DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Visual Enhancement of Liquid Simulation using Secondary Particles

Georg Kohl

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Visual Enhancement of Liquid Simulation using Secondary Particles

Visuelle Verbesserung von Flüssigkeitssimulationen durch Sekundärpartikel

Author:	Georg Kohl
Supervisor:	Prof. Dr. Nils Thuerey
Advisor:	Dr. Kiwon Um
Submission Date:	15.09.2017

I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.09.2017 Georg Kohl

Acknowledgments

I want to thank Prof. Dr. Nils Thuerey for coming up with the interesting and challenging topic for this thesis and for improvement ideas during the implementation phase. In addition, thank you to Dr. Kiwon Um who always had time for my questions. Without his helpful advice and tips the quality of my implementation would have been much lower. Last but not least, I thank my brother for his constructive feedback on the simulation videos and for helping me with design choices like lighting and camera positions for rendering.

I express my gratitude to everyone who proof-read this thesis: my advisor Dr. Kiwon Um, my brother Paul and my friends Nepomuk, Quirin and Simon. Without your support many typing errors, wrong words and badly written sentences would have remained undetected.

Abstract

This thesis investigates methods to simulate the interaction between air and water. Ihmsen et al. [Ihm+12] proposed a secondary particle approach for Smoothed Particle Hydrodynamics (SPH). Based on their model we present a new adaptation for Fluid-Implicit-Particle (FLIP), which is a popular hybrid particle-grid method for fluid simulations in computer graphics that has been suffering from the difficulty in capturing small-scale effects coming from such interactions. With our new method it is possible to create spray, foam and air bubbles that significantly augment the visual realism.

It is realized as a post-processing step that does not influence the underlying fluid simulation. This means simulating the main water volume can be separated from detail effects and thus supports the animators natural workflow, in first developing a coarse concept and enhancing it later without the necessity of readjusting the basis. Furthermore, this thesis contains a detailed discussion of the many user-defined constants that allow very precise control over the simulation result and leave room for artistic freedom.

Finally, we compare both methods in regard to their visual impression and evaluate their performance. We demonstrate that our model can efficiently handle millions of particles and is even slightly faster than our implementation of the method from Ihmsen et al. [Ihm+12]. Our presented approach uses dissolving foam structures to enhances small surface waves especially well. Without the secondary refinement these are easily overlooked. The so-called surf zone where waves become unstable and start to break receive spray particles that can compensate the insufficient droplet creation from FLIP. In addition, air bubbles which are most beneficial for setups with low camera angles are realistically created by impacts and near obstacles.

Contents

1 Introduction

There are two fundamentally different approaches to simulate fluids, the Eulerian and Lagrangian approach. The former is a grid-based method where the fluid is described from stationary points in space. Quantities like density or pressure are saved and computed for each grid cell. The latter is a particle-based method where particles that are advected along with the fluid carry all the information. Several possible variations and hybrid methods exist, but a typical one is FLIP, which utilizes particles solely for advection while everything else is computed on grids.

Although they have different strengths and weaknesses, all the mentioned methods, followed by appropriate rendering, are able to capture the characteristic appearance of water: Incidence of light in steep angles leads to refraction caused by the different densities of water and air. Low angles on the other hand result in reflection of light, letting water look similar to a mirror. With its low viscosity water moves in all scales: from large waves spreading over the entire surface to droplets smaller than a fingernail. With all these effects combined it is possible to simulate rather calm waters like lakes realistically, and even more turbulent small-scale simulations look convincing for example pouring a glass of water.

But when it comes to large, highly turbulent waters such as oceans, basic fluid simulations are stretched to their limits. The reason for this is the missing interaction between water and air. While waves in a glass of water can be seen as small deviations in the height of the surface, oceans create breaking waves. That means water is sprayed in the air to generate mist, and air is dragged inside the water in the form of different sized air bubbles and foam on the water surface. To create these details mostly multi-scale methods that rely on adaptively changing the resolution or hybrid methods that use other simulation techniques in detail areas were employed in the last years. The problem with these approaches is that they are often relatively complicated and cannot capture all the phenomena.

But to create visually convincing larger scaled fluids air bubbles, spray and foam are essential and different methods were developed for instance in the movie industry (see [FGP07] and [Gei+06]). However, these papers do not describe in detail which regions inside the fluid are enhanced or how the secondary effects move along with the water. This means it is difficult to recreate their results and add secondary simulations to fluid solvers. Currently, only little other research on secondary effects exists as this

topic became relevant just recently, caused by faster hardware that can now handle large bodies of water. One of the most impactful papers was the pioneering work from Ihmsen et al. [Ihm+12] that presented secondary effects for SPH fluids. Yang et al. [Yan+15] used a depth-image-based analysis to determine detail regions for FLIP, that are enhanced with secondary particles as a tradeoff between quality and efficiency.

This thesis proposes a new adaption of the method from Ihmsen et al. [Ihm+12] for FLIP that solely utilizes criteria from the fluid itself for better quality without expensive high-resolution simulations. First, we show how to scale the data from FLIP to a domain with higher resolution to exploit all the information contained in the underlying simulation. The computations of the criteria, which are proposed by Ihmsen et al. [Ihm+12] to define areas to be enhanced, are adapted for the different neighbor structure in FLIP. The sampling process is only slightly adapted with randomization to prevent regular patterns in the secondary material. Instead of hard-coded thresholds, we refined the secondary particle classification with an adaptive neighbor ratio to get more control over the advection. Details from our implementation regarding lifetime initialization and boundary handling are shown, and we present the rendering setup that produces visually pleasing results for our secondary effects. Then, our model is compared to an implementation of the work from Ihmsen et al. [Ihm+12] regarding visual impression and performance with three differently designed scenes. Finally, we demonstrate that our method can control the amount of detail and the secondary particle movement via the user-defined parameters.

2 Related Work

Processors, memory and graphic cards are constantly increasing in speed and computation power. They greatly advanced the quantity and quality of fluid simulations in the last two decades. Nevertheless, it is still challenging to efficiently simulate various effects of fluid phenomena. We categorized existing work in three classes, according to the fundamental idea in the way simulations are improved.

Combining different techniques to capture details. Using a combination of methods can compensate weaknesses of a single one and thus lead to better results. Furthermore, differently scaled approaches allow to simulate some regions with a high amount of detail if necessary, while the computation speed of a coarser simulation can nearly be preserved.

Thuerey et al. [TRS06] merged a 2D shallow water simulation with a 3D free surface simulation. This enables the creation of many droplets, using a fluid turbulence model that locally increases the fluid viscosity for small vortices. To simulate a fluid and an air phase simultaneously, Hong and Kim [HK03] combined the volume-of-fluid method and the front-tracking method. Wang et al. [Wan+13] used a hybrid approach to combine the grid-based Lattice Boltzmann method with an SPH simulation that interact over a coupling band. This enables large fluid volumes and preserves small details at the same time. Additionally, dynamic spray, foam and bubble particles are added to further enhance the surface details. A FLIP simulation for the fluid is linked to an SPH simulation for droplets in the work of Yang et al. [Yan+14]. Furthermore, they proposed a grid containing density values to represent spray and mist. To prevent a loss of mass the three components are correlated to one another by a transition model. Kim et al. [Kim+06] utilized the particle level set method to simulate water volumes and created small splashes with marker particles. In the rendering stage they smoothed the particle clusters which then contributed to the volumetrically rendered mist density in the air.

Two-way coupled approaches for water and air. Besides the water, two-way coupled approaches also simulate the air, either implicitly by handling the transition area or as an explicit air phase. Thereby, water influences the air and vice versa. This creates high visual realism but due to the computational overhead the simulation takes more

time or the possible number of particles, respectively the maximum grid size is severely decreased.

In the work of Takahashi et al. [Tak+03] water and air are explicitly simulated with the Cubic Interpolated Propagation method. Spray and foam particles that move along in a velocity grid are added in the transition region. Mihalef et al. [MMS09] used a Eulerian-Lagrangian method for the two phases and added sub-scale droplets and bubbles according to the physics-based Weber number. With small adjustments their method can be converted to a post-processing step, at the cost of visual realism. Patkar et al. [Pat+13] utilized bubbles conserved in the level set of a grid-based simulation in combination with Lagrangian particles in under-resolved areas for a bubble simulation. Complex interactions between bubbles like merging and chaotic movement are considered. They also propose a boundary handling scheme for bubbles and a creation criterion near obstacles according to their surface roughness. In a very similar way, the work of Hong et al. [Hon+08] adds bubbles that behave like SPH particles to a Eulerian fluid simulation on an octree grid and models interaction between them. Ihmsen et al. [Ihm+11] likewise employed Lagrangian bubbles for an SPH fluid simulation. Cleary et al. [Cle+07] simulated beer and other carbonated drinks with a mesh-free SPH method. They used a physically motivated model to simulate the transition from dissolved gas to discrete bubbles. The latter two approaches used a foam model in addition. For shallow water simulations, Thuerey et al. [Thu+07b] proposed a model where bubbles are represented as spherical vortices and particles. Foam is created using an SPH approach with surface tension for clustering. Two-way coupled methods allow to combine water with other phases too. For example Baek et al. [BUH15] simulated muddy water with sand and water particles. In addition they modeled the transition from a suspension with medium sized particles to a colloid with small particles to a state where water and sand are mostly separate.

One-way coupled methods and secondary details. A general one-way coupled approach exclusively computes the influence of the water on a representation of air and not vice versa. Often, they can be detached from the main simulation as a secondary effect. As less interactions are computed, these methods are typically less accurate but allow to simulate more details in less time than two-way coupled simulations.

The simplest and fastest way to add detail is applying textures for rendering. This is especially useful for real-time applications where speed is the most important factor and realism only comes second. Chentanez and Müller [CM11] used wave textures and foam maps to enhance their real-time large cell grid simulations. Thousands of particles for spray, mist and foam further improved the visual realism. In a similar way, the method proposed by Thuerey et al. [Thu+07a] for real-time shallow water simulations employed

foam textures. Normally, shallow water simulations cannot represent breaking waves due to the height field as their main component and adding foam would not be particularly useful. But Thuerey et al. determined regions where waves are supposed to overturn and added a particle-based fluid sheet that is moved along with the wave and benefits from foam. For fast bubble simulations, Greenwood and House [GH04] created a method for one-way coupled, particle-based bubbles that are generated in under-resolved regions of a particle level set. The reason for the improved speed is that bubbles are passive, that means they do not influence one another during the simulation. Losasso et al. [Los+08] combined the particle level set method with SPH in a two-way coupled simulation. Afterwards, they utilized a one-way coupled secondary air simulation to add fine detailed foam and mist effects. The method by Yang et al. [Yan+15] is very closely related to our approach, since they used diffuse material in the form of spray, foam and bubble particles as a secondary simulation for FLIP as well. The main difference is the way how regions to enhance are detected: they used time-space analysis of a depth image of the fluid to capture detail regions, while we determine the criteria from the fluid itself. Last and most important, this thesis builds on the work from Ihmsen et al. [Ihm+12]. We implemented their approach to generate unified spray, foam and bubble particles for SPH fluids and adapted it to work for FLIP and grids in general. Both concepts do not feature interactions among secondary particles and this enables the efficient handling of millions of particles.

Additionally, there is research on pure foam, bubble or spray simulations: The work of Kück et al. [KVG02] describes how to simulate foam with spheres and reconstruct the foam surface during rendering. Yue et al. [Yue+15] examined how foam can be simulated as a continuum instead of several single structures that represent bubbles. For instance, this is used to create realistic shaving foam or whipped cream. The method proposed by Zheng et al. [ZYP06] is capable of simulating soap bubbles with a thin liquid film and surface tension using the regional level set method. Last but not least, Nielsen and Østerby [NØ13] proposed a two-way coupled, two-continua approach where water and air coexist at each point in space to simulate spray more accurate than Eulerian methods.

3 SPH and FLIP

The established physical model to describe the behavior of fluids are the Navier-Stokes Equations. As it is still unknown if an analytical solution in three dimensions exists, fluid solvers compute a numerical solution for it instead.

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{\nabla P}{\rho} + \nu \nabla^2 u + g \tag{1}$$

This equation describes the conservation of momentum inside the fluid. Here u denotes the velocity, P is the pressure the fluid exerts, ρ is the fluids density and usually assumed to be constant, ν is the kinematic viscosity coefficient that indicates the thickness of the fluid and g is the acceleration due to gravity. In nature fluids do change their volume and therefore their density to a certain degree, for instance acoustic sound waves in water are nothing more than regions with different densities. But simulating compressible flow correctly is complicated and expensive, instead Equation (2) is used to simulate incompressible flow.

$$\nabla \cdot u = 0 \tag{2}$$

It is called the incompressibility condition which ensures that the fluid does not change its volume. Fluid solvers can be divided in two groups that generally follow contrasting paradigms.

The Lagrangian approach is relatively intuitive: the fluid consists of particles, conceivable as molecules or small drops, that can move arbitrarily in space. They influence one another, for instance two particles coming close repel each other similar to rigid body collisions of spheres. All fluid quantities such as pressure, velocity etc. are stored on the particles. This makes mass conservation effortless and advection very accurate, at the cost of high computational effort for the neighborhood calculation.

The Eulerian approach represents the simulation domain as a grid. Typically it uses a staggered grid, which defines velocities at the cell faces and other quantities at the cell centers. For the most basic simulations every cell is marked either as fluid, empty or solid. The fluid moves by changing the marked cells, but the grid itself is static. On the one hand, this method results in faster simulations than the Lagrangian approach as the regular grid structure simplifies the computation of neighborhood relations and spatial derivatives. On the other hand advection and conserving the mass by keeping the

velocity grid divergence-free is relatively complicated. The main issue of this approach is that it is not possible to preserve fluid details without a dedicated algorithm.

Bridson and Müller-Fischer [BM07] give a very descriptive analogy with weather observation to better understand the two ideas. The Lagrangian approach is like using a weather balloon. It flies along with the fluid, in this case air, and measures the fluid quantities in a radius around it. The Eulerian approach is the analogue to weather stations at different locations on the ground. They always stay in the same place and can only measure the fluid that happens to move near them.

3.1 Smoothed Particle Hydrodynamics (SPH)

The following section is a short summary of the current state-of-the-art report on SPH fluids from Ihmsen et al. [Ihm+14b] that only covers the basic idea of SPH that is necessary for the following chapters.

The SPH method proposed by Monaghan [Mon92] is a Lagrangian approach and describes how any quantity A_j can be interpolated to arbitrary particle positions x_i from the surrounding particles x_j.

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W(x_{ij}, h) \tag{3}$$

$W(x_{ij}, h)$ is a kernel function that weights surrounding particles according to their distance x_{ij} and the smoothing length h that restricts the influence of particles with $x_{ij} > h$. Several different kernel functions are used, but most common is the cubic spline [Mon92].

