Screen Space Rendering Solution for Multiphase SPH Simulation

Caio Brito[*] Voxar Labs CIn - UFPE

Mozart William S. Almeida[†] Voxar Labs CIn - UFPE

André Luiz B. Vieira-e-Silva[‡] Voxar Labs CIn-UFPE

João Marcelo Teixeira[§] Departamento de Eletrônica e Sistemas - UFPE

Veronica Teichrieb[¶] Voxar Labs CIn - UFPE

Abstract

Fluid simulation using meshless methods has increasingly become a robust way to solve mechanics problems that require dealing with large deformations, and has become very popular in many applications such as naval engineering, mechanical engineering, movies and games. One of the main methods is the Smoothed Particle Hydrodynamics (SPH). This work has two main goals: to propose a multiphase SPH formulation by extending the work of Silva et al. [22] and to propose a shader based render solution for this kind of simulation. The proposed SPH method was able to simulate multiphase fluids with up to one million particles and the renderer was able to generate visually plausible results up to 60fps.

Keywords: SPH, Multiphase, Rendering, Screen Space

Index Terms: I.3.7 [Computing Methodologies]: Computer Graphics—Three-dimensional Graphics and Realism, Virtual Reality; I.6.8 [Computing Methodologies]: Simulation and Modeling—Types of Simulation, Animation.

1 Introduction

Some of the fluid dynamics problems in naval engineering and mechanical engineering are intended to be simulated with high numerical accuracy. The classic method for this type of simulation is the Finite Element Method (FEM), which can deal effectively with the vast majority of simulation problems, but becomes inefficient in cases where there are large deformations and with boundary regions [40].

To overcome such challenges, meshfree methods may be used such as the Smoothed Particle Hydrodynamics (SPH) [26] and Moving Particles Semi-implicit (MPS) methods [35]. These techniques can simulate fluids efficiently using a system with a discrete number of particles and solving the Navier-Stokes equation of motion without the need to use a grid, making the method with a high degree of flexibility in cases where the traditional methods become very complex [55].

The SPH method was designed over three decades ago by Lucy [42] and Gingold and Monaghan [26] and focused astrophysics applications. Since its conception, there have been continuous improvements and adaptations of the original method to simulate various kinds of physical phenomena. Due to its flexibility and robustness, the method is able to simulate fluid flow [48], spray [32], air bubbles [3], cloth-fluid coupling [74] and many other fluid behaviors [33].

[*]e-mail: cjsb@cin.ufpe.br
[†]e-mail: mwsa@cin.ufpe.br
[‡]e-mail: albvs@cin.ufpe.br
[§]e-mail: jmxnt@cin.ufpe.br
[¶]e-mail: vt@cin.ufpe.br

Vieira-e-Silva et al. [22] proposed a weakly compressible SPH method that relies only on the XSPH formulation to simulate viscosity and prevent particle penetration problem (boundary condition). This approach led to a small SPH formulation with a relatively high numerical precision, which can be used for interactive applications due to its small number of calculations.

According to Violeau and Rogers [66], a major challenge for particle methods is the modeling of the interaction between solids and fluids (boundary condition). Several solutions have been presented, which may lead to a high degree of numerical precision or only the visual accuracy [40]. Another challenge in the field is how to simulate multiphase flow. The SPH method has been used for multiphase flows with approaches that have major changes in the standard SPH formulation [66], which can be quite difficult to implement, or with an adaptive boundary condition, which is very time-consuming. So, these methods may not be the best options for online applications. Looking for better performance, Solenthaler and Pajarola [59] present a multiphase SPH formulation which does not affect the performance negatively and is easy to implement due to its simple modifications on the standard SPH method.

Given the high flexibility of implementation of this type of method, it has been used in the gaming industry [65], in the film industry [29] and the method has become one of the most popular particle-based methods in the animation industry [70].

In those industries, another big challenge is to render the simulated fluid realistically. To visualize the simulation results it is necessary to rebuild the surface using particles identified as free surface [72]. The rendering of the results can be made using several methods such as direct rendering [73], 3D scalar field [72], volume rendering [24] or a screen space approach [65].

Recently, the literature indicates two main methodologies: the use of a 3D scalar field and the screen space approach. In the first one, each particle of the system is associated with a scalar value, and the surface of the fluid is reconstructed using those values, for instance, using a Marching Cubes algorithm [41]. The second approach renders the particles as spheres or point sprites and applies a smoothing filter to the depth map to create a better final looking surface render.

