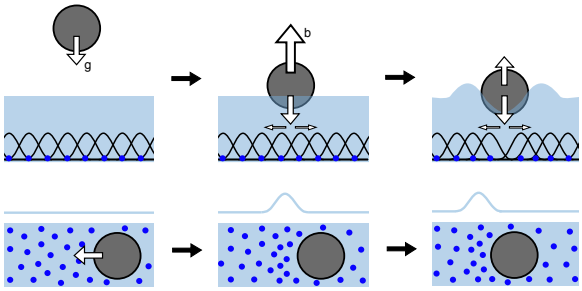

Figure 3: *Our collision force moves the particles, producing higher density values and thus impact waves. Top: Side view of an object impact. Bottom: Top view of a horizontally moving object.*

4.4. Surface Definition

We create a surface map s by combining a modified density field ρ^* from the particles with the underlying terrain:

$$s(\mathbf{r}) = \frac{\rho^*(\mathbf{r})}{\rho_0} + H(\mathbf{r}). \tag{29}$$

This surface map is then used as a vertex displacement map to determine the vertex positions of the final surface mesh.

The simplest approach is to use the SPH density given by Equation (26) for the rendering. This field, however, is unable to reconstruct a flat surface since bumps related to the particle distribution are well visible. Normalization to alleviate these problems cannot be applied because it would result in a constant function $\rho(\mathbf{r}) = m$. We therefore apply the kernel smoothing concept to the discrete density (see Figure 4 (a)), which is

$$\rho(\mathbf{r}) = \sum_j \rho_j W(\mathbf{r} - \mathbf{r}_j, l), \tag{30}$$

and normalize this field resulting in

$$\hat{\rho}(\mathbf{r}) = \frac{\rho(\mathbf{r})}{\sum_j W(\mathbf{r} - \mathbf{r}_j, l)}. \tag{31}$$

The normalized field reduces surface bumps, but on the other hand, artifacts at the surface borders are visible (see Figure 4 (b)). In order to get the best of both formulations, flat surfaces and smoothly declining borders, we smoothly interpolate between the unnormalized and the normalized density fields given by Equations (30) and (31) (see Figure 4 (c)):

$$\rho^*(\mathbf{r}) = \alpha\hat{\rho}(\mathbf{r}) + (1-\alpha)\rho(\mathbf{r}), \tag{32}$$

where α is determined by $\alpha = \frac{\rho(\mathbf{r}) - \rho_{min}(\mathbf{r})}{\rho_{max}(\mathbf{r}) - \rho_{min}(\mathbf{r})} \in [0..1]$, and $\rho_{min}(\mathbf{r})$ and $\rho_{max}(\mathbf{r})$ are the minimal and maximal density value in the neighborhood of $\mathbf{r}$.

With the particle representation, water flowing into empty regions can be simulated in a straightforward way. However, in such cases it might happen that individual particles separate and appear as large blobs in the rendered result as shown

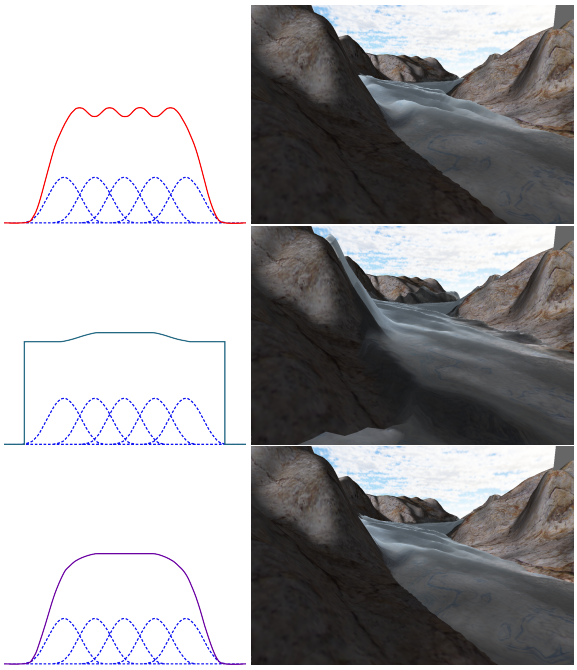

Figure 4: *We smoothly interpolate between the unnormalized (top) and normalized density field (middle) to get the best of both formulations: Smoothly declining borders and flat surfaces (bottom).*