Algorithm 1: basic SPH algorithm.

foreach *fluid particle i* **do**
  update particle neighbors j;
end
foreach *fluid particle i* **do**
  compute density ρ_i;
  compute pressure P_i;
  $F_i = F_i^{press} + F_i^{visc} + F_i^{bound} + F_i^{grav}$;
  $v_i(t + \Delta t) = v_i(t) + \Delta t F_i / m$;
  $x_i(t + \Delta t) = x_i(t) + \Delta t v_i(t + \Delta t)$;
end

With Equation (3) and one of many ways to approximate the spatial derivatives from Equation (1), Ihmsen et al. [Ihm+14b] show how to compute all the quantities necessary for the simulation: ρ_i can be computed with Equation (3), P_i can for instance be determined by an equation of state, pressure and viscosity forces can be calculated directly from Equation (1) and there are some other forces to consider, like gravity and boundary forces.

This leads to a very basic SPH algorithm, shown in Algorithm 1. Many improvements exist, for example the neighborhood search can be accelerated by uniform grids, different versions of index sort algorithms or with hashing methods like compact hashing. Instead of computing the pressure with an equation of state it is also possible

to solve a pressure Poisson equation for better performance. For a more detailed explanation of the SPH concept we suggest the work of Monaghan [Mon92], Ihmsen [Ihm13] and Ihmsen et al. [Ihm+14b].

3.2 Fluid-Implicit-Particles (FLIP)

The FLIP method, which was first introduced into graphics by Zhu and Bridson [ZB05], is based on a grid-based solver following the Eulerian approach. In this thesis we will not discuss how grid-based simulations work in general and instead refer to the paper from Bridson and Müller-Fischer [BM07] to understand the basic concept. For an efficient implementation for example the approach from Chentanez and Müller [CM11] can be used.

Algorithm 2: FLIP algorithm.

```
foreach grid velocity cell do
    compute weighted average of nearby particle velocities;
end
save grid velocity;
perform all non-advection steps of a grid-based fluid solver;
foreach updated grid velocity cell do
    subtract updated velocity from saved velocity;
end
foreach particle do
    add interpolated velocity difference to particle velocity;
    update particle position using grid velocity;
    push outside of boundaries;
end
output particles;
```

FLIP aims to eradicate the biggest weakness of grid-based schemes, specifically advecting the fluid correctly. This is accomplished by utilizing particles as their strength is a highly accurate advection. Algorithm 2 shows one time step of the FLIP process. Coarsely summarized, the first step is to map the particle velocities to a grid velocity by computing a weighted average and solve the non-advection tasks on the grid. Next, the particle velocity is updated using the interpolated grid velocity. Finally, the particles are moved according to their velocity, for instance with a standard ODE solver.

This technique allows to keep the simulation speed of grid-based approaches and outperforms SPH by a lot. There are approaches combining SPH with FLIP to get the benefits of both methods with increased speed, for example from Cornelis et al. [Cor+14].

In the general FLIP method small details are only partially preserved due to the grid representation and other artifacts exist as well. One of them are thin fluid filaments when small amounts of fluid are in mid-air and the particles tend to stick together. This is caused by the transfer from the grid to the particles and the other way around, where velocities are smoothed twice due to the interpolation. This has a high impact on the visual quality of our method that can be seen in Chapter 5.

4 Secondary Particles for SPH and Adaptations for FLIP

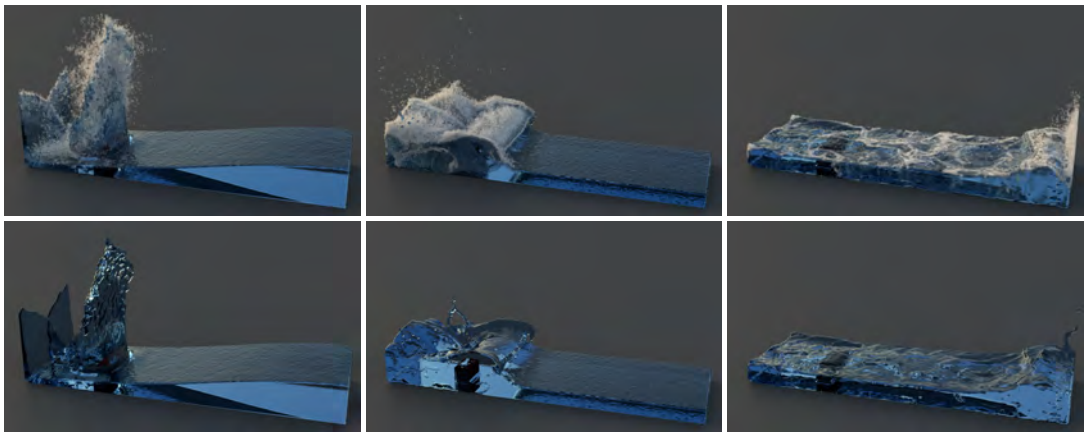

Figure 1: Our method (top) significantly enhances the typical FLIP simulation (bottom) in visual realism using secondary particles, that capture interesting effects of air bubbles, spray, and foam.

After the basic fluid simulation is completed the effects of spray, foam and air bubbles have to be simulated using secondary particles, in the following also called diffuse particles or diffuse material. There are three conceptually different categories of effects that represent the classification of secondary particles in our model (also see Figure 1):

- **spray**: small droplets that move in the air with no connection to the major water volume (left and middle)
- **foam**: many slowly dissolving air bubbles surrounded by a thin liquid film floating on the water (middle and right)
- **bubbles**: air enclosures inside the water volume that rise to the surface (middle)

In the following chapters we will refer to our implementation of the secondary particle method presented by Ihmsen et al. [Ihm+12] as SPH method and to our new adaptation that can be used for FLIP and other grid-based fluids as FLIP method.

The general steps of both methods are relatively similar, at first the data from the fluid simulation has to be processed as described in Section 4.1. Afterwards the likelihood to create diffuse material is determined for each region inside the fluid in Section 4.2. Therefor, three different physically-based potential values are computed for each fluid particle in the SPH method and for each fluid cell in the FLIP method respectively. Section 4.3 discusses the number and position of secondary particles for their initialization. Next, according to the list above each diffuse particle is advected with respect to its classification, which is shown in Section 4.4. Section 4.5 and 4.6 provide details about how secondary particles are dissolved and how we handled boundaries like borders of the simulation domain or other static obstacles. In the final Section 4.7 we summarize the entire secondary particle algorithm and explain how the images used in this thesis were rendered.

4.1 Loading Simulation Data

As a first step, we need to load the data from the basic fluid simulation. For the SPH method the particle positions and velocities for the current time step and the particle positions in the next time step are used later on. Because no further treatment is necessary here, loading the data is trivial.

For the FLIP method it is possible to employ a similarly simple solution: First, we would load the grid where cells are marked as fluid, air or obstacle and then the grid with the current fluid velocity for each cell. FLIP typically utilizes the sub-resolution value two; this means in 3D every cell is sampled by 2^3 FLIP particles. Thus, we would end up using only half of the possible resolution in each dimension for the diffuse material. Therefore, most of the information contained in the FLIP simulation would be neglected. This means small-scale details in the fluid flow would influence the secondary particle distribution only coarsely or not at all.

To address this issue, we propose to make use of a grid with doubled resolution in each dimension and this can be generalized to an arbitrary upscale factor s. Using the coarse grid to create the fine grid is hardly possible as the details are already lost and cannot be restored. Instead, we load the FLIP particles from the basic simulation, scale each particle position by s and mark each grid cell that contains at least one particle as fluid. But due to the non-uniform distribution of FLIP particles, this leads to empty grid cells inside the fluid even for small s.

To correct this we created a particle level set φ from the FLIP particles and mark every grid cell where $\varphi < 0$ holds as fluid cells. It is important to employ an influence radius high enough to erase the gaps. For our implementation with a grid cell size of 1, an influence radius of 1.1 gave good results. For information about the level set

method in general and details on how to create particle level sets the reader may look into the work of Osher and Fedkiw [OF03] or Enright et al. [EMF02]. As φ is used later to compute the surface normals, this does not add computational overhead to the simulation.

The scaled grid velocity can then be obtained by using standard trilinear interpolation of the FLIP particle velocities, but afterwards each velocity component has to be multiplied by s to compensate the larger grid size.

4.2 Potentials for Diffuse Material

In nature there are two different scenarios for a large amount of foam or air bubbles in the water. On the one hand much foam can be observed at large waves: it mostly appears on the crest of the wave and only when it starts to break. The reason for this phenomenon is the interaction of water and air at the wave crest: both phases form a mixture, as water and air are not combined chemically. A good example for this are ocean waves reaching a beach, where the water return flow from the previous wave enforces the overturning of following waves and thereby the generation of foam.

On the other hand air bubbles are typically created when water surrounds a larger portion of air and drags it inside the water. This happens mostly when breaking waves hit the shallow water surface at the ocean or when water hits an obstacle like a boulder.

Additionally, faster streaming water is more likely to mix with air. This effect is not as obvious but when looking at a river after a day of heavy rain and the same river after a week of mild weather, the difference becomes visible. Of course there are other effects to consider in this example as well. But to correlate the amount of spray, bubbles and foam to the speed of the water flow as a simplified model should be precise enough.

In the SPH method there are three main criteria to recreate these effects: for all SPH particles three different values, called kinetic energy potential I_k, trapped air potential I_{ta} and wave crest potential I_{wc}, are computed that influence the number of diffuse particles created by each SPH particle. Out adaptation computes the potentials for all fluid cells from FLIP, and this requires some changes presented in the next sections. Since all calculations lead to arbitrarily high values for each potential, a clamping function Φ is necessary to normalize the potentials to the interval $[0, 1]$.

$$\Phi(I, \tau^{min}, \tau^{max}) = \frac{min(I, \tau^{max}) - min(I, \tau^{max})}{\tau^{max} - \tau^{min}} \tag{4}$$

Here τ^{min} and τ^{max} denote user-defined thresholds, we describe their effect in detail in Section 5.3.1.

Two of the criteria, the trapped air potential and the wave crest potential, need

neighboring particles or grid cells for their computation. Therefore, a kernel function is necessary to weight the contribution to the computation similar as seen in Section 3.1. With increasing distance the weight should approach zero relatively fast. For the SPH method commonly used kernels [MCG03], for instance the cubic spline [Mon92], give mediocre results at free surfaces because of the low number of neighbors in these regions. For higher accuracy Ihmsen et al. [Ihm+12] propose a different radially symmetric weighting function, and our experiments showed that for the FLIP method it is more appropriate too.

$$W(x_{ij}, h) = \begin{cases} 1 - \frac{||x_{ij}||}{h} & \text{if } ||x_{ij}|| \leq h \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In this weighting function h is the kernels smoothing length, also called support radius. For the SPH method h should be equal to the influence radius of the fluid simulation. For the FLIP method we adapted h as described in Section 4.2.2. x_{ij} denotes the distance vector from the current position i where the kernel is employed to the neighbors position j. Since the position of a grid cell is not unique; we define it to be the cells center.

At this point the reader may be asking himself why computing all quantities for grids is necessary although FLIP already features particles that could be used instead. The crucial point is that FLIP particles are not distributed uniformly while SPH particles are. For instance a small SPH region, being not near the surface, will contain approximately the same number of particles over the entire simulation time. But a FLIP region may have a large deviation of the particle number over a few time steps. Starting from the premise that all particles from FLIP create secondary particles this leads to high distortions in the number of secondary particles created in each region and would notably reduce the visual quality. Creating secondary particles for each grid cell instead guarantees that all regions with the same potentials create a similar number of particles.

4.2.1 Kinetic Energy Potential

To measure the flow speed of the water in the SPH method the kinetic energy $E_{k,i} = 0.5\, m_i\, v_i^2$ for every particle i is used. Correspondingly, we can compute the kinetic energy of the fluid in each grid cell j by $E_{k,j} = 0.5\, m_j\, v_j^2$. The resulting kinetic energy potential is $I_k = \Phi(E_{k,j}, \tau_k^{min}, \tau_k^{max})$, where τ_k^{min} and τ_k^{max} are user-defined constants.

Note, that the mass m_j could be computed exactly as density and volume of the simulated liquid are known. But since τ_k^{min} and τ_k^{max} can be chosen as desired and have to be fine-tuned, it is possible to just use an arbitrary constant for the mass when adjusting the thresholds accordingly.

4.2.2 Trapped Air Potential

Air gets trapped in the water where it forms vortices and other highly turbulent areas. Thus, the curl operator can be a reasonable choice for detection. But to capture impacts as well, for example when a breaking waves hits the shallow surface, the FLIP method relies on scaled velocities differences of neighboring fluid cells. This idea was first introduced by Ihmsen et al. [Ihm+11] and originally used to create air bubble simulations for SPH. Later it was advanced by Ihmsen et al. [Ihm+12] for the secondary particle model for SPH.

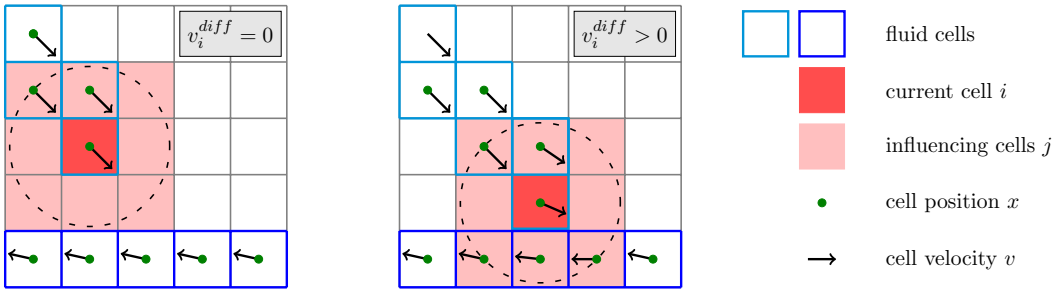

Figure 2: Schematic side view of a wave crest (light blue) hitting the water surface (dark blue).

To compute the velocity differences for the grid cell i we iterate over all neighbor cells j having a maximum metric, also called chessboard distance, of less or equal than a radius r. In 3D this leads to a neighbor cube with cell i in the middle, consisting of $(2r+1)^3 - 1$ cells. The red tiles in Figure 2 show a 2D example of this concept.

We have to weight all cells j according to their distance to i, using the weighting function W from Equation (5). We set $h = \sqrt{3}\,r$ as this describes a circumscribed sphere of the cell centers using the entire neighbor cube (dashed circles in Figure 2).