The 3D scalar field approach presents the challenge of choosing the most appropriate kernel function to determine the density of scalar field of the surface particles. To create a smooth surface, the function is calculated using neighboring particles, which tends to be quite costly, making the rendering method more suitable for offline applications [72].

The screen space option may be more suitable for real-time applications because each particle renders individually, without the need to apply a function on the neighboring particles. Once reconstructed, the surface may have a resemblance to jam. To overcome this characteristic, a smoothing function is applied to the depth map of the scene, which is used to calculate the normal at each point. Finding the best function to create a smooth surface is the main challenge of this type of technique [54]. However, those methods only render a single fluid on their graphical pipeline.

In the search of a multiphase SPH formulation that may be used in interactive applications and a rendering solution which can handle multiphase fluid flows with the balance between performance and visual quality, this work has as main purpose: to propose a multiphase SPH formulation by extending the work of Vieira-e-Silva et al. [22] and validate if the XSPH can be integrated with a multiphase solution. Also, to propose a graphical pipeline for real-time multiphase particle-based simulation based on the work of [65].

So, the main contributions of this work are:

1. An overview of SPH simulation and rendering for particle-based simulation.
2. A simple multiphase SPH model which depends only on the XSPH to handle viscosity and boundary conditions.
3. The SPH implementation can handle up to millions of particles.
4. A rendering solution that can be used in real-time applications with good visual quality.
5. A shader based render that can be integrated into many particle-based fluid simulations.

In the next section, the state of art of SPH simulation and particle-based fluid rendering is presented along with the related work of multiphase flows. Then, both simulation and rendering methods are explained to deal with multiphase simulation. In section 4 the test cases to validate the approaches proposed are presented, and then the visual and performance results are presented in section 5. Finally, in section 6, the conclusions are discussed together with future possibilities and enhancements.

2 State of Art

The SPH method was introduced by Lucy [42], Gingold and Monaghan [26] to model astrophysical phenomena. Since then it has been vastly extended to model fluids [6, 13] and even solids behavior [14], mainly under those aspects which could limit the applications simulated by mesh-based approaches, such as large deformations, for instance.

A straightforward adaptation of the original SPH method is the application of weakly compressible fluids. In this kind of fluid, the pressure can be calculated by an equation of state. The works [38, 39, 57, 62] show comparisons between implementations of weakly compressible and truly incompressible methods, in which are applied techniques such as the one introduced by Commins and Rudman [21].

In the truly incompressible methods, the pressure is calculated by the Poisson equation. This equation can be represented by a sparse linear system. Those kind of methods can generate a more accurate solution but require more computation time as stated in the works of [7, 12, 25, 69].

The main difference between the original (astrophysical) SPH method and the newer particle-based fluid simulations is the inclusion of boundary conditions. There are several ways of containing the fluid inside of a bounded geometry, such as, explicit forces [40], pressure influence [69], density compensation [5] or a geometrical approach [46].

To create fluid with distinct behavior some characteristics can be adapted depending on the problem being simulated, such as viscosity, turbulence, smoothing function, among others. The works of [38, 53, 58, 71] show some existing options concerning viscosity improvements.

In order to explain the effects of the smoothing function on meshless simulations, the works of [8,10,11,40,58,61] discuss the changes in behavior when varying the smoothing functions, evaluating the accuracy and stability of the methods.

Given the meshless characteristics of the simulation, it is possible to create a parallel solution, using cluster technology or general purpose programming for graphics processors (GPGPU) techniques, in order to decrease the time consumption, as shown in the works of [19, 27, 28, 37, 76].

In the VR community, SPH simulation has been used in many applications. Cirio et al. [16] proposed a six Degrees of Freedom (DoF) haptic interaction with fluid simulated using a weakly compressible SPH, providing a force-based feedback and being able to achieve real-time performance (90 fps) for 32,768 particles. This work was extended to deal with melting and freezing phenomena [18] and to interact with deformable and rigid bodies [17].

Another example in the VR community is the work of Pan et al. [50], which used a PhysX built-in SPH-based fluid solver to simulate bleeding effects in a virtual reality-based surgical simulator, being able to achieve a performance of 49 fps with 5,000 particles. Also, Wang and Wang [67] proposed a haptic interaction with fluid using SPH and finite elements method and used to operate a canoe with two paddles rowing in the fluid.

In the SPH literature, many methods have been presented to reconstruct the surface given a set of particles. A possible approach is to use a 3D scalar field; the liquid surface is defined by calculating a kernel function which will define a scalar density field. Then, a Marching Cubes algorithm [41] is used to generate a triangular mesh of the isosurface of this 3D field. The choice of the scalar field formulation is the key for creating a high-quality surface; the simplest choice is to use a blobbies approach, also known as metaballs [9], but this method can create surface dumps, depending on the particle distribution.