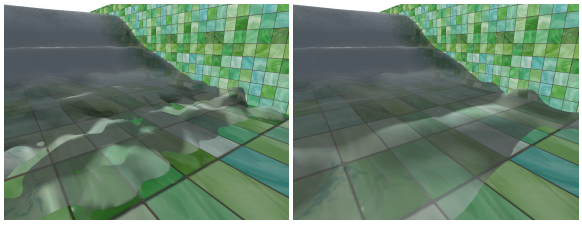

Figure 5: *Particles that separate from the main flow are rendered as large blobs. To improve the rendering in these situations, we smooth the density field over a larger area.*

in the left image in Figure 5. We solve this problem by detecting such low density areas in the fluid and smoothing the density field in these situations. This is achieved by increasing the kernel size in Equations (30) and (31) with decreasing density. To this end, we have defined two density thresholds, $\rho_1 = 30$ and $\rho_2 = 150$, in between the kernel radius l is increased linearly up to a maximum size which we have set to $3l$ in all our examples. Note that variable kernel sizes are only used for rendering. The effect is shown in the right image of Figure 5.

5. Results and Discussion

We implement our method in CUDA and run the simulations on a 2.66 GHz Core i7 and NVIDIA GTX 460. We used NVIDIA's PhysX SDK for the rigid body dynamics. We have used two different setups to measure the performance, the basin and the whirlpool scenarios shown in Figure 6. Table 1 lists the performance values of both examples with increasing particle numbers as well as increasing number of rigid objects. We used a time step of $0.002s$ in these examples. The timings are given in fps, once for the physics computation only, and once including the rendering after each physics step. The performance data show that we can simulate and render 128k particles at an interactive rate of 20fps. With the same number of particles but adding 196 objects, the frame rate decreases to 7fps. We note that the particle-based SWE formulation does not reach the same performance as the grid-based SWE models. However, the particle-based approach offers many benefits over grid-based solvers like the simple handling of complex and sparsely filled domains. The choice of a particular solver should therefore depend on the designated simulation scene and application.

The benefits of the particle representation is demonstrated in the following examples. In all of them, we achieved interactive performances. In Figure 7, the fluid is interacting with non-axis aligned boundaries which a user dynamically removes causing the water to flow out. The interaction with a complex terrain boundary is shown in Figure 1 and 8. We used height maps of sizes 512x512 and 256x256, respectively. In the latter example, the solid objects get advected by the river which is discretized by 100k particles in total. Due to the chosen camera positions, however, only about 15k particles are visible in the particular screenshots. The rigid objects either float or sink depending on their weight, in both cases producing realistic waves and ripples. If not stated otherwise, we used our real-time OpenGL renderer in all our examples. The result of our real-time renderer is visually compared to the offline Povray raytracer in Figure 8.

One drawback of our method is that several parameters have to be set, each of them influencing the visual result of the simulation. While this offers a high degree of artistic freedom to an animator, it can also be a tedious process to achieve the desired result. We have determined the parameters once as stated in the corresponding sections, and used these values to produce all our examples.

While our surface definition accounts for a smooth rendering of low density regions, it would be desirable to control the actual number of particles in a certain region dynamically. This is mainly important in simulations with unbounded domains to guarantee a minimal particle sampling rate. This could be achieved by locally adjusting the resolution with particle splitting and merging processes, analogously to the idea presented in [APKG07] for 3D SPH simulations. The problem is, however, to change the resolution

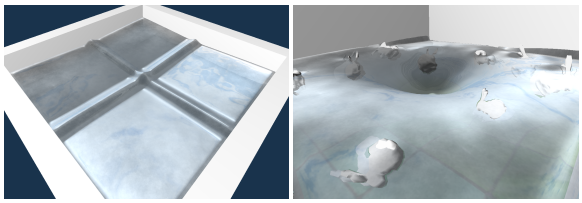

Figure 6: *The basin (left) and whirlpool (right) examples are used to evaluate the performance of our method presented in Table 1.*

Figure 6	#particles	#objects	physics [fps]	physics + rendering [fps]
Basin (left)	16k	-	629	120
	32k	-	317	86
	64k	-	131	55
	128k	-	41	20
Whirlpool (right)	16k	16	254	73
	32k	36	144	39
	64k	81	62	18
	128k	196	21	7

Table 1: *Performance values of two different examples, measured for various particle and object numbers.*

locally in a stable way, avoiding artifacts at the surface as well as abrupt momentum changes.