Summing up the weighted velocity differences results in $\sum_j \|v_{ij}\| W(x_{ij}, h)$. Additionally, the SPH method scales the velocity differences with $(1 - \hat{v}_{ij} \cdot \hat{x}_{ij})$ to increase the importance of particles that move towards each other and decrease it for particles moving away from each other. As before, this works in the same way for grids, here $\hat{x}_{ij} = (x_i - x_j)/\|x_i - x_j\|$ is the normalized distance vector and analogue $\hat{v}_{ij} = (v_i - v_j)/\|v_i - v_j\|$ is the normalized relative velocity. Due to the definition of the scalar product

$$\hat{v}_{ij} \cdot \hat{x}_{ij} = \begin{cases} \in [-1, 0) & \text{if } \hat{v}_{ij} \text{ and } \hat{x}_{ij} \text{ are directed towards each other} \\ 0 & \text{if } \hat{v}_{ij} \text{ and } \hat{x}_{ij} \text{ are perpendicular} \\ \in (0, 1] & \text{if } \hat{v}_{ij} \text{ and } \hat{x}_{ij} \text{ are directed apart from each other} \end{cases} \tag{6}$$

multiplying $(1 - \hat{v}_{ij} \cdot \hat{x}_{ij})$ has the desired scaling effect. So the scaled velocity difference v_i^{diff} reads:

$$v_i^{diff} = \sum_j \|v_{ij}\|(1 - \hat{v}_{ij} \cdot \hat{x}_{ij}) W(x_{ij}, h) \tag{7}$$

Finally, the trapped air potential is computed as $I_{ta} = \Phi(v_i^{diff}, \tau_{ta}^{min}, \tau_{ta}^{max})$. Again, with two user-defined clamping thresholds τ_{ta}^{min} and τ_{ta}^{max}. The left half of Figure 2 illustrates a scenario where the trapped air potential for the red cell is zero as the surrounding velocities point in the same general direction. On the right it is large because the fluid cells from the surface below now influence the computation.

4.2.3 Wave Crest Potential

Foam and spray occur on wave crests, especially when the wave starts to overturn. The waves top part is no longer supported by water below it, so it can only mix with air. Additionally, high speed differences between air and water on larger waves enforce this phenomenon. Similar to the SPH method, we use surface normals to estimate the surface curvature and detection functions to differentiate wave crests from other highly curved regions.

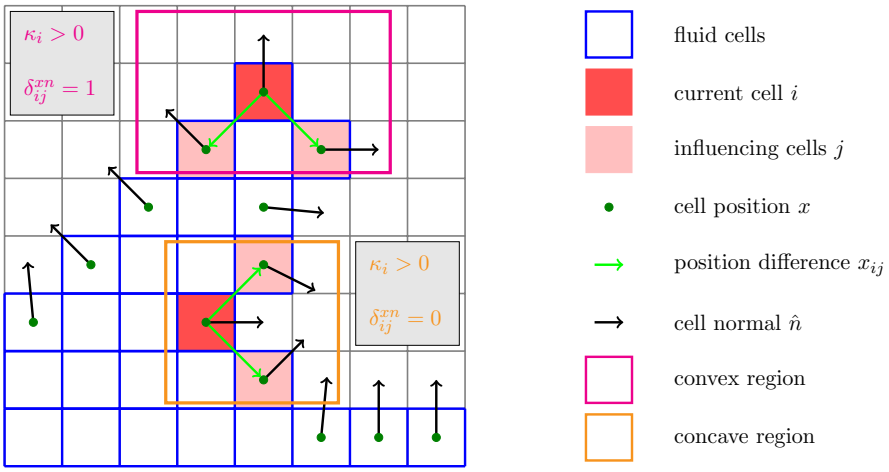

Figure 3: Schematic side view of a wave (blue) containing two exemplary regions of interest (magenta and orange). Both red cells possess high surface curvature κ caused by the orientation of the neighboring normals.

At first, we can re-utilize the level set φ that was used to fill the gaps in the marker grid in Section 4.1 to compute the surface normals. This is done by using the gradient

of φ as it is equal to a vector perpendicular to the tangent plane of the surface. This only works for differentiable points of the level set but we can assume this without loss of generality. We want to create normals only near the surface because inside the fluid the gradients Euclidean norm is very small, so normalizing can lead to normals pointing in nearly arbitrary directions. Therefore, we extrapolate φ to create a signed distance function [Bri16]. Additionally to indicating if a cell is below the surface or not, it also contains the distance to the surface. It is then used to restrict the normal creation to areas near the surface. Next, all normals n are normalized according to $\hat{n} = n/\|n\|$.

Considering two cells i and j with corresponding normals $\hat{n}_i$ and $\hat{n}_j$. With the same reasoning as seen in Equation (6) $(1 - \hat{n}_i \cdot \hat{n}_j)$ is roughly zero when the normals face in the same direction and is larger for greater angles between the normals. To estimate the surface curvature κ for cell i, we can iterate over all neighbors in a neighbor cube and weight them in the exact same way as for the trapped air potential.

$$\kappa_i = \sum_j (1 - \hat{n}_i \cdot \hat{n}_j) W(x_{ij}, h) \tag{8}$$

Figure 3 contains several fluid cells with high surface curvature, two of them are marked in red. Evidently, high curvature does not only appear on wave crests, but for instance at the waves bottom right in the transition zone to the relatively shallow surface too. To distinguish a wave crest from other highly curved regions, its convexity can be used since wave crests are inevitably convex and never concave. A convexity detection function can look like this:

$$\delta_{ij}^{xn} = \begin{cases} 0 & \text{if } \hat{x}_{ij} \cdot \hat{n}_i \geq 0 \\ 1 & \text{if } \hat{x}_{ij} \cdot \hat{n}_i < 0 \end{cases} \tag{9}$$

To understand why this function detects convex regions, both regions in Figure 3 feature the position difference vectors x_{ij} in light green and the normal n_i. The term $\hat{x}_{ij} \cdot \hat{n}_i$ indicates if the angle between x_{ij} and n_i is acute and therefore the region is concave or if the angle is obtuse and thus the region convex.

But there is one small problem when using convexity: there are some convex regions that are not wave crests. They typically occur when fluid is added to the simulation or due to the boundaries of the domain. The reason therefor are artificial edges, for example the simulation borders almost always contain edges and the water is forced to form an edge there as well. Excluding these regions can be done by checking if the fluid moves in the normal direction because that is not possible at almost all artificial edges. Equation (10) shows a simple way to do this utilizing the scalar product to verify that both vectors, the normal and the velocity of the current cell, point in similar directions. The value 0.6 was heuristically determined by Ihmsen et al. [Ihm+12] and

worked for our adaptation likewise. If some artificial edges are still recognized we suggest to increase it accordingly, but too high values can mean that some actual wave crests are not enhanced correctly.

$$\delta_i^{vn} = \begin{cases} 0 & \text{if } \hat{v}_i \cdot \hat{n}_i < 0.6 \\ 1 & \text{if } \hat{v}_i \cdot \hat{n}_i \geq 0.6 \end{cases} \tag{10}$$

Summarizing, the adjusted surface curvature $\tilde{\kappa}_i$ that discards concave regions and artificial edges can be computed by multiplying κ_i with both detection functions accordingly.

$$\tilde{\kappa}_i = \delta_i^{vn} \sum_j \delta_{ij}^{xn} (1 - \hat{n}_i \cdot \hat{n}_j) W(x_{ij}, h) \tag{11}$$

Due to the convexity detection function δ_{ij}^{xn} the concave regions end up with $\tilde{\kappa}_i = 0$ while the convex regions yield $\tilde{\kappa}_i > 0$. With two more thresholds τ_{wc}^{min} and τ_{wc}^{max} given by the user, the wave crest potential is $I_{wc} = \Phi(\tilde{\kappa}_i, \tau_{wc}^{min}, \tau_{wc}^{max})$.

4.3 Sampling and Initialization

With the three potentials computed for each particle in the SPH method and for all fluid cells for the FLIP method respectively, we can now add the secondary particles to the simulation. Ihmsen et al. [Ihm+12] construct a cylinder around every SPH particle for this purpose. It has the same radius as the fluid particle and its height is determined by the particles current position at time step t and the position at the next time step $t + \Delta t$ in direction of the particles velocity. The secondary particles are distributed uniformly inside this cylinder and their number is determined by the three potentials.

Our adaption for grids works in a similar fashion. Due to the regular alignment of the grid using the center x_f of fluid cell f as a starting point for the sampling process is not a good idea, because this leads to unrealistic regular foam patterns like strips. To avoid this, three random variables $X_{\Delta x}$, $X_{\Delta y}$ and $X_{\Delta z}$ each uniformly distributed in range $[-r_f, r_f]$ are used, where r_f is half of the cells diameter. The new starting point $\tilde{x}_f$ of the sampling process is now uniformly distributed over the entire volume of f, with:

$$\tilde{x}_f = \begin{pmatrix} X_{\Delta x} \\ X_{\Delta y} \\ X_{\Delta z} \end{pmatrix} + x_f \tag{12}$$

An illustration with exemplary assigned random variables can be seen in Figure 4. To get the current velocity vector v at $\tilde{x}_f$ standard tri-linear interpolation of the grid velocity can be employed.

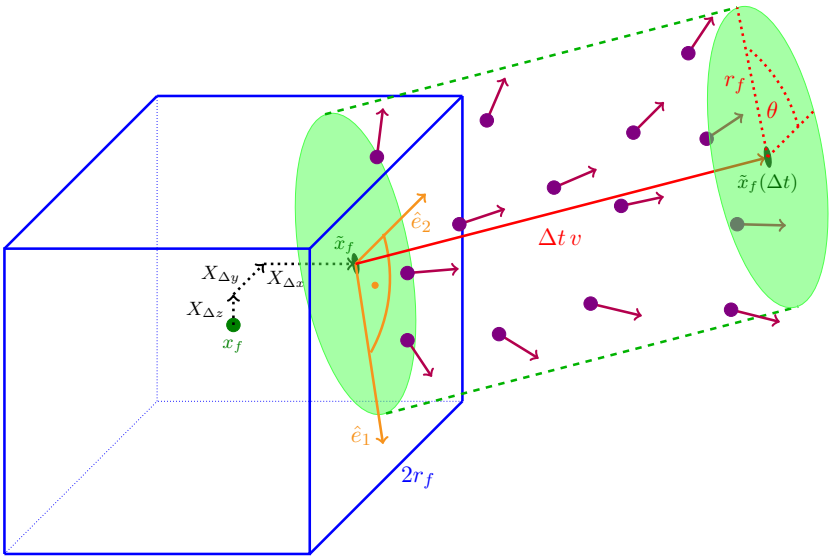

Figure 4: Secondary particles (purple) generated by a fluid cell f, sampled uniformly in a cylinder with radius r_f positioned at $\tilde{x}_f$. The velocity for each secondary particle is determined by their position relative to $\tilde{x}_f$ and the velocity v.

The bottom of the cylinder is represented by the orthogonal plane to v with the origin $\tilde{x}_f$ and the orthogonal unit direction vectors $\hat{e}_1$ and $\hat{e}_2$ shown in orange in Figure 4. These are obtained when normalizing

$$e_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \cdot v \qquad \text{and} \qquad e_2 = v \times e_1. \tag{13}$$

To estimate the position $\tilde{x}_f(\Delta t)$ where the portion of fluid currently inside cell f ends up in the next time step Δt, the vector v can be scaled with Δt and its norm is the height of the sampling cylinder.

Alongside with the SPH method, we introduce two more constants k_{ta} and k_{wc} that allow the animator to control the amount of detail by limiting the maximum number of diffuse particles generated per fluid cell and time step. We describe their influence in Section 5.3.2. The number of secondary particles for each cell can then be computed according to Equation (14).

$$n_f = I_k(k_{ta} I_{ta} + k_{wc} I_{wc})\Delta t \tag{14}$$

To initialize them inside the cylinder, three additional random variables X_r, X_θ, X_h that are uniformly distributed in the interval [0,1] are employed. Similar to polar

coordinates $r = r_f \sqrt{X_r}$ indicates the particles distance to $\tilde{x}_f$ on the cylinders base plane and $\theta = X_\theta \, 2\pi$ is the azimuth or angular position that shows the rotation around the cylinders axis. Additionally, $h = X_h \, \|\Delta t v\|$ is the particles distance from the base plane or height inside the cylinder. Together they form a cylindrical coordinate system that can describe all points inside the cylinder and due to the random variables it is sampled uniformly. Figure 4 features a schematic depiction of it in red. The uniform sampling is very important as the volumetric impression of foam and spray is entirely lost when the particles are not distributed properly. Finally, the position of each diffuse particle x_d can be described by:

$$x_d = \tilde{x}_f + r\hat{e}_1 \cos\theta + r\hat{e}_2 \sin\theta + h\frac{v}{\|v\|}. \tag{15}$$

Supplementary to the fluid velocity, secondary particles get an additional amount of laterally directed velocity relative to their distance to the cylinder axis. This is necessary to add the possibility of splashes different from the main fluid movement.

$$v_d = r\hat{e}_1 \cos\theta + r\hat{e}_2 \sin\theta + v \tag{16}$$

The purple vectors representing secondary particle velocities in Figure 4 show the influence of Equation (16). Particles near the axis have velocities with almost identical direction as v, while particles further at the outside tend to move more sideways.

4.4 Advection

The next step is moving the diffuse material. Both models do not consider interactions among secondary particles, which is the main reason why it is efficiently possible to simulate large amounts of spray, foam and bubbles. Instead, the particles are advected by the fluid movement. To accomplish this, we once again iterate over neighboring fluid cells in a neighbor cube and weight their velocity according to the distance difference. This time a commonly used kernel $K(\|x_{ij}\|, h)$ like the cubic spline is sufficient for weighting.

4.4.1 Classification

Secondary particles are divided in three types: spray flying through the air, foam near the water surface and air bubbles inside the water (see Figure 5). Since all types are advected differently, at first we need to classify each particle in one category. Our implementation of the SPH method does this by building a neighbor list l for all diffuse particles p that contains their respective fluid neighbors. Then the actual classification is

done via constant thresholds on its size $|l|$, for instance using the following classification function with thresholds determined by Ihmsen et al. [Ihm+12]:

$$type(p) = \begin{cases} \text{spray} & \text{if } |l| < 6 \\ \text{air bubble} & \text{if } |l| > 20 \\ \text{foam} & \text{else} \end{cases} \tag{17}$$

Other ideas to classify particles exists as well, for example Müller et al. [MCG03] use the gradient of the density field to find particles near the surface but because we consider the fluid to be nearly incompressible, using the number of neighbors is simpler and gives similar results.