Smoother surfaces can be found using a scalar field based on the weighted average of particles close to each other and calculated by an isotropic kernel [1,2,49], which can generate a better visual result in sharp features and edges [72].

A second approach to render the fluid surface is to use an explicit method, frequently applied in an Eulerian context [23]. In the SPH literature, a few works can be found using this approach, such as [52], which changes the surface position using information of the particle simulation. Those methods have a high memory consumption and, to avoid this problem, methods such as point splatting [44], and a ray-isosurface intersection with metaballs can be used [75].

Another solution is to render the fluid particles using screen space, which are more suitable methods for real-time applications. Those methods interpret each particle as a sphere to reconstruct the surface from the depth map, the surface is smoothed, and the thickness of the fluid is used to attenuate the fluid color. Many algorithms can be used to smooth the depth map: a binomial filter [47], a Gaussian filter [65], a curvature flow [4, 65], and a post-smoothing filter [54].

To create a more realistic fluid rendering, the screen space approaches have been used together with ray tracing to simulate reflection and refraction [68, 77].

Multiple approaches have been used to render multiphase fluids. [59] uses a ray tracing algorithm [60] to render the fluid, taking 20s to 40min to render a single frame. [70] renders the fluid using mental ray, which takes 1 second to 1.5 minutes depending on the complexity of the presented scene and [68] renders fluids with different colors by mixing ray tracing with particle splatting, being able to render 800k particles with approximately 23 fps in a 1920 1920 resolution.

2.1 Related Work

The multiphase flow was firstly introduced by Monaghan [43], which provided a general and easily extended SPH method to handle multiphase air simulation. Muller at al. [48] was able to simulate multiple fluids with small density ratios by changing the mass and the rest density. Hu and Adams [30] present a constant-density approach, which corrects the intermediate density errors by adjusting the halftime-step velocity with exact projection, allowing to simulate incompressible flows with high-density ratios by the projection SPH method. Recently, Yang et al. [70] extended the SPH method to cover solid phases, including deformable bodies and granular materials using the concept of volume fraction and established a new way of modeling fluid-solid interaction.

Many works benefited from the introduction of multiphase fluid flow in SPH. One example is the work of Tartakovsky and Meakin [63], where it simulates miscible and immiscible fluid flow. In this work, the authors use a new SPH model for immiscible flow that combines number density based SPH flow equations and interparticle interactions. They also present applications of the miscible flow model to the simulation of pore-scale flow and transport [64].

Another example is the work of Hu and Adams [31], where a multiphase SPH for macroscopic and mesoscopic flow was developed. It handles naturally density discontinuities across phase interfaces. There are also newly formulated viscous terms that allow for a discontinuous viscosity and ensure continuity of velocity and shear stress across the phase interface. The authors also introduced thermal fluctuations in a straightforward way based on this formulation and developed a new algorithm capable of dealing with three or more immiscible phases. Lastly, mesoscopic interface slippage is included based on the apparent slip assumption which ensures continuity at the phase interface. For validation purposes, numerical examples of capillary waves, three-phase interactions, drop deformation in a shear flow, and mesoscopic channel flows are considered.

As most of the work found in the literature are done using 2D simulation, there are not many works on rendering multiphase fluid [15, 51]. But, when it is necessary, most works use a ray tracing as in [59, 70]. This approach results in a high-quality level but in both works the rendering is done offline or with a performance of 1 fps.

3 Simulation and Rendering Methods

In this section, the SPH method, based on the work of Vieira-e-Silva et al. [22], and its modifications to support multiphase fluids, are explained. Then, the multiphase rendering solution that extends the work of Laan [65] is explained.

3.1 SPH Formulation

The Smoothed Particle Hydrodynamics (SPH) is a Lagrangian method created originally to simulate astrophysics problems and lately has been used mainly to simulate hydrodynamics problems solving the Navier-Stokes equation, defined by Eq. (1). This approach can be described in two parts: the kernel approximation and the problem discretization.

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla P + \frac{1}{\rho}\nabla\cdot\boldsymbol{\tau} + Fext \tag{1}$$

The Navier-Stokes equation describes the fluid movement regarding three main components: pressure, viscosity and external forces. The SPH solves the fluid movement by considering the fluid as a weakly compressible system, which is based on the fact that every incompressible fluid is a little compressible, and because of that, the method simulates a quasi-incompressible equation to model the simulation.