6. Conclusions and Future Work

We have presented a new approach for simulating height field fluids based on 2D SPH particles. Discretizing the height field with particles offers many benefits over traditional grid-based approaches, namely the simple handling of complex and sparsely filled domains. This allows the interaction of a user with the flow and the environment in a more flexible way. Our surface representation is modeled so that we avoid visual artifacts related to the underlying particle distribution.

There are several ways to further improve the visual appearance of the fluid. First, splashes, foam and waterfall effects could be modeled, for example by including simple particle systems as in [CM10]. And second, our method could be extended to allow the dynamic insertion and deletion of particles to guarantee a minimal particle sampling rate. Dynamically changing the particle number in a stable way is, however, a difficult task and needs further studies in the future.

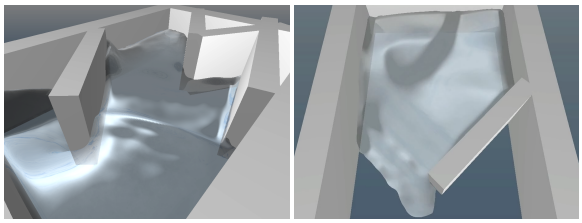

Figure 7: *Individual walls are dynamically removed by a user, causing the fluid to flow into empty domain regions.*

Figure 8: *A dam built of several logs collapses, resulting in the fluid and the debris to flow down the valley. An unbounded domain is used, and the particles can flow to arbitrary locations. The bottom row shows different rendering techniques; on the left the surface is rendered offline with Povray, and on the right we used our real-time OpenGL renderer.*

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. (Proceedings of SIGGRAPH) 26*, 3 (2007), 48–54. 6

[AS05] ATA R., SOULAIMANI A.: A stabilized SPH method for inviscid shallow water flows. *International Journal for Numerical Methods in Fluids 47* (2005), 139–159. 2

[Bri05] BRIDSON R.: Shallow water discretization, Lecture notes Animation Physics, 2005. University of British Columbia. 2

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A K Peters, 2008. 2

[CM10] CHENTANEZ N., MÜLLER M.: Real-time simulation of large bodies of water with small scale details. In *Proceedings of ACM SIGGRAPH / EUROGRAPHICS Symposium on Computer Animation* (2010). 2, 7

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Computer Animation and Simulation '96* (1996), pp. 61–76. 2

[dLTA08] DE LEFFE M., TOUZÉ D. L., ALESSANDRINI B.: SPH

modeling of shallow-water coastal flows. In *Proceeding of the 8th International Conference on Hydrodynamics* (2008). 2

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

[HHL*05] HAGEN T., HJELMERVIK J., LIE K.-A., NATVIG J., HENRIKSEN M. O.: Visual simulation of shallow-water waves. *Simulation Modelling Practice and Theory 13*, 8 (2005), 716 – 726. 2

[HNC02] HINSINGER D., NEYRET F., CANI M.-P.: Interactive animation of ocean waves. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), pp. 161–166. 2

[HW04] HOLMBERG N., WÜNSCHE B. C.: Efficient modeling and rendering of turbulent water over natural terrain. In *Proceedings of GRAPHITE* (2004), pp. 15–22. 2

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

[LH10] LEE H., HAN S.: Solving the shallow water equations using 2d sph particles for interactive applications. *The Visual Computer 26*, 6-8 (2010), 865–872. 1, 2, 4

[LVDP02] LAYTON A. T., VAN DE PANNE M.: A numerically efficient and stable algorithm for animating water waves. *The Visual Computer 18* (2002), 41–53. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. 3