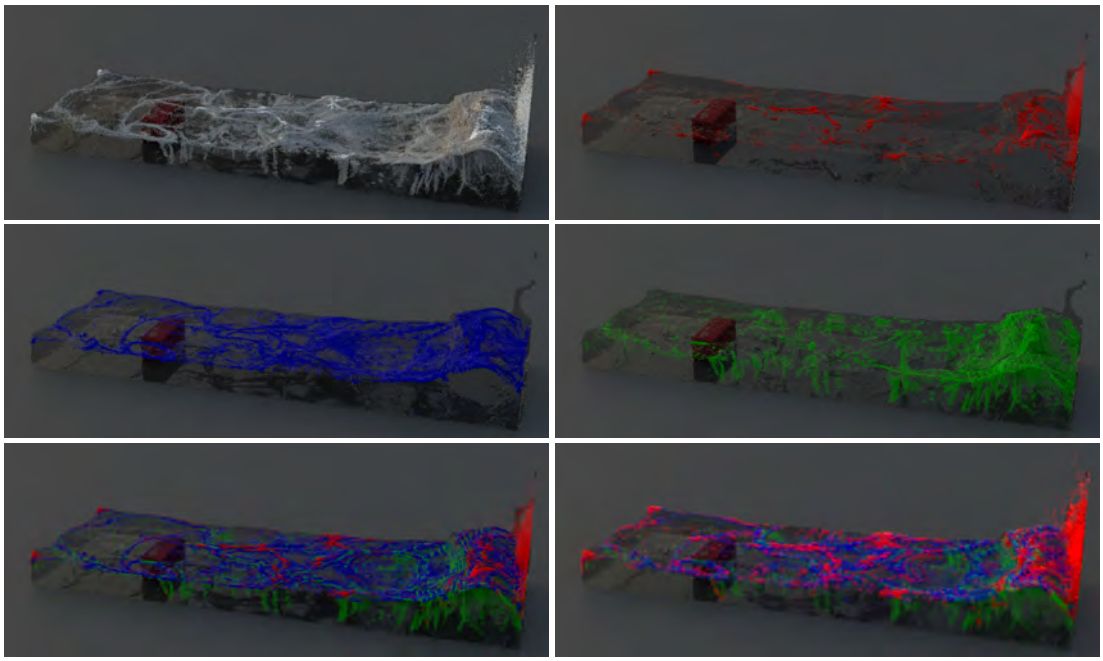

Figure 5: Particle classification with nearly transparent water and particles rendered as discrete spheres instead of volume rendering. Spray particles are red, foam is blue and air bubbles are green in all images. The four upper images show the initial frame and the classified particles in their respective color. The bottom row displays all of them combined, with volume rendering on the right and without it on the left.

For the FLIP method we propose an approach that can handle different radii of the neighbor cube without additional effort and at the same time simplifies advection near

boundaries. First, we calculate the total number of neighbors $n_{all} = (2r+1)^3 - 1$ for each fluid cell i and subtract the number of neighbors that are boundary or obstacle cells n_{wall}. This results in the maximum number of cells that could be potentially fluid cells $n_{maxFluid} = n_{all} - n_{wall}$. Secondly, we count the number n_{fluid} of actual neighboring fluid cells. With these two quantities a neighbor ratio

$$n_r = \frac{n_{fluid}}{n_{maxFluid}} \in [0,1] \tag{18}$$

can be obtained for every cell, that indicates to what extent it is surrounded by fluid. For example, $n_r = 1$ means i is entirely encompassed by fluid cells or $n_r = 0$ implies that no fluid cells are around.

With the neighbor ratio and two user-defined classification thresholds c_s and c_b it is easy to determine the type of a diffuse particle p positioned in fluid cell i with Equation (19). This is now independent from the size of the neighbor cube r and from the upscale factor from Section 4.1.

$$type(p) = \begin{cases} \text{spray} & \text{if } n_r(i) < c_s \\ \text{air bubble} & \text{if } n_r(i) > c_b \\ \text{foam} & \text{else} \end{cases} \tag{19}$$

Here, c_s and c_b control the transition between air bubbles and foam respectively foam and spray. A detailed description of their influence can be found in Section 5.3.3.

4.4.2 Spray

Spray particles have only very few neighboring fluid cells and their influence on the particle movement is negligible. Therefore, we can update the spray particle position and velocity with a simple semi-implicit Euler step. Only gravity g and external forces F_{ext} have a considerable influence on the movement.

$$v_{spray}(t + \Delta t) = v_{spray}(t) + \Delta t \, (\frac{F_{ext}(t)}{m} + g) \tag{20}$$

$$x_{spray}(t + \Delta t) = x_{spray}(t) + \Delta t \, v_{spray}(t + \Delta t) \tag{21}$$

This works in the exact same way for the FLIP and the SPH method.

4.4.3 Foam

To the contrary, foam is heavily influenced by the surrounding fluid flow because it is supposed to be floating along on the water surface. As a first step in the FLIP method,

it is necessary to find the fluid cell in which the current diffuse particle i is located. We can build a neighbor cube of edge length r like in Section 4.2.2 consisting of cells j around it. Note, that here boundary cells have to be explicitly excluded from the computation, otherwise foam particles would sometimes stick the to walls.

To move the foam along with the fluid we use the weighted averaged velocity

$$\tilde{v}_i(t+\Delta t) = \frac{\sum_j v_j(t)\, K(\|x_{ij}\|, h)}{\sum_j K(\|x_{ij}\|, h)} \tag{22}$$

from the surrounding fluid. The SPH method uses $v_j(t+\Delta t) = (x_j(t+\Delta t) - x_j(t))/\Delta t$ instead of $v_j(t)$ in the numerator, where j denotes neighboring fluid particles. $x_j(t+\Delta t)$ can simply be loaded from one time step ahead. But for grids we do not know $x_j(t+\Delta t)$ as in contrast to particles there is no correlation between the time steps. But we can estimate $x_j(t+\Delta t) = x_j(t) + \Delta t\, v_j(t)$ with the current velocity. Plugging this term into the formula used by the SPH method leads to Equation (22) because through rearranging, $x_j(t) - x_j(t)$ resolves to zero and Δt can be canceled. We tried to load the grid velocity from the next time step too and just evaluate $v_j(t+\Delta t)$ in the neighbor cube there, but that did not produce visually plausible results. The reason is that this would mean to consider velocities from fluid portions which are not near i in the current time step.

Finally, the foam particle is updated with the weighted averaged velocity. Since foam particles should be restricted to the water surface, only their positions and not their velocities are updated.

$$x_{foam}(t+\Delta t) = x_{foam}(t) + \Delta t\, \tilde{v}_i(t+\Delta t) \tag{23}$$

4.4.4 Bubbles

In a very similar way, the water flow is a strong influence on the air bubbles and drags them along. Additionally, bubbles rise to the water surface due to the high density difference between water and air. The drag effect is created as above with the weighted averaged velocity. The buoyancy effect can be modeled by an inverse gravity that is directed upwards. The two effects are multiplied with two constants k_d and k_b, allowing the user to control the bubble movement (see Section 5.3.4 for a comparison of different values for k_d and k_b).

$$v_{bubble}(t+\Delta t) = v_{bubble}(t) + \Delta t\, (-k_b g + k_d \frac{\tilde{v}_i(t+\Delta t) - v_{bubble}(t)}{\Delta t}) \tag{24}$$

$$x_{bubble}(t+\Delta t) = x_{bubble}(t) + \Delta t\, v_{bubble}(t+\Delta t) \tag{25}$$

4.5 Dissolution

At the transition zone between water and air foam dissolves after a short time. For example, this can be observed when ships move across the water as they leave behind a trail of foam that is wide directly behind the ship and becomes narrower further behind. Foam is a paragon of the law of exponential decay, that means at the start it dissolves rapidly and then steadily slower, approaching a state without foam. This can be verified with the simple act of pouring a glass of beer and measuring the height of the beer foam at constant time intervals. Plotting the measured points will always lead to an approximation of an exponential function with a negative exponent.

For the dissolution each secondary particle gets a lifetime value l at its initialization that is reduced by Δt every time step. Once l is below or equal to zero the particle is removed.

To capture the effect of the exponential decay in a visually plausible but simple way, our experiments showed that it is sufficient if large foam clusters will persist longer than smaller ones. Therefor, we can initialize large particle groups with a high value of l and isolated particles with a low value. Computing additional quantities is not required as we already have a measurement of regions where a large number of particles is created, with the potentials from Section 4.2. The lifetime can then be initialized according to the three potentials of the generating fluid particle or fluid cell respectively, where each potential contributes equally:

$$l = (l_{max} - l_{min}) \frac{I_k + I_{ta} + I_{wc}}{3} + l_{min} + 0.1 X_l \tag{26}$$

X_l is a random variable in range $[0, 1]$ that is not stringently required, but helps if some regions with few diffuse particles dissolve with a regular pattern. The factor in front of it can be bigger to suppress regular dissolution even further. However, too large values may destroy the impression of realistic foam, because particles disappear too randomly. Equation (26) enforces $l \in [l_{min}, l_{max} + 0.1]$ where l_{min} and l_{max} denote thresholds for maximum and minimum lifetime. Their effect is shown in Section 5.3.5.

4.6 Boundaries

The neighbor ratio presented in Section 4.4.1 ensures that in the FLIP method secondary particles are advected correctly near boundary cells. But the SPH method uses a simpler classification with neighbor lists. In our SPH implementation obstacles are realized using stationary particles on its borders that apply repelling forces to nearby fluid particles. These stationary particles are added to the neighbor lists as well because

otherwise some secondary particles could be incorrectly classified as foam, for instance near the ground due to a lower number of surrounding particles there.

But with the stationary particles, now spray particles near walls are considered to be foam as the lower spray threshold from Equation (17) is exceeded. This causes them to stick to walls because they cannot be advected with the fluid flow as there is now fluid around. To prevent this, we additionally check if the neighbor list solely consists of obstacles particles and use spray advection if it is the case.

Due to estimates in the particle creation and advection process, and additionally to small numerical errors, it is effectively impossible to prevent some diffuse particles from ending up inside obstacles or boundaries. We tried several boundary handling approaches for them, each resulting in a different artifact:

Position correction. Moving penetrating particles out of the walls along the walls normal leads to large amounts of diffuse material clumping when a wave hits a wall. This is enforced by the high trapped air potential in these regions. As a result, the noise and the duration of the rendering process is increased drastically.

Repelling forces. Adding repelling forces ensuing from the boundary particles or boundary cells respectively results in similar problems. Furthermore, spray particles can obtain implausible trajectories.

Reflection. Reflecting the velocity of each particle on the walls surface normal in different (physically impossible) angles causes many visually noticeable wrong trajectories, but prevents clumping.

Deletion. Deleting penetrating particles prevents clumping and incorrect spray movement, but can cause the impression that walls suck in nearby diffuse material and absorb it. In general, this leads to the least visual disturbances so we employed it for our implementation.

Depending on the scene setup, very thin obstacles like walls may cause problems as fast moving particles can end up in front of a wall in one time step and already passed through it in the next step. This effect is known as tunneling. Here it can help to check if obstacles are in the current trajectory of each secondary particle by simulating smaller time steps. This is done by temporarily multiplying rising factors from $[0, 1]$ to Δt in the position updates in Section 4.4 and checking if the resulting position lies in a cell that is marked as an obstacle. If that is the case the particle is deleted. Too many anti tunneling checks for each particle can decrease the performance; then we recommend

to broaden the walls in the scene or change the setup to reduce the fluid speed before the impact.

4.7 Implementation and Rendering

Algorithm 3 summarizes all tasks that have to be completed during one time step to add spray, foam and air bubbles in the form of secondary particles to a fluid simulation. We always utilized $r = 2$ from Section 4.2.2 to create the neighbor cube, that is used for the wave crest potential, the neighbor ratio and all advection steps. The only exception to this is the trapped air potential where $r = 1$ already produced high values for the velocity differences.

For the fluid simulations we employed an IISPH [Ihm+14a] with adaptive time stepping similar as proposed by Ihmsen et al. [Ihm+10] and the FLIP implementation presented by Zhu and Bridson [ZB05]. Our adaptation of the secondary particle model for FLIP can easily be transferred to entirely grid-based simulations since all quantities are computed on grids. Obviously, the upscale factor has to be smaller or equal to one then, because in contrast to the FLIP method no additional information exists that could be used for a higher resolution.

Algorithm 3: One time step for the secondary particle algorithms for SPH and FLIP.

```
load data from fluid simulation;
scale data to higher grid resolution;
compute particle level set φ;
foreach fluid particle / fluid cell do
    compute surface normal;
    compute potential I_k, I_ta and I_wc;
    initialize secondary particles;
    update neighbor ratio;
end
foreach secondary particle do
    update particle neighbor list;
    classify and advect particle;
    handle boundaries;
    update lifetime;
    delete if dissolved;
end
output secondary particles;
```

Using a level set generated from the SPH and FLIP particles in an upscaled domain, we created a fluid mesh for every frame with the marching cubes algorithm.

For rendering we used the ray-tracing render engine called Cycles included in the open-source software Blender v2.78. The fluid mesh is then rendered with a blue tinted glass shader combined with volumetric absorption to adjust the amount of light in deeper water regions. For some images in Section 5.3 where the air bubbles are focus of the image, additionally a transparent shader was applied.

Next, for each frame a cubical three dimensional density texture with a resolution of 512^3 voxels is built from the secondary particles. The particles influence the density

value for the voxel they are mapped to and its neighborhood with a volume radius and a density factor. The finished texture is then applied to a domain cube which is rendered with a volume render technique through a shader that combines volumetric absorption and volumetric scattering. Thereby, the texture is accessed via a bounding volume hierarchy for more efficient ray tracing. To create a smooth, foam-like impression the density values are interpolated with tri-linear interpolation.

We used two different textures for most of our renders, one for foam and bubbles and the other only consisting of spray particles. This can achieve increased realism as spray needs to look more compact and sharper, for example by employing a slightly smaller volume radius, a higher density factor and nearest-neighbor interpolation instead. Some images even required a single texture for every particle type, for instance when coloring the particle types with three different tints in Figure 5. These renders used up large amounts of memory due to the dynamically created textures, and the rendering speed decreased by a lot as nested volumes are really expensive to render. This means render times can exceed simulation times by up to two orders of magnitude.

Nevertheless, using density textures looks a lot better than rendering every particle as a single sphere by creating a slightly blurred look, similar to real foam and spray (as a comparison see the bottom row of Figure 5). In addition, this prevents an issue for a large number of particles with specular or transparent surfaces that are rendered as discrete spheres: the exponential growth in the number of reflection and transmission rays. That makes appropriate ray tracing very difficult and can cause much noise in the resulting images. This is a well-known problem, for example mentioned by Jakob and Marschner [JM12]. A different rendering approach was proposed by Teschner et al. [Aki+13]. It is a GPU-based multi-pass technique that combines images in screen space for increased rendering speed, while still producing visually similar results to our method.

5 Results

In the following, we present the results of our work by showcasing three different FLIP simulations enhanced with our secondary particle model. Then, their visual impression is compared to our implementation of the SPH method proposed by Ihmsen et al. [Ihm+12] in Section 5.1. In the second Section 5.2 we analyze the performance of both methods and also mention the speed of the underlying fluid simulations. Finally, an overview of the influence of most user-defined constants from our model is given in Section 5.3.

Table 1: Domain size, frames, particle numbers and simulation timings for all scenes.