Vieira-e-Silva et al. [22] proposed a SPH formulation into a series of steps.

The first step is to calculate the particle density, which is calculated using the continuity equation as in Eq. (2).

$$\frac{d\rho_i}{dt} = \sum_j m_j(\mathbf{u}_i - \mathbf{u}_j)\nabla W_{ij} \tag{2}$$

After calculating the density of the particles in the system, the next step is to solve their pressures, which are calculated by the Tait‘s equation (3), as shown in Vieira-e-Silva et al. [22]:

$$P_i = B((\frac{\rho_i}{\rho_0})^{\gamma} - 1) \tag{3}$$

where k_p and B are the pressure constants, ρ_0 is the rest density of the fluid and γ is a constant that usually has a value of 7.

The pressure force is commonly calculated using Eq. (4). This approach ensures a modular equality between two particles and conserves linear and angular momentum, leading to a more stable simulation, as shown in the work of Monaghan [45]:

$$\frac{1}{\rho_i}\nabla P_i = \sum_j m_j(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2})\nabla W_{ij} \tag{4}$$

To simulate the viscosity and to prevent a particle penetration problem, the XSPH method is used. This method forces particles near each other to move with almost the same velocity. The method is computationally cheaper than other methods and has only one tunable parameter making it easy to change [56]. This parameter controls the viscosity influence on the fluid; the higher the parameter value, the greater the influence of the viscosity of the fluid.

To conserve linear and angular momentum, Eq. (5) was used to calculate an intermediate velocity v^* and Eq. (6) was used to calculate the new velocity.

$$v_i^* = v_i + a_i\triangle t \tag{5}$$

$$v_i = v_i^* + \varepsilon\sum_j m_b\frac{v_i^* - v_j^*}{\rho_j}w_{ij} \tag{6}$$

where a_i is the particles acceleration and ε is the tunable parameter of the XSPH method.

Finally, the final term in the Navier-Stokes governing equation is related to the external forces in the system which in most systems is the gravity. The particle new velocities and positions are calculated using a simple first order Euler time integration by Eq. (7) and Eq. (8), respectively, as suggested by Schechter and Bridson [56]:

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t + \mathbf{a}_i^t\triangle t \tag{7}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{u}_i^{t+1}\triangle t \tag{8}$$

where $\mathbf{a}_i$ is the particle i acceleration and $\triangle t$ is the time step of the simulation.

The SPH method flow can be found in Fig. 1

3.1.1 Handling Multiphase Simulation

To handle multiphase simulation, Solenthaler and Pajarola [59] proposed to calculate the density of a particle by treating its neighbors as if they would have the same rest density and mass as itself. So, to extend the SPH technique explained before, the density should be calculated as Eq. (9).

$$\frac{d\rho_i}{dt} = \sum_j m_i(\mathbf{u}_i - \mathbf{u}_j)\nabla W_{ij} \tag{9}$$

In the method proposed by Vieira-e-Silva et al. [22], the XSPH contribution is calculated for every neighbor of a fluid particle. But, to create a more realistic behavior, only the fluid particle contribution is used to handle multiphase flows.

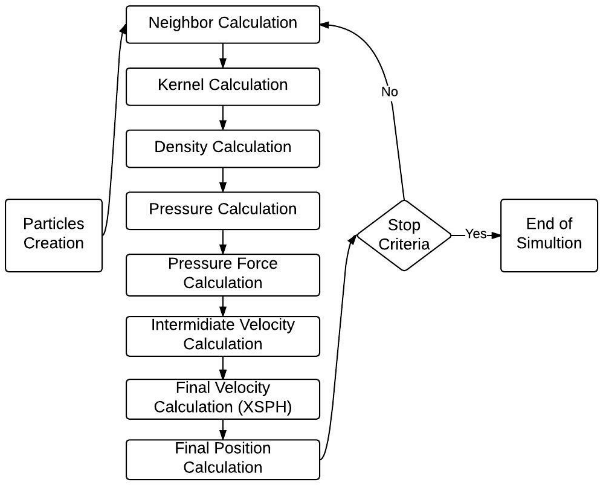

Figure 1: SPH simulation flow.

3.2 Multiphase Rendering Solution

The proposed rendering is a shader solution based on the work of Laan, Green and Sainz [65], which uses the particles position input to the rendering pipeline. The method can be described as a series of steps: for each fluid, the surface depth and thickness are calculated into different maps, and the depth map is smoothed. Then, combining the depth map with the thickness map, each fluid is rendered separately and, finally, both fluids are rendered together and combined with the surrounding scene.