[Mon05] MONAGHAN J. J.: Smoothed Particle Hydrodynamics. *Rep. Prog. Phys. 68* (2005), 1703–1759. 2

[MY97] MOULD D., YANG Y.-H.: Modeling water for computer graphics. *Computers & Graphics 21*, 6 (1997), 801 – 814. 2

[OH95] O'BRIEN J. F., HODGINS J. K.: Dynamic simulation of splashing fluids. In *Proceedings of Computer Animation* (1995), pp. 198–205. 2

[RPB05] RODRIGUEZ-PAZ M., BONET J.: A corrected smooth particle hydrodynamics formulation of the shallow-water equations. *Comput. Struct. 83* (2005), 1396–1410. 2, 3

[TDG00] THON S., DISCHLER J.-M., GHAZANFARPOUR D.: Ocean waves synthesis using a spectrum-based turbulence function. In *Proceedings of CGI* (2000), p. 65. 2

[Tes99] TESSENDORF J.: Simulating ocean water. *SIGGRAPH course notes* (1999), 8. 2

[TMFSG07] THUREY N., MULLER-FISCHER M., SCHIRM S., GROSS M.: Real-time breaking waves for shallow water simulations. In *Proceedings of Pacific Conference on Computer Graphics and Applications* (2007), pp. 39–46. 2

[vBBK08] ŠT'AVA O., BENEŠ B., BRISBIN M., KŘIVÁNEK J.: Interactive terrain modeling using hydraulic erosion. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 201–210. 2

[YHK07] YUKSEL C., HOUSE D. H., KEYSER J.: Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 26*, 3 (2007), 99. 2, 4, 5

Appendix A: Derivation of the Viscosity Kernel Properties

In the following, the scaling factor in Equation (10) is derived, satisfying the properties given in Equations (11) and (12) as well as the unity constraint.

In polar coordinates and for symmetric functions, the Laplacian is

$$\Delta f = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) \tag{33}$$

So

$$\begin{aligned}
\frac{\partial f}{\partial r} &= \frac{10}{9\pi l^5}\left(-12r^2+18lr-6l^3\frac{1}{r}\right)\\
r\frac{\partial f}{\partial r} &= \frac{10}{9\pi l^5}\left(-12r^3+18lr^2-6l^3\right)\\
\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) &= \frac{10}{9\pi l^5}\left(-36r^2+36lr\right)\\
\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) &= \frac{10}{9\pi l^5}\left(-36r+36l\right)\\
\Delta f &= \frac{40}{\pi l^5}(l-r)
\end{aligned}$$

and

$$\begin{aligned}
f(l) &= \frac{10}{9\pi l^5}\left(-4l^3+9l^3-5l^3+6l^3(\ln l-\ln l)\right)\\
&= 0
\end{aligned}$$

and

$$\begin{aligned}
\left.\frac{\partial f}{\partial r}\right|_l &= \left.\frac{10}{9\pi l^5}\left(-12r^2+18lr-6l^3\frac{1}{r}\right)\right|_l\\
&= \frac{10}{9\pi l^5}\left(-12l^2+18l^2-6l^2\right)\\
&= 0
\end{aligned}$$

and

$$\begin{aligned}
&\int_{\mathbf{r}} W_{\text{viscosity}}(\mathbf{r},l)d\mathbf{r}\\
&= \int_0^{2\pi}\int_0^{l}\frac{10}{9\pi l^5}\left(-4r^3+9r^2l-5l^3+6l^3(\ln l-\ln r)\right)r\,dr\,d\phi\\
&= \frac{20\pi}{9\pi l^5}\left[-\frac{4}{5}r^5+\frac{9}{4}lr^4-\frac{5}{2}l^3r^2+6l^3\left(\frac{1}{2}r^2\ln l-(\frac{1}{2}r^2\ln r-\frac{1}{4}r^2)\right)\right]_0^l\\
&= \frac{1}{9l^5}\left(-16l^5+45l^5-50l^5+6l^3(5l^2)\right)\\
&= 1
\end{aligned}$$