Scene	Size (l x h x w)	Frames	#Fluid	#Diffuse (max. / avg.)	Time (fluid / diffuse)
Cube FLIP	166 x 156 x 56	300	664k	3.30m / 1.42m	15.7 min / 24.1 min
Cube SPH	134 x 126 x 46	300	341k	4.06m / 1.35m	375.7 min / 34.5 min
Stairs FLIP	206 x 206 x 106	300	853k	7.82m / 4.25m	46.2 min / 53.1 min
Stairs SPH	166 x 166 x 86	300	444k	4.77m / 2.91m	543.7 min / 52.1 min
Swirl FLIP small	206 x 86 x 206	500	2.76m	2.03m / 0.99m	81.9 min / 116.4 min
Swirl FLIP large	206 x 86 x 206	500	2.76m	7.43m / 3.50m	81.9 min / 139.0 min

Table 1 contains data for the scenes used in the following; the constant comparisons from Section 5.3 are not included. For the secondary simulations for FLIP we always used an upscaling factor of two, leading to a domain with an eight times larger volume than the underlying fluid simulation. All simulations were performed and measured on a Windows machine with a quad-core Intel i5-6500 CPU with 3.20 GHz.

5.1 Comparison between SPH and FLIP

Cube Scene. Figure 6 shows different frames of the Cube Scene. It consists of a breaking dam and a small cuboid as an obstacle to generate additional splashes and more air entrainment. On the left half the water is simulated with FLIP and on the right with SPH. The outer columns solely show the underlying fluid simulations. While FLIP produces thin fluid sheets and filaments, SPH generates many small droplets in the top row. The wave in the middle row looks similar although SPH still has more details. FLIP has less splash details, but small waves on the water surface are preserved much finer and longer as visible in the bottom row.

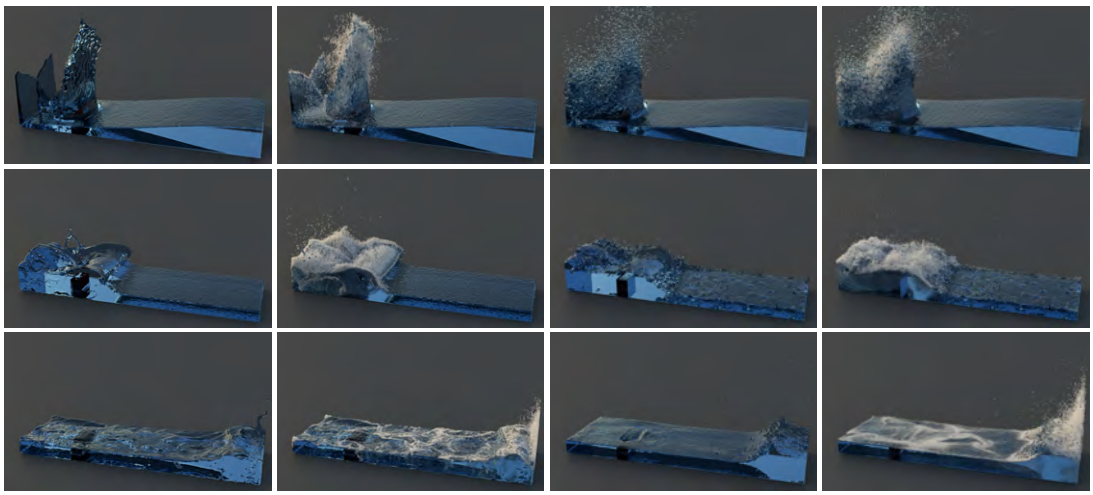

Figure 6: Cube Scene. Comparison of FLIP (left half) and SPH (right half), each with and without diffuse material. The frames 40, 96 and 189 are displayed from top to bottom.

As shown in Figure 6, the diffuse material alleviates the differences between SPH and FLIP such that their simulation results look more alike. The general weaknesses of the underlying fluid remain but are attenuated: SPH produces more realistic spray while the spray in the FLIP method tends to stick together seen in the top row. In the bottom row the small surface waves for SPH are enhanced but FLIP still looks more convincing here. Air bubbles only play a minor role for the visual impression and even some bubble formations in calm water on the left side only slightly harm the realism. These are produced by small errors in the marker grid in the upscale process from Section 4.1, caused by inaccurate regions in the particle level set.

Stairs Scene. The Stairs Scene presented in Figure 7 comprises three steps with equal height difference and a cube of water on top of the uppermost. Small railings prevent the water from pouring straight downwards and instead force it in a sinuosity similar to the shape of some fish ladders. Afterwards, it drops into a cube of stagnant water below as a small waterfall. With its more continuous flow, this scene creates a more natural look than the Cube Scene. Because of the thin walls, it is a challenge for both methods especially with respect to tunneling and boundary handling. Without the diffuse material in Figure 7 the SPH method looks better in both frames due to the impact and the waterfall with a lot more splashes. Once again the secondary particles weaken the divergent impression and increase the visual plausibility.

The upper images with the diffuse particles emphasize the impact of water on the railing. The SPH method has the edge over the FLIP method, but without the direct comparison our adaptation can still be classified to create convincing spray. Apart from the basic fluid movement the waterfall in the lower image looks very similar. Once more the foam patterns are much more detailed in our FLIP method than for SPH.

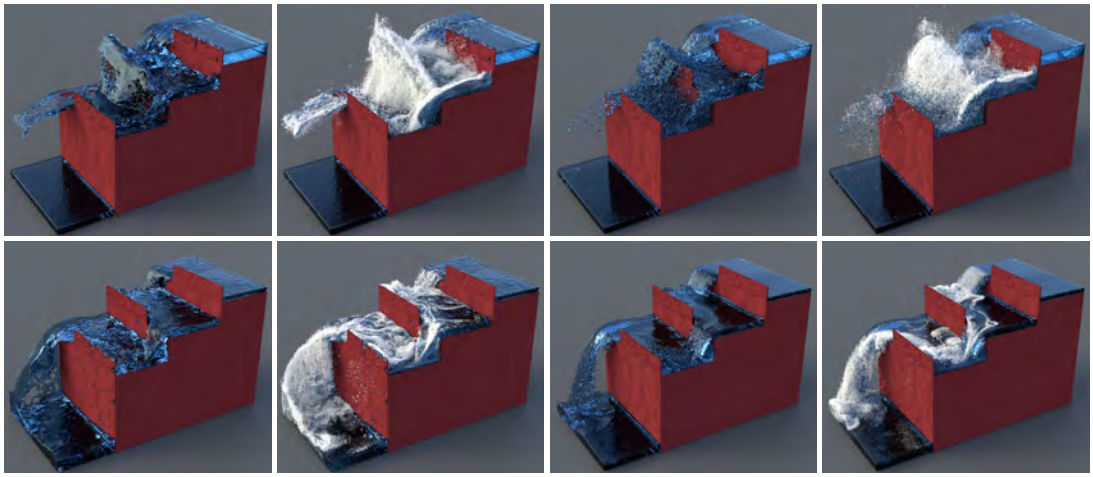

Figure 7: Stairs Scene. Comparison of FLIP (left half) and SPH (right half), each with and without diffuse material. Frame 91 is shown in the upper row, Frame 299 below.

In conclusion, the strengths and weaknesses of the two approaches are closely related to the underlying fluid simulations. On the one hand SPH looks better for spray effects as it spreads diffuse material relatively uniformly along with the small fluid droplets. FLIP produces spray that seems to stick together because the fluid does this too and spray is only created by fluid cells. On the other hand FLIP excels at foam structures due to the better preserved small surface waves that are quickly removed by the quite chaotic movement of SPH particles. For air bubbles both methods look similar, but can definitely not compete with the highly realistic, two-way coupled bubble simulations presented in Chapter 2.

Swirl Scene. Here, a thin layer of water covers the entire floor of the simulation domain. Four cuboids of water on top of it and four walls reaching through it are radially symmetric arranged to form a large whirlpool in the center. We only simulated this scene with FLIP but employed two different secondary simulations, one with few and one with much diffuse material to show how the number of particles influences the resulting images for the same body of water.

The second row in Figure 8 features the frames from the pure FLIP simulation in the third row, enhanced with an average of nearly one million secondary particles per frame. Spray is generated at wave crests, but appears relatively thin in some places of the first two images. Most of the larger surface waves are covered with foam on the right hand side but tiny details are not visible. In the first row about 3.5 times more diffuse material is present on average. This means spray looks more smooth and even the smallest surface details receive some number of foam particles. The comparison shows that large numbers of secondary particles do look better, but even few particles are sufficient to create a much more realistic impression than water alone.

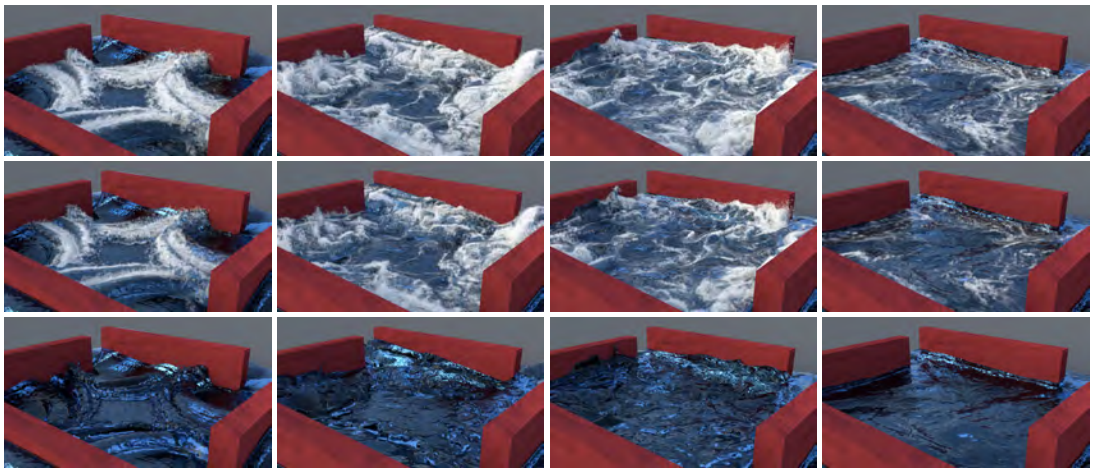

Figure 8: Swirl Scene. Comparison of different amounts of diffuse material: The top row uses 3.5 times more diffuse materials than the middle row, and the bottom row shows the basic FLIP simulation. The frame numbers are 45, 145, 245, and 499 (left to right).

5.2 Performance

Figure 9 displays the timings for the underlying fluid simulations without any enhancement. On the left, the SPH simulations take about one to three minutes per frame resulting in an average time of 1.25 minutes for the Cube Scene and 1.81 minutes for the Stairs Scene. For the Cube Scene we had to reduce the time step to 0.0008 seconds for all frames smaller than 20 and between 145 and 175 manually to avoid well-known artifacts from the IISPH (see [Ihm+14a]) that could not be handled by the adaptive time stepping. Otherwise, the simulation would become unstable due to repeated compression and decompression inside the fluid. As a result the orange plot shows

significant spikes at the respective frames.

The basic FLIP simulations, drawn on the left hand side of Figure 9, are one order of magnitude faster with average simulation times per frame of 3.14 seconds for the Cube Scene, 9.24 seconds for the Stairs Scene and 9.83 seconds for the Swirl Scene. The critical factor for the simulation times for SPH is the number of particles while for FLIP the grid size and the number of fluid cells, and therefore the number of particles, are important.

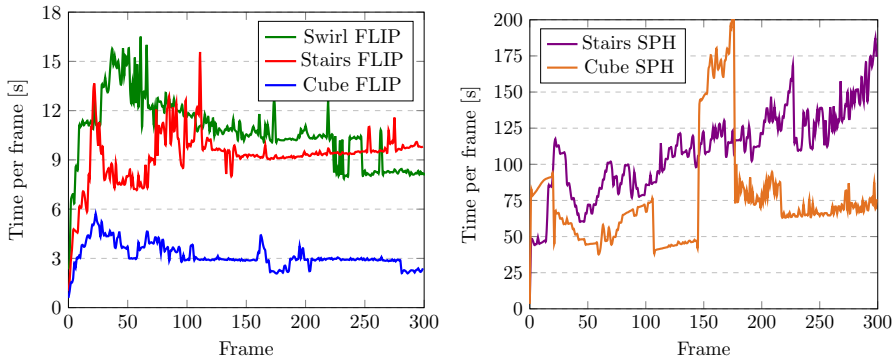

Figure 9: Detailed performance of the underlying fluid simulations from Table 1. The time for the SPH scenes increases by a factor of roughly ten with respect to the FLIP scenes. The remaining frames 300–500 from the Swirl Scene are not displayed.

Figure 10 shows the number of secondary particles for every simulation in black. The general shapes of the plots for the scenes in each row are closely related although they are different in height. The upper colored line in each diagram represents the simulation time of the corresponding secondary particle algorithm. It always follows the black curve as more particles obviously need more time to compute. The dotted, colored line below shows what remains when subtracting the time used for the simulations bottleneck that scales most with the number of secondary particles as outlined in the next paragraphs. This should be the starting point of future performance improvements.