The illustration of the rendering method can found in Fig. 2.

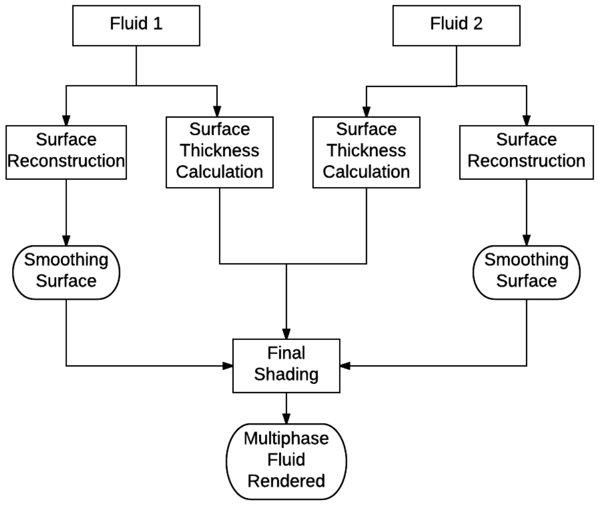

Figure 2: Render solution chart.

3.2.1 Surface Reconstruction

To reconstruct the surface of the fluid, each particle is rendered as a sphere using a point sprite (screen oriented quads) with depth replacement in the fragment shader, which means that the depth test is enabled on OpenGL. The point sprite size is calculated inversely proportional to the viewer's distance to the fluid surface, in other words, the sprites size increases as the camera approaches the fluid. The normals are calculated from the depth values, and so, will also be affected by the smoothing step.

3.2.2 Surface Thickness Calculation

To give a more reliable impression on the fluid, it is expressed by Beers law that a fluid becomes less visible depending on the amount of fluid in front of the observer, which will be referred as thickness [65].

This calculation is quite similar to the depth map creation, but instead of the depth value, the fragment shader keeps the thickness of the particle. The depth test is enabled, but additive blending is used to accumulate the value of the thickness on a certain pixel.

3.2.3 Surface Smoothing Method

After rendered the particles as spheres, the surface will resemble a jam. So, to avoid this behavior, the depth map is smoothed using a Bilateral Filter, which provides a good visual quality preserving the silhouette edges and presenting a better performance if compared with other methods [65].

The Bilateral Filter is divided into two passes, a horizontal and a vertical one. Each pass applies a spatial kernel, and the weight of a pixel inside the kernel also depends on a function in the intensity domain which decreases the weight of pixels with large intensity differences.

3.2.4 Final Shading

To create the final result, first, each fluid is rendered individually compositing the intermediate results explained before. In this step, only Phong specular highlight and a Fresnel based reflection are considered, which can be calculated as Eq. (10).

$$C_{fluid} = a(1 - F(\boldsymbol{n} \cdot \boldsymbol{v})) + bF(\boldsymbol{n} \cdot \boldsymbol{v}) + k_s(\boldsymbol{n} \cdot \boldsymbol{h})^{\alpha} \tag{10}$$

where F is the Fresnel function, a is the refracted fluid color, b is the reflected scene color, k_s and α are specular constants, $\boldsymbol{n}$ is the surface normal, $\boldsymbol{h}$ is the half-angle between camera and the light and $\boldsymbol{v}$ is the camera vector.

The view-space normal of a certain point is determined by the finite differences of the depth map. This approach may result in artifacts close to the fluid silhouettes, so, in that case, the difference is calculated in the opposite direction which can be detected by the smallest finite difference [65].

As both fluids are transparent, the thickness $T(x,y)$ is used to attenuate the refracted color [65]. So, the thicker the fluid, the less background should be seen. To create the illusion of a refraction, the thickness is also used to linearly perturb the background pixel color as seen in Eq. (12).

$$a = lerp(C_{fluid}, B(x + \beta \boldsymbol{n}_x + \beta \boldsymbol{n}_y), e^{-T(x,y)}) \tag{11}$$

$$\beta = T(x,y)\gamma \tag{12}$$

where B is the background color, γ is a constant which depends on the fluid and is used to determine how much the background is perturbed.

Then, the final color is the sum of both fluid colors and can be calculated as Eq. (13).

$$C_{final} = k(C_{f_1} + C_{f_2}) \tag{13}$$

where k is a saturation coefficient, which avoids the image to become oversaturate, f_1 and f_2 are fluid one and the other fluid, respectively.

4 Test Cases

To validate both multiphase SPH and rendering technique two test cases were performed: a density equilibrium and a 3D double dam break.

4.1 Density Equilibrium

In the first test, two fluids with different densities, 1000 kg/m^3 and 3000 kg/m^3, are inside a container. This test is made to validate the behavior of two fluids with different densities that after some time behave in a way that the heavier fluid must be under the lightest one.

The denser fluid column has a height h of 1m, width L and depth D of 3m, while the lighter fluid has height H of 2m, width L and depth D of 3m and the boundary is a box with height H_b of 6m, width L and depth D of 3m as can be seen in Fig. 3.

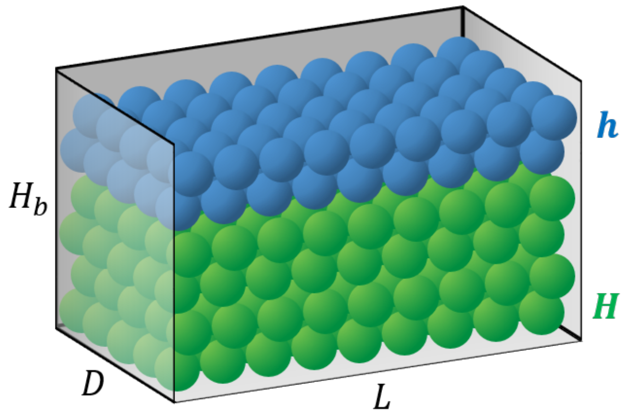

Figure 3: Density equilibrium test case. The denser fluid is illustrated by the blue particles and the lighter is represented by the green particles.

The test case was done with a 40k fluid particles, initial spacing of 0.085m and XSPH constant equal to 0.08 and $\triangle t$ of 0.0005 seconds.

4.2 3D Double Dam Break

The second test case is a 3D double dam break, where the fluid on the left has a density of 1000 kg/m^3, and the other fluid has a density of 3000 kg/m^3. This scenario is useful for solving flows that have a constant free-surface variation. Both fluids columns have a height H, width L and initial depth D of 3m, and the bottom side of the domain has width size of 12m, as can be seen in Fig. 4.

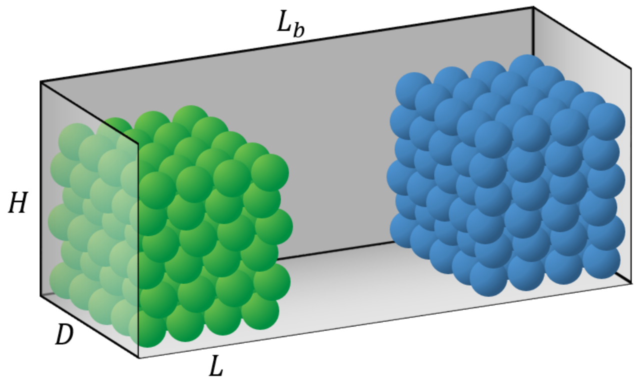

Figure 4: 3D double dam break test case. The denser fluid is illustrated by the blue particles and the lighter is represented by the green particles.

The test case was done with 400k fluid particles, XSPH constant equals to 0.08 and $\triangle t$ of 0.0005 seconds.

Also, two different configurations were constructed for the 3D dam break, one with 100k and another with 400k fluid particles, to analyze the influence of the particle number in the surface reconstruction result. And finally, to understand how the particles number influences the performance, three scenarios were used: 100k, 500k and 1M fluid particles.

Table 1: Comparison of performance (fps) varying the percentage of screen filled.

Percentage of screen filled (%)	Performance (fps)
25	60
50	40
75	15
100	10

5 Results

The proposed method was implemented on the open-source code DualSPHysics [20], which is written in C++, uses a grid to calculate the neighborhood of a particle and OpenMP to accelerate the calculations being able to simulate up to millions of particles. The rendering solution was implemented using C++, OpenGL and GLSL shader language version 4.1.

The CPU used to run the test cases was an Intel Core i7-4790L CPU @ 4.00 GHz with 32 GB of installed RAM and a Windows 10 64-bit operating system (x64). The GPU used was a NVIDIA GeForce 960 with 4 GB of RAM.

The rendering was done in 1024 x 768 resolution, which is the resolution used in [65], and the reference paper was able to achieve a performance between 44 and 55 fps for 64k particles. Although this article results cannot be compared with our results, it is important to notice that this technique reaches a real-time performance.

5.1 Density Equilibrium

The simulation proved to have a realistic behavior as the denser liquid goes to the bottom of the container over time and the XSPH was able to create realistic boundary conditions, as can be seen in the right part of Fig. 5. However, the simulation took a lot of time to converge, because the XSPH method has a big influence in particles with a few neighbors of the same fluid.