For the FLIP method two significant bottlenecks exist, each taking roughly the same amount of time: the advection step and outputting the secondary particles. The latter is very specific and depends on the operating system, the fluid framework and the required input for the rendering software, so improvements for this are not particularly useful. In our implementation the advection step is relatively inefficient and can most likely be optimized, for instance by avoiding to recompute the entire neighbor cube or by reducing the number of cache misses with dedicated data structures, but we leave the details to future work. Apart from some minor variations the resulting plots for

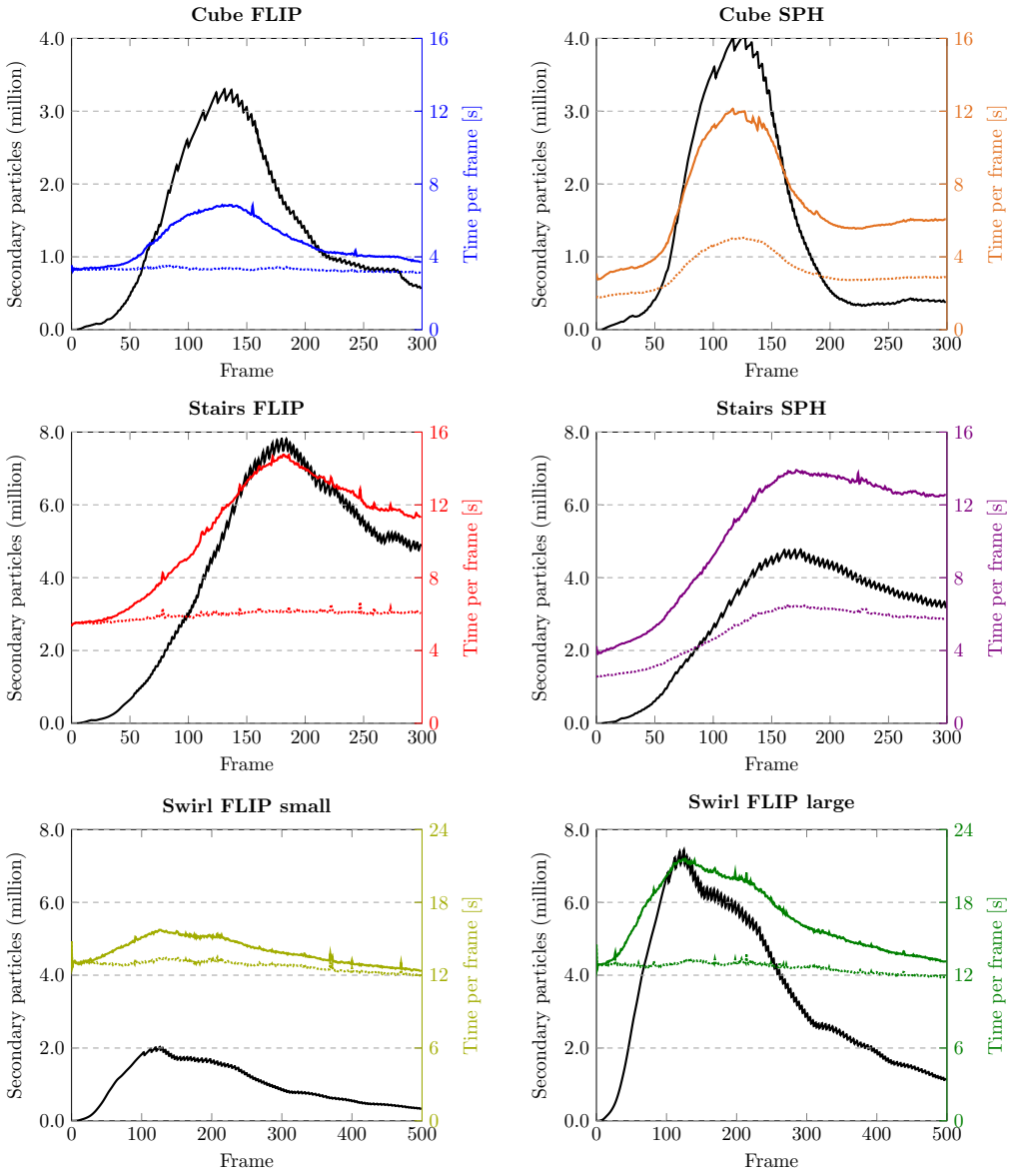

Figure 10: Performance of both secondary particle algorithms. For each scene the number of diffuse particles is drawn in black with the corresponding scale on the left. The colored plots belong to the scale on the right and show the entire simulation time for every frame in each scene. The dotted, colored plot below indicates the base time for each frame i.e., without the time-critical step that scales most with the number of secondary particles.

FLIP are entirely flat and only depend on the number of fluid cells from the underlying simulation, shown clearly in the last row of Figure 10. This nearly constant function allows quick estimates for the lower limit of the entire duration of the secondary simulation with only few simulated frames.

For the SPH method the bottleneck is the computation of the fluid neighbors for each diffuse particle, that are used for advection as described in Section 4.4. There are already methods to compute this efficiently; Ihmsen et al. [Ihm+12] suggest to use compact hashing. Our implementation only uses a naive, inefficient approach here because implementing an acceleration structure from scratch is beyond the scope of this thesis. The resulting plots for SPH shows some curvature because there are still steps that scale with the number of secondary particles, but these are minor compared to the neighbor computation.

In relative terms, for our algorithm the ratio of one fluid time step to one secondary time step is not optimal and the FLIP method takes similar time as the fluid simulation itself. The reason for this behavior is the upscaled domain that increases the number of fluid cells that have to be computed and when using an upscale factor of one, a similar ratio to the SPH method could be achieved. But when comparing the absolute timings it can be seen that our FLIP adaptation is capable of simulating secondary particles in similar numbers even faster than the SPH method. However, timings remain in the same order of magnitude for both implementations: For the Cube Scene the average number of secondary particles is about 1.4 million in both cases and the FLIP method is 30% faster than the SPH method. Both simulations take nearly 53 minutes for the Stairs Scene, but our method handles 46% more diffuse material than the SPH method on average.

5.3 Influence of User-Defined Constants

Our model contains many user-defined constants that control the simulations result. Fine-tuning them for an optimal render outcome can be tedious and time consuming. Therefore, we want to show the influence of the most important constants and provide values that can be used as a starting point for users or future research.

Table 2: Set of constants as basis for the influence of user-defined constants.

τ_k^{min}	τ_k^{max}	τ_{ta}^{min}	τ_{ta}^{max}	τ_{wc}^{min}	τ_{wc}^{max}	k_{wc}	k_{ta}	c_s	c_b	k_b	k_d	l_{min}	l_{max}
5	50	5	20	2	8	1000	200	0.3	0.77	1.0	0.6	0.7	1.6

We always used the values from Table 2 for the following simulations and only altered few parameters to get comparable results. We kept the settings throughout rendering consistent as well. When sometimes different settings were necessary for illustration purposes it is mentioned in the respective section. The uniform rendering also means that images from the following sections that look relatively unrealistic could create a completely different visual impression when appropriate rendering is applied. For instance, simulations that contain very few secondary particles would look better with a higher volume radius for rendering as a compensation.

5.3.1 Potential Thresholds τ^{min} and τ^{max}

The values for the potential thresholds, used in Equation (4), map the generation criteria from an arbitrary range to the potential range $[0, 1]$ and therefore they influence how many particles are generated where inside the grid. We tried different value combinations for all thresholds and as long as the generation criterion of all cells is reasonably distributed over the interval $[\tau^{min}, \tau^{max}]$, nearly no visual difference could be noticed. Only when choosing the maximum threshold clearly too low or the minimum threshold clearly too high, nearly all cells end up with potential values of zero and one respectively and thus influence the simulation result negatively. We also advise against using very low values for the minimum threshold in combination with a small interval because this can cause every cell to create particles at every time step and lead to exponentially growing storage consumption.

The potential thresholds from Table 2 gave good results for all scene setups we tested. Ihmsen et al. [Ihm+12] proposed to use the same values for the SPH method too. In addition, according to Equation (14) the number of generated particles is heavily influenced by the choice of the sampling constants, thus we suggest to set potential thresholds in a suitable order of magnitude and solely control the number of particles by k_{wc} and k_{ta}.

5.3.2 Sampling Constants k_{wc} and k_{ta}

These constants determine how much diffuse material is created at wave crests and at regions where air is trapped according to Equation (14). Therefore, they influence the amount of secondary detail visible in the scene. For all the simulations in this section the corresponding other sampling constant is set to zero to illustrate the effects of the currently discussed constant more clearly.

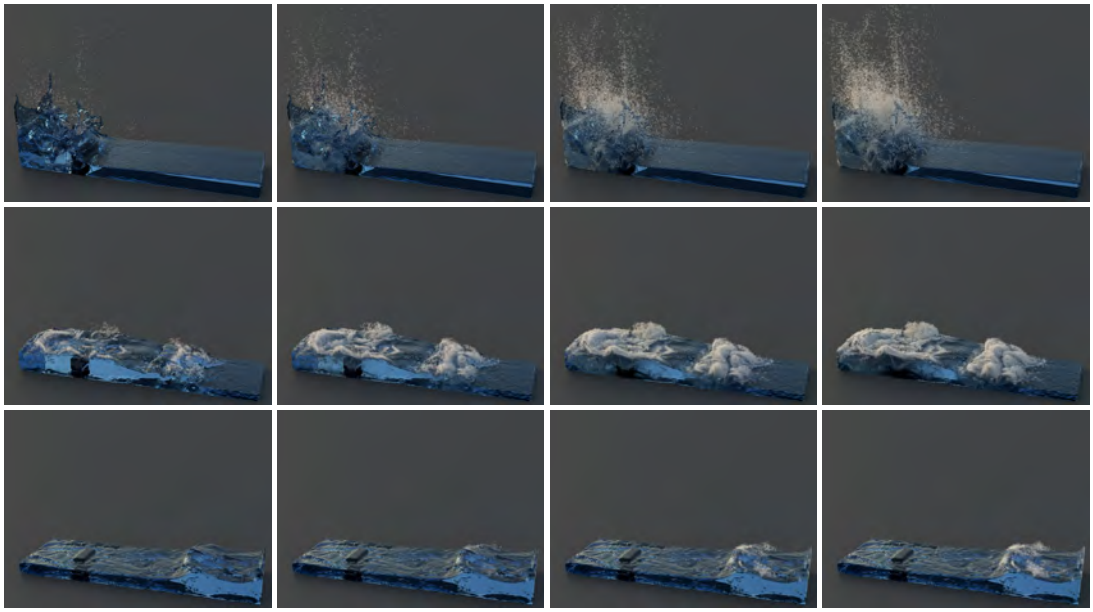

Figure 11: The four values 50, 100, 200 and 300 for k_{ta} displayed in the columns (left to right) while k_{wc} is set to zero in all images. The frames 70, 135 and 210 can be seen in every row (top to bottom).

Figure 11 shows the results from simulations where k_{wc} is reduced to zero for illustrating purposes and k_{ta} is altered. As described in Section 4.2.2 trapped air primarily appears at large impacts and where waves hit the water surface. These effects are apparent in the first two rows as they show significantly different amounts of air bubbles in each column shortly after the impact in frame 70 and near the obstacle in frame 135. There is some influence in the spray creation due to the advection that drags particles from the water in the air, but compared to the influence of k_{wc} in Figure 12 the amount of spray here is rather irrelevant. After the water becomes calmer in the bottom row, nearly no air is trapped and thus the amount of diffuse material is small. On the one hand the wave in the middle row hits the surface with a relatively high speed and much spray and foam is created there. On the other hand the wave in the

lowermost row has few spray particles for all values of k_{ta}, because it is slow and only slightly breaks. Note, that both waves only have few particles at their lip in contrast to the two lower rows of Figure 12.

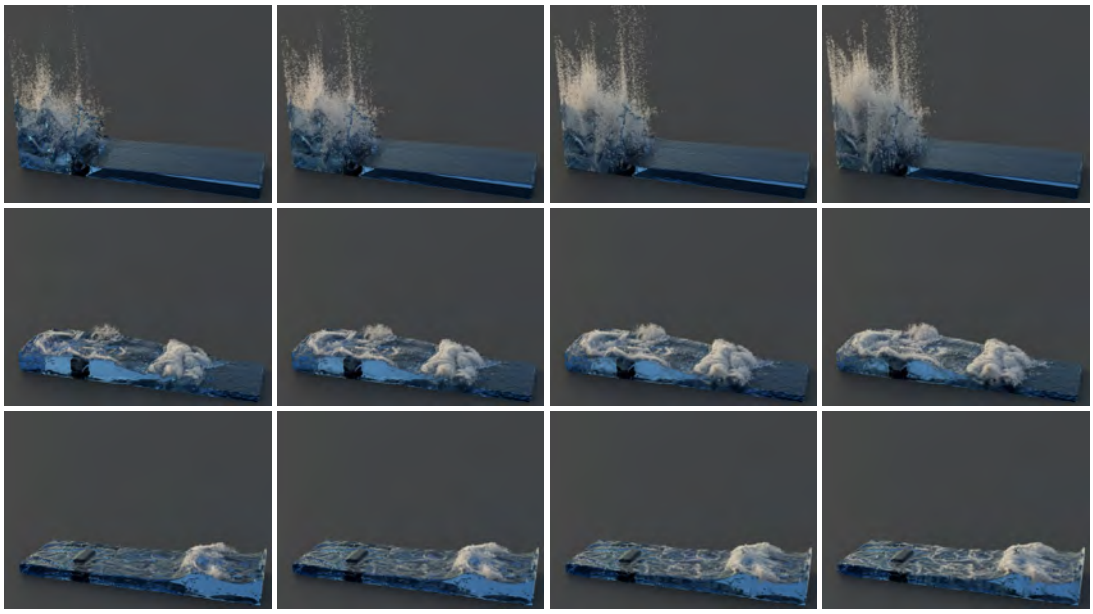

Figure 12: Again the frames 70, 135 and 210 can be seen in the different rows (top to bottom). This time k_{ta} is set to zero and the simulations used 125, 250, 500 and 1000 for k_{wc} (left to right).

The wave crest constant k_{wc} determines the number of secondary particles near the water surface on the lip of each wave. Figure 12 displays in its topmost row that higher values lead to generally more spray. Only if the region is very turbulent and no clear surface is recognizable bubbles may be created, visible in the two images on the right. Therefore, the next row contains nearly no bubbles. The last frame indicates the influence of k_{wc} on foam very clearly: on the left nearly no details on the surface can be observed and going to the right the level of detail incrementally increases. In the rightmost image even smallest waves on the surface are enhanced and create the very distinct impression of dissolving foam patterns. You can notice that in this row some small bubble structures appear. These are caused by errors in the normal computation or can happen due to regions that should be fluid cells, but were missed in the gap filling process in Section 4.1 as it is not completely accurate.

Figure 13 shows how the different sampling constants influence the number of secondary particles over the entire simulation time. Both diagrams show a similar peak,

but it is clearly visible that k_{ta} only emphasizes impacts while k_{wc} always enhances waves with some amount of diffuse material. It is also interesting how doubling k_{wc} always results in roughly twice as many particles overall, while the particle number grows a lot faster when doubling k_{ta}. The reason for this behavior is the higher number of fluid cells with high trapped air potential, than the number of cells with high wave crest potential. This means to adjust the number of particles in the scene generally, starting by altering k_{ta} is suggestive as it provides most of the particles.

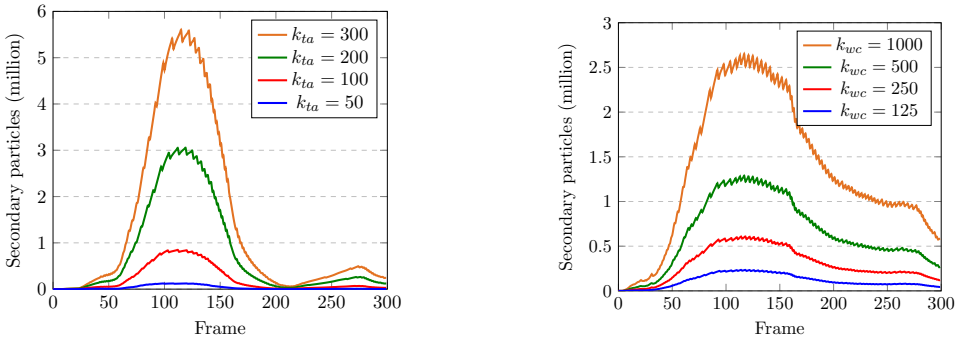

Figure 13: Amount of diffuse material for different sampling constants. On the left trapped air potentials from Figure 11 are used and on the right wave crest potentials according to Figure 12 are employed.