5.2 3D Double Dam Break

In the second test case, the expected behavior is also achieved as the denser liquid goes to the bottom of the container but also pushes the other fluid to the top, as can be seen in the middle of Fig. 6. It is also possible to notice the XSPH contribution as the fluid stick to the wall due to its viscosity and also the boundary condition is well established, as visualized in the right part of Fig. 6.

5.3 Rendering Performance

The render proved to have visually plausible results being able to identify each layer of fluid individually, the interface between two fluids and when they are mixed as can be seen in left part of Fig. 5. Also, as shown in the work of Laan, Green and Sainz [65], the thickness influence on the color and the transparency of the fluid were able to create a more realistic result.

The rendering had a performance of up to 60 fps depending on how close the camera was to the fluid, as can be seen in Table 1, which compared the performance (fps) on how the fluid filled the screen. The performance drop happens because, if the camera is too close to the fluid, more pixels will be smoothed and more processing will happen.

The four scenarios used in the comparison aforementioned were tested with three different number of fluid particles: 100k, 500k and 1M. It was noticed that the growth in the number of particles practically does not affect the cases in which the fluid filled 25%, but, as the number of particles increases the performance drops in the other three scenarios, as can be seen in Table 2.

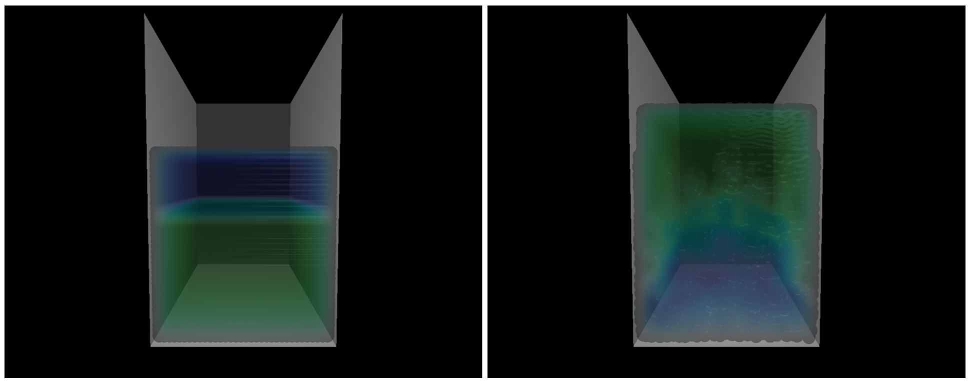

Figure 5: Density equilibrium results. Left: initial stated (t= 0s). Right: t = 10s.

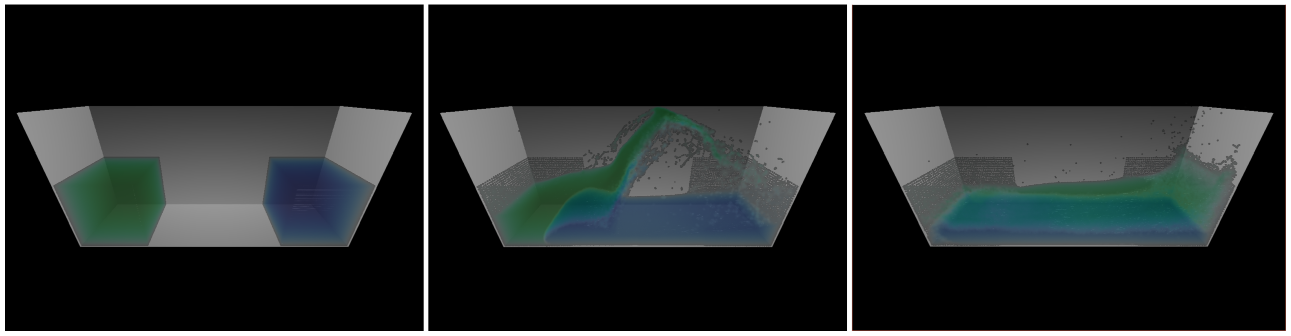

Figure 6: 3D Double Dam Break results. Left: initial stated (t = 0s). Middle: t = 2s Right: t = 5s.

Table 2: Comparison of performance (fps) varying the Percentage of screen filled and the number of particles.

% screen filled \n particles	100k	500k	1M
25%	60	60	60
50%	43	31	15
75%	27	20	10
100%	18	15	7

The happens because, as the number of particles increases, the input for the surface reconstruction and surface thickness calculation also increases.