5.3.3 Classification Thresholds c_s and c_b

The classification thresholds on the neighbor ratio determine whether secondary particles are rated as spray, foam or air bubbles following Equation (19) and control the transition among the different groups. In addition, they indirectly control how the diffuse material moves. To illustrate this Figure 14 contains all combinations of two values for each classification threshold over three frames. The two topmost images of the left two columns with a spray classification of 0.15 show that particles remain in a foam state longer and the gravity from Equation (21) for spray advection takes effect relatively late. The spray particles move far away from the water and although the main impact on the wall is already over in frame 85, still a lot of spray is in the air. On the right, where c_s is increased by 0.3, spray particles move only near the water and shortly after the impact almost all spray is gone. Note, that the small spray filaments in the air only appear due to the way the water moves. The appearance of filaments in the water is a well-known problem for FLIP simulations. The spray filaments are closely related to this issue and would disappear if this artifact of FLIP simulations can be prevented in any way.

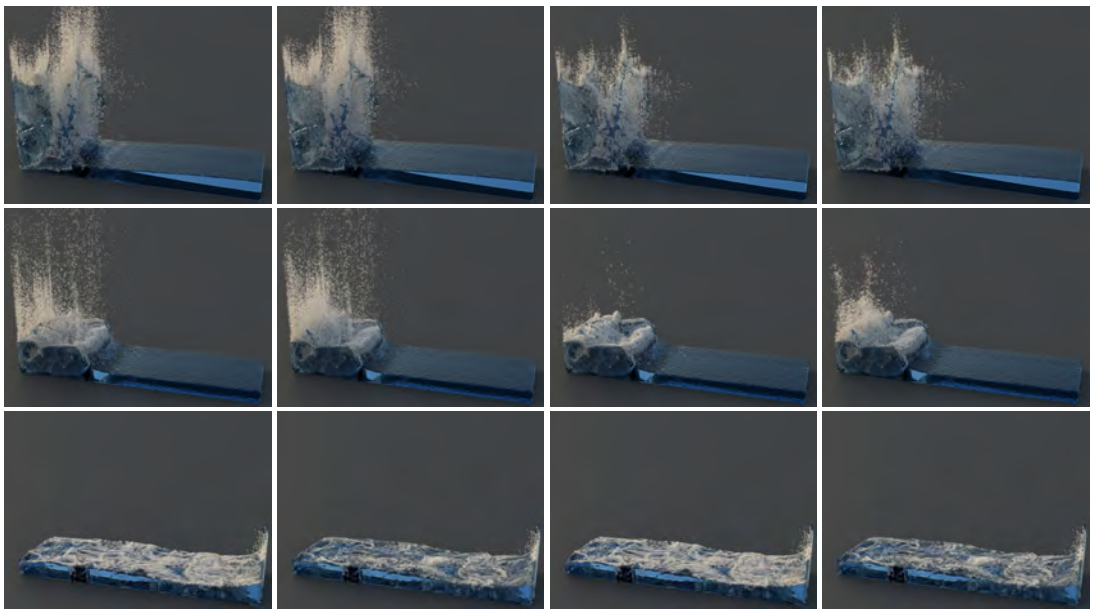

Figure 14: The four possible combinations of $c_s \in [0.15, 0.45]$ and $c_b \in [0.6, 0.9]$ in each column. The order of the (c_s, c_b) pairs is (0.15, 0.6), (0.15, 0.9), (0.45, 0.6) and (0.45, 0.9) from left to right. Every row consists of the frames 60, 85 and 170 (top to bottom).

In the lowest row of Figure 14 the influence of the bubble threshold c_b is more evident. On the one hand, low values like in the first and third column force all bubbles to rise to the very surface and appear as white foam in the images. High values on the other hand mean that some air bubbles are classified as foam already slightly below the water surface, for instance seen in column two and four. As a result they stop to rise there and only move along with the water. Thus, they contribute less to the amount of foam displayed on top of the surface.

5.3.4 Buoyancy and Drag Constants k_b and k_d

The bubble movement in our model from Equation (24) is affected by the buoyancy and drag constants. The buoyancy constant k_b controls how fast air bubbles rise in the water, measured as inverse directed and scaled gravitational force. Figure 15 illustrates the development of simulations with different buoyancy constants over 40 frames in each row. The initial image on the far left always shows a similar bubble state, because the impact of the water on the left boundary shortly before frame 120 created them. Clearly, in the two top rows with values of 0.25 and 0.5 for k_b bubbles rise very slowly

and even at frame 160 the obstacle is nearly entirely surrounded by air bubbles. In contrast, the two bottom rows show quicker rising bubbles and the water next to the obstacle is nearly free from particles in frame 160. This is caused by higher buoyancy values of 1.0 and 2.0. It might seem like higher buoyancy leads to faster dissolution as the bottom right corner shows a lot less diffuse material than the top right corner. That is not the case, because we altered the way the scene was rendered here: to make the bubble movement more apparent, a transparent shader was applied to the water and foam and spray particles were entirely excluded from rendering. That means in the bottom right corner most bubbles are already integrated in the foam and thus are not visible in the image, but still dissolve at the same speed.

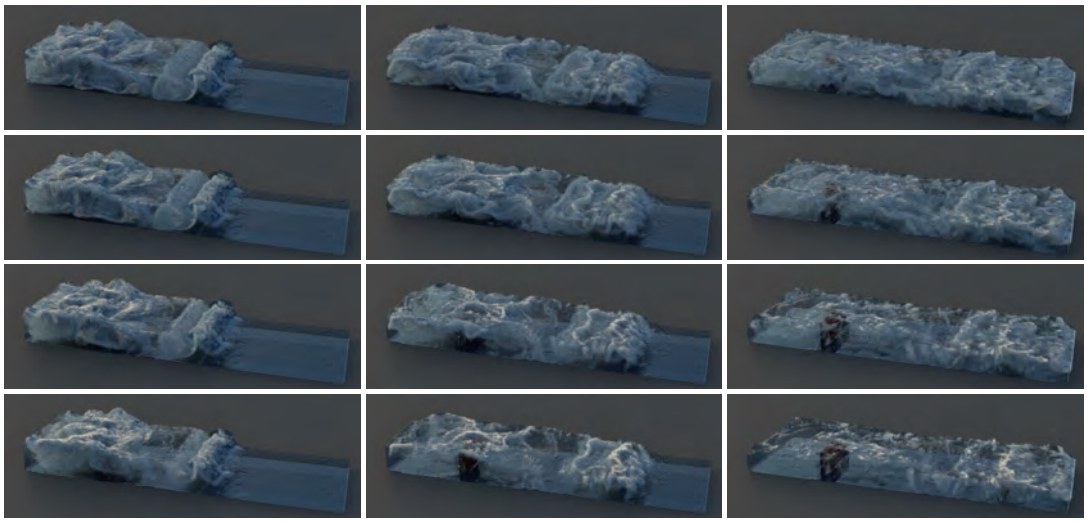

Figure 15: A comparison of different buoyancy values for three different frames. The rows display simulations with buoyancy values of 0.25, 0.5, 1.0 and 2.0 (top to bottom). The columns show the frames 120, 140 and 160 for each row (left to right). Spray and foam particles were excluded from rendering and an additional transparent shader is applied to the water.

The bubble movement along with the surrounding water is determined by the drag constant k_d from Equation (24) as well. Small values like 0.2 from the top row of Figure 16 produce bubble movement mainly determined by the buoyancy. This means bubbles rise nearly straight from where they were created and build a relatively thick, uniform foam on the water surface as visible in the first two frames of this row. Small foam details, like those seen in the third frame, can tower above the surface to a certain degree as particles oscillate between a foam and bubble state and thereby slightly move out of the water. In the second row a drag value of one was used, moving the bubbles

mainly with the fluid flow, similar to the foam advection from our model. Choosing k_d greater than one means bubbles move faster than the water and overtake it which generally leads to more spray particles near waves. This can be observed in the first frame of the bottom row in Figure 16. It can happen that some spray particles move in the opposite direction than the main waves movement and bubbles can be dragged back downwards. These effects are apparent in the next frame: there are more bubbles near the obstacle than above and near the back of the wave spray starts to move to the left.

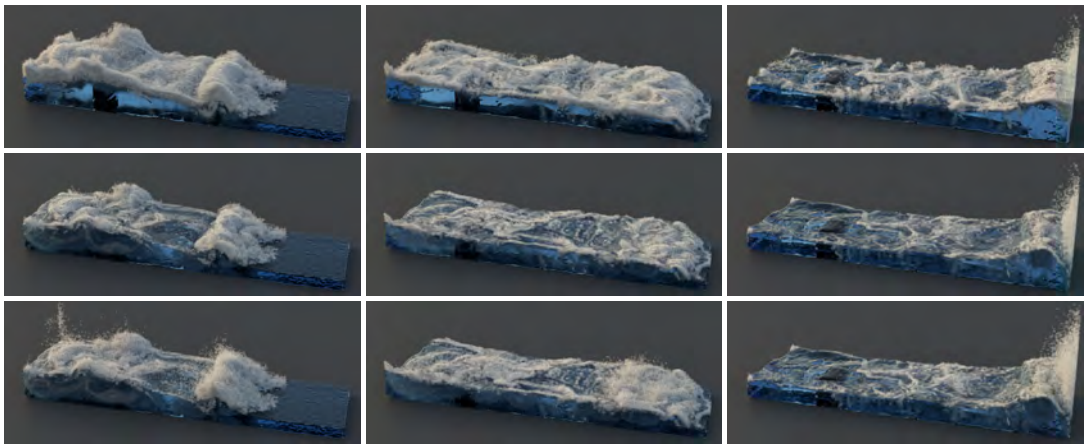

Figure 16: Selection of the three drag values 0.2, 1.0 and 1.4, one for every row (top to bottom). The columns represent the frames 130, 155 and 190 for each row (left to right).

5.3.5 Lifetime Limits l_{min} and l_{max}

Both lifetime limits control how long secondary particles stay in the simulation domain before they get removed. As seen in Equation (26), l_{min} and l_{max} create an interval for the initial lifetime of every particle. The topmost row in Figure 17 shows that very small lifetime values can cause spray particles to dissolve in mid-air in the top left corner, but for high enough values all simulations look similar in the columns further on the right. The differences become more obvious in the two later frames. Raising l_{min} and l_{max} increases the duration small foam details are preserved. In the first two columns foam dissolves quickly behind waves, while larger values for l_{min} and l_{max} further on the right drag detailed foam filaments along on the water surface for some time before dissolving them. This effect is especially strong near the obstacle that swirls the water surface.

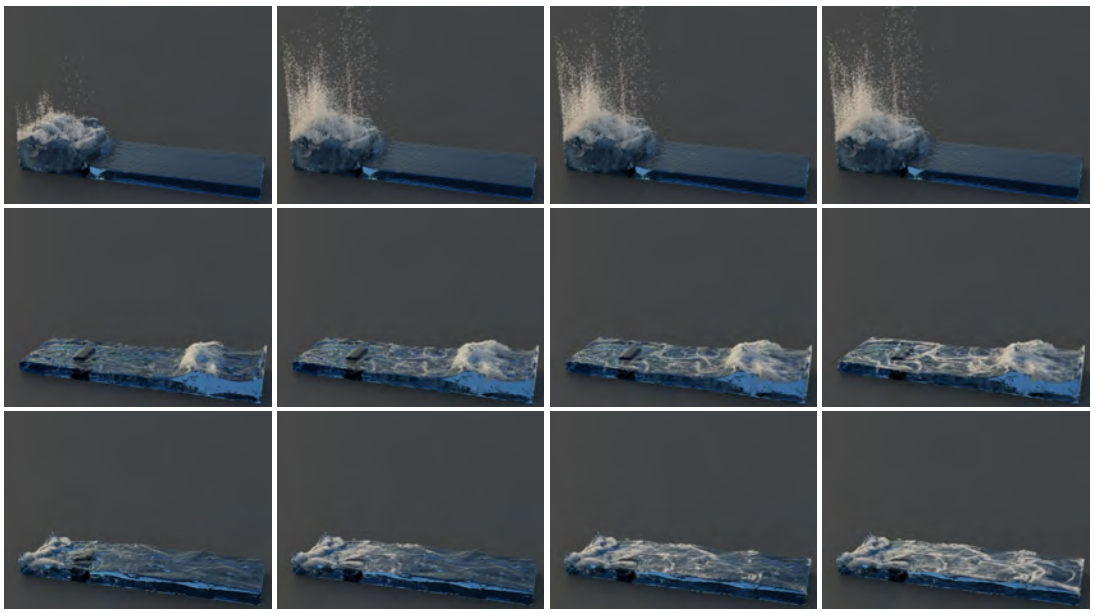

Figure 17: Various combinations of lifetime limits. The rows represent the frames 80, 210 and 280 (top to bottom). From left to right the (l_{min}, l_{max}) pairs used for the simulations were (0.2, 0.7), (0.7, 1.6), (1.6, 2.1) and (1.6, 4.0).

Obviously, longer lifetime increases the overall number of diffuse particles in the scene. The diagram in Figure 18 contains the number of secondary particles corresponding to the simulations from Figure 17. The three plots in blue, red and green use the same interval size of 0.5 and thus they show a closely related general shape with a relatively spiky peak of different height around frame 120. Larger ranges flatten the curve and diffuse material dissolves more slowly, displayed in orange. To emphasize the influence of the lifetime limits and reduce the number of particles for the simulation with the large lifetime range, we used smaller sampling constants in all simulations by dividing the numbers for k_{wc} and k_{ta} from Table 2 in half.

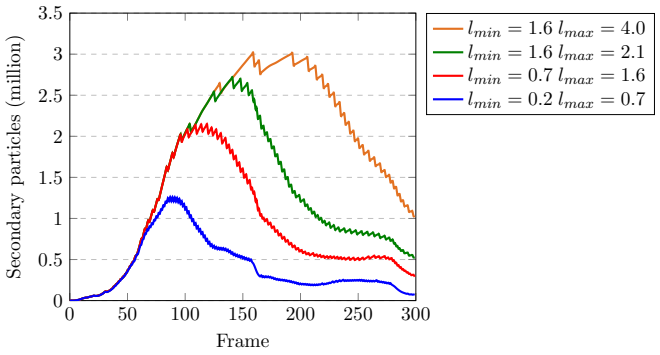

Figure 18: Number of particles for the different lifetime limits from Figure 17.

6 Conclusion

In this thesis we showed how to simulate visually convincing spray, foam and air bubble effects for FLIP and other grid-based fluids using secondary particles.

First, we gave an overview over the different possibilities to enhance fluid simulations and showed how spray, foam and bubble effects are created in related research. The basic concepts of SPH and FLIP, two very common methods for fluid simulations, were shortly outlined and further literature was mentioned.

We explained in detail how we altered the approach presented by Ihmsen et al. [Ihm+12] for purely grid-based and FLIP fluids. Therefore, potentials were computed according to observable effects in real fluid flow that determined where and how many particles are initialized. To recreate the volumetric impression of foam and spray we presented an adapted sampling technique to avoid regular patterns. The secondary particles were moved in the grid along with the fluid flow or due to gravity depending on the particles classification using the new concept of a neighbor ratio. Particles are dissolved in a way to preserve small foam patterns on the surface for high realism and we compared different boundary handling schemes with their resulting artifacts. Some insight in our implementation was provided and we discussed how secondary particles can be displayed by volume rendering.

Next, we compared the visual impression of the SPH method to our grid adaptation and discussed the performance of both approaches. Shortly summarized, the SPH method excels in the depiction of realistic spray while FLIP produces better looking small surface waves and foam details. Finally, the influence of the various user-defined constants of our model was displayed using several frames from our simulation videos. This can help users or future researchers to find appropriate starting values for realistic simulations faster.

7 Future Work

Secondary enhancement methods for fluid simulations in general are still relatively unexplored and will be topic of future research as it delivers appealing simulations with low computational cost. Our secondary particle model for FLIP can still be improved as well. We found four different directions as a starting points for future work.

Optimization of the underlying simulation. Both approaches, our adaptation and the one from Ihmsen et al. [Ihm+12], are highly dependent on the quality of the underlying fluid simulation. The reason is the advection of the diffuse material along with the water, where artifacts from the fluid create unrealistic initialization or movement. For instance, some pictures in Chapter 5 show spray filaments due to artifacts from the FLIP simulation. There are already some approaches to get rid of them and increase the overall fidelity of FLIP simulations, for example the work from Um et al. [UHT17]. They propose to use neural networks trained with data from physically accurate simulations to generate small splashes that basic FLIP could not create. In addition, some of the combined methods mentioned in Chapter 2 could also be enhanced by merging secondary particles for SPH and for grids in one large framework. This seems quite promising, because the weaknesses of both secondary particle models could be removed as each method would only be employed where it excels.

Direct improvements for our model. The general boundary handling of our model is not optimal as discussed in Section 4.6 and could be improved. Slopes that are already challenging for the underlying fluid simulation cause even more problems for secondary particle creation and movement. They are typically represented as a stair shape with small steps, due to the discretization on the grid. This leads to regular foam patterns as many small wave crests are recognized that also create heavy aliasing effects in the rendering process. Overall, in the current state of our solver implementation slopes create highly implausible results. We did not test our suboptimal boundary handling schemes for moving objects but considering the results for static obstacles and slopes we assume that this requires additional work as well. Seeding bubbles with respect to the surface roughness of obstacles as done by Patkar et al. [Pat+13] could be employed too. Approaches with interactions among secondary particles would reduce

the performance significantly, but could be worth exploring. Heuristic methods would be a better solution to increase the realism, for example the dispersed bubble flow from Kim et al. [KSK10]. More effects could be added too, for instance mist as a finer version of spray in the air similar as proposed by Yang et al. [Yan+14].

Performance. As seen in Section 5.2 our new method already results in reasonably fast simulations and some potential performance optimizations were already mentioned there. In addition, it may be possible to advance it even further to work for real-time applications, like the one from Chentanez and Müller [CM11]. This would require heavily reduced resolutions and particle numbers, but would be a great addition to more convincing real-time simulations. For hybrid methods of shallow water simulations, like proposed by Thuerey et al. [Thu+07a], an entirely new approach for foam and spray with secondary particles will be necessary due to a very different fluid representation.

Rendering efficiency. Last and most important, the main bottleneck of our simulations is rendering. With three density textures of size 512^3 for the volume rendering the 16 GB of RAM from the used machine were stretched to the limit. For higher image resolutions this means a single voxel results in several pixels in the rendered picture and thus the voxel structure becomes visible from a close distance, even when the single pixels cannot be recognized yet. In addition the textures only support volume radii up to a certain minimum and therefore restrict the number of secondary particles that can be displayed. Ray tracing the textures that are applied to several nested volumes is very time-consuming and Blender does not fully support GPUs for this yet. This could already result in some performance improvements due to higher parallelization. Future research needs to decrease time and memory consumption of the rendering process or develop new techniques for accurate volume rendering. This aspect could be a problem if the main secondary particle algorithm will be adapted for real-time applications: the rendering remains very expensive even for relatively few secondary particles and needs to be addressed before real-time foam and spray effects can be simulated convincingly. Additionally, rendering particles with adaptive radii and color could further improve the visual realism.

Bibliography

[Aki+13] N. Akinci, A. Dippel, G. Akinci, and M. Teschner. "Screen Space Foam Rendering." In: *Journal of WSCG* 21.3 (June 2013), pp. 173–182. ISSN: 1213-6972.

[BM07] R. Bridson and M. Müller-Fischer. "Fluid Simulation: SIGGRAPH 2007 Course notes." In: *ACM SIGGRAPH 2007 Courses*. SIGGRAPH '07. San Diego, California: ACM, 2007, pp. 1–81. ISBN: 978-1-4503-1823-5. DOI: 10.1145/1281500.1281681.

[Bri16] R. Bridson. *Fluid simulation for computer graphics*. Second edition. Boca Raton ; London ; New York: CRC Press, 2016, XI, 263 Pages. ISBN: 978-1-4822-3283-7.

[BUH15] S. Baek, K. Um, and J. Han. "Muddy water animation with different details." In: *Computer Animation and Virtual Worlds* 26.3-4 (2015), pp. 347–355. ISSN: 1546-427X. DOI: 10.1002/cav.1646.

[Cle+07] P. W. Cleary, S. H. Pyo, M. Prakash, and B. K. Koo. "Bubbling and Frothing Liquids." In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. San Diego, California: ACM, 2007. DOI: 10.1145/1275808.1276499.

[CM11] N. Chentanez and M. Müller. "Real-time Eulerian Water Simulation Using a Restricted Tall Cell Grid." In: *ACM SIGGRAPH 2011 Papers*. SIGGRAPH '11. Vancouver, British Columbia, Canada: ACM, 2011, 82:1–82:10. ISBN: 978-1-4503-0943-1. DOI: 10.1145/1964921.1964977.

[Cor+14] J. Cornelis, M. Ihmsen, A. Peer, and M. Teschner. "IISPH-FLIP for incompressible fluids." In: *Computer Graphics Forum* 33.2 (2014), pp. 255–262. ISSN: 1467-8659. DOI: 10.1111/cgf.12324.

[EMF02] D. Enright, S. Marschner, and R. Fedkiw. "Animation and Rendering of Complex Water Surfaces." In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. San Antonio, Texas: ACM, 2002, pp. 736–744. ISBN: 1-58113-521-1. DOI: 10.1145/566570.566645.

[FGP07] E. Froemling, T. Goktekin, and D. Peachey. "Simulating Whitewater Rapids in Ratatouille." In: *ACM SIGGRAPH 2007 Sketches*. SIGGRAPH '07. San Diego, California: ACM, 2007. ISBN: 978-1-4503-4726-6. DOI: 10.1145/1278780.1278862.

[Gei+06] W. Geiger, M. Leo, N. Rasmussen, F. Losasso, and R. Fedkiw. "So Real It'll Make You Wet." In: *ACM SIGGRAPH 2006 Sketches*. SIGGRAPH '06. Boston, Massachusetts: ACM, 2006. ISBN: 1-59593-364-6. DOI: 10.1145/1179849.1179874.

[GH04] S. T. Greenwood and D. H. House. "Better with Bubbles: Enhancing the Visual Realism of Simulated Fluid." In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '04. Grenoble, France: Eurographics Association, 2004, pp. 287–296. ISBN: 3-905673-14-2. DOI: 10.1145/1028523.1028562.

[HK03] J.-M. Hong and C.-H. Kim. "Animation of Bubbles in Liquid." In: *Computer Graphics Forum* 22.3 (2003), pp. 253–262. ISSN: 1467-8659. DOI: 10.1111/1467-8659.00672.

[Hon+08] J.-M. Hong, H.-Y. Lee, J.-C. Yoon, and C.-H. Kim. "Bubbles Alive." In: *ACM SIGGRAPH 2008 Papers*. SIGGRAPH '08. Los Angeles, California: ACM, 2008, 48:1–48:4. ISBN: 978-1-4503-0112-1. DOI: 10.1145/1399504.1360647.

[Ihm+10] M. Ihmsen, N. Akinci, M. Gissler, and M. Teschner. "Boundary Handling and Adaptive Time-Stepping for PCISPH." In: *Proc. VRIPHYS*. Copenhagen, Denmark, Nov. 11, 2010, pp. 79–88.

[Ihm+11] M. Ihmsen, J. Bader, G. Akinci, and M. Teschner. "Animation of Air Bubbles with SPH." In: *Proceedings of the International Conference on Computer Graphics Theory and Applications - Volume 1: GRAPP, (VISIGRAPP 2011)*. INSTICC. SciTePress, 2011, pp. 225–234. ISBN: 978-989-8425-45-4. DOI: 10.5220/0003322902250234.

[Ihm+12] M. Ihmsen, N. Akinci, G. Akinci, and M. Teschner. "Unified Spray, Foam and Air Bubbles for Particle-based Fluids." In: *Vis. Comput.* 28.6-8 (June 2012), pp. 669–677. ISSN: 0178-2789. DOI: 10.1007/s00371-012-0697-9.

[Ihm+14a] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner. "Implicit Incompressible SPH." In: *IEEE Transactions on Visualization and Computer Graphics* 20.3 (Mar. 2014), pp. 426–435. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.105.

[Ihm+14b] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner. "SPH Fluids in Computer Graphics." In: *Eurographics 2014 - State of the Art Reports, Strasbourg, France, April 7-11, 2014*. 2014, pp. 21–42.

[Ihm13] M. Ihmsen. "Particle-Based Simulation of Large Bodies of Water with Bubbles, Spray and Foam." 2013.

[JM12] W. Jakob and S. Marschner. "Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport." In: *ACM Trans. Graph.* 31.4 (July 2012), 58:1–58:13. ISSN: 0730-0301. DOI: 10.1145/2185520.2185554.

[Kim+06] J. Kim, D. Cha, B. Chang, B. Koo, and I. Ihm. "Practical Animation of Turbulent Splashing Water." In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '06. Vienna, Austria: Eurographics Association, 2006, pp. 335–344. ISBN: 3-905673-34-7.

[KSK10] D. Kim, O.-y. Song, and H.-S. Ko. "A Practical Simulation of Dispersed Bubble Flow." In: *ACM SIGGRAPH 2010 Papers*. SIGGRAPH '10. Los Angeles, California: ACM, 2010, 70:1–70:5. ISBN: 978-1-4503-0210-4. DOI: 10.1145/1833349.1778807.

[KVG02] H. Kück, C. Vogelgsang, and G. Greiner. "Simulation and Rendering of Liquid Foams." In: *Proceedings - Graphics Interface*. June 2002.

[Los+08] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. "Two-Way Coupled SPH and Particle Level Set Fluid Simulation." In: *IEEE Transactions on Visualization and Computer Graphics* 14.4 (July 2008), pp. 797–804. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.37.

[MCG03] M. Müller, D. Charypar, and M. Gross. "Particle-based Fluid Simulation for Interactive Applications." In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '03. San Diego, California: Eurographics Association, 2003, pp. 154–159. ISBN: 1-58113-659-5.

[MMS09] V. Mihalef, D. Metaxas, and M. Sussman. "Simulation of two-phase flow with sub-scale droplet and bubble effects." In: *Computer Graphics Forum* 28.2 (2009), pp. 229–238. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01362.x.

[Mon92] J. J. Monaghan. "Smoothed Particle Hydrodynamics." In: *Annual Review of Astronomy and Astrophysics* 30.1 (1992), pp. 543–574. DOI: 10.1146/annurev.aa.30.090192.002551.

[NØ13] M. B. Nielsen and O. Østerby. "A Two-continua Approach to Eulerian Simulation of Water Spray." In: *ACM Trans. Graph.* 32.4 (July 2013), 67:1–67:10. ISSN: 0730-0301. DOI: 10.1145/2461912.2461918.

[OF03] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. New York: Springer-Verlag, 2003, XIII, 273 Pages. ISBN: 978-0-387-95482-0. DOI: 10.1007/b98879.

[Pat+13] S. Patkar, M. Aanjaneya, D. Karpman, and R. Fedkiw. "A Hybrid Lagrangian-Eulerian Formulation for Bubble Generation and Dynamics." In: *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '13. Anaheim, California: ACM, 2013, pp. 105–114. ISBN: 978-1-4503-2132-7. DOI: 10.1145/2485895.2485912.

[Tak+03] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Ueki. "Realistic Animation of Fluid with Splash and Foam." In: *Comput. Graph. Forum*. Vol. 22. Sept. 2003, pp. 391–400.

[Thu+07a] N. Thuerey, M. Müller-Fischer, S. Schirm, and M. Gross. "Real-time Breaking Waves for Shallow Water Simulations." In: *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*. Oct. 2007, pp. 39–46. DOI: 10.1109/PG.2007.33.

[Thu+07b] N. Thuerey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Gross. "Real-time Simulations of Bubbles and Foam Within a Shallow Water Framework." In: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '07. San Diego, California: Eurographics Association, 2007, pp. 191–198. ISBN: 978-1-59593-624-0.

[TRS06] N. Thuerey, U. Rüde, and M. Stamminger. "Animation of Open Water Phenomena with Coupled Shallow Water and Free Surface Simulations." In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '06. Vienna, Austria: Eurographics Association, 2006, pp. 157–164. ISBN: 3-905673-34-7.

[UHT17] K. Um, X. Hu, and N. Thuerey. *Liquid Splash Modeling with Neural Networks*. Apr. 2017. arXiv: 1704.04456 [cs.GR].

[Wan+13] C.-b. Wang, Q. Zhang, F.-l. Kong, and H. Qin. "Hybrid particle-grid fluid animation with enhanced details." In: *The Visual Computer* 29.9 (Sept. 2013), pp. 937–947. ISSN: 1432-2315. DOI: 10.1007/s00371-013-0849-6.

[Yan+14] L. Yang, S. Li, A. Hao, and H. Qin. "Hybrid Particle-grid Modeling for Multi-scale Droplet/Spray Simulation." In: *Computer Graphics Forum* 33.7 (2014), pp. 199–208. ISSN: 1467-8659. DOI: 10.1111/cgf.12488.

[Yan+15] L. Yang, S. Li, Q. Xia, H. Qin, and A. Hao. "A Novel Integrated Analysis-and-simulation Approach for Detail Enhancement in FLIP Fluid Interaction." In: *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*. VRST '15. Beijing, China: ACM, 2015, pp. 103–112. ISBN: 978-1-4503-3990-2. DOI: 10.1145/2821592.2821598.

[Yue+15] Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. "Continuum Foam: A Material Point Method for Shear-Dependent Flows." In: *ACM Trans. Graph.* 34.5 (Nov. 2015), 160:1–160:20. ISSN: 0730-0301. DOI: 10.1145/2751541.

[ZB05] Y. Zhu and R. Bridson. "Animating Sand As a Fluid." In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. Los Angeles, California: ACM, 2005, pp. 965–972. DOI: 10.1145/1186822.1073298.

[ZYP06] W. Zheng, J.-H. Yong, and J.-C. Paul. "Simulation of Bubbles." In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '06. Vienna, Austria: Eurographics Association, 2006, pp. 325–333. ISBN: 3-905673-34-7.