The number of particles also influences the surface reconstruction. For the same particle size, as the number of particles increases, the surface reconstruction is smoother. These results can be seen in figure Fig. 7, which compares the smoothing results for 100k, 500k and 1M fluid particles.

As the number of particles increases, the smoothing results improves, which can be observed by comparing the left and middle results from Fig. 7. But, as the number keeps increasing, the improvement is almost imperceptible, which can be seen by comparing the middle and right results from Fig. 7.

6 Conclusion

This work had a twofold contribution: to propose a multiphase SPH formulation by extending the work of Vieira-e-Silva et al. [22] to validate if the XSPH can be integrated with a multiphase solution and to propose a render solution for this kind of simulation.

The proposed SPH method was able to simulate multiphase fluids but, to achieve that purpose, the XSPH must only be calculated for fluid particles. The method was implemented on the DualSPHysics open-source code and can be simulated up to millions of particles.

The rendering solution was able to render the fluid with performance between 10 and 60 fps for 400k fluid particles. The solution was implemented using GLSL and has the particle position as primitive, so, it can be used on other particle-based simulation methods, MPS [36], PIC [34]. The approach has limitations, such as the solution only deals with two fluids, only supports non-miscible fluids and the refraction is not realistic.

6.1 Future Works

This work can be improved in many ways. First, for the SPH model, a new force can be added to be able to control interface tension and to perform a more realistic simulation [59]. Also, a GPU solution can be used to achieve real-time simulation performance. Finally, the model can be used for coastal and other hydraulic applications to validate the model in a real world problem [66].

The rendering solution can also be improved. First, the solution should be able to render more than two fluids. Also, a more accurate blur function can be used such as the one proposed in the work of Reichl, Chajdas and Schneider [54]. Another possibility is to integrate the proposed solution into a ray tracing solution to be able to create a more realistic result [68].

Acknowledgments

The present work was supported by the CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (National Council of Scientific and Technological Development - Brazil), which supports this research, process number 456332-2013-8, and also supports the master's degree of the co-author Caio Brito, process number 830098/1999-2.

References

[1] G. Akinci, N. Akinci, M. Ihmsen, and M. Teschner. An efficient surface reconstruction pipeline for particle-based fluids. 2012.

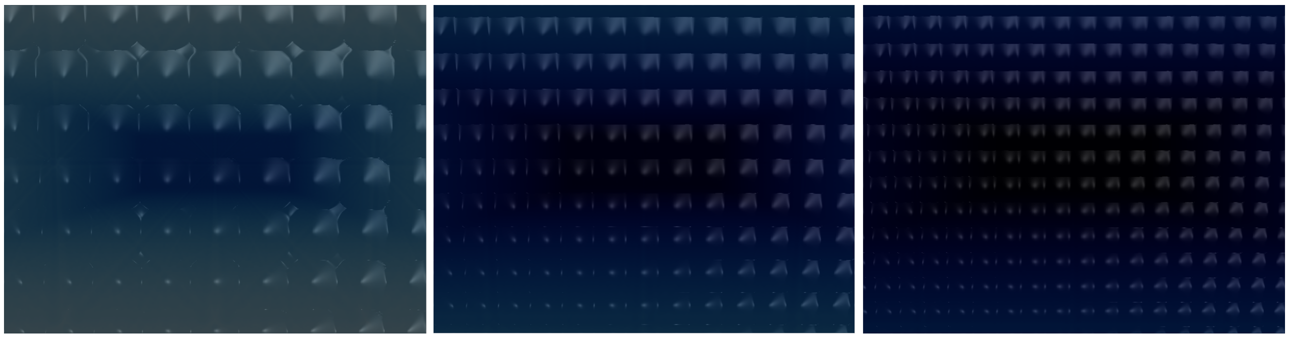

Figure 7: Density Equilibrium

[2] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner. Parallel surface reconstruction for particle-based fluids. In *Computer Graphics Forum*, vol. 31, pp. 1797–1809. Wiley Online Library, 2012.

[3] M. I. J. B. G. Akinci and M. Teschner. Animation of air bubbles with sph. 2011.

[4] N. Akinci, A. Dippel, G. Akinci, and M. Teschner. Screen space foam rendering. 2013.

[5] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Transactions on Graphics (TOG)*, 31(4):62, 2012.

[6] C. Andrea. *A meshless lagrangian method for free-surface flows and interface flows with fragmentation*. PhD thesis, PhD thesis, Universita di Roma La Sapienza, 2005.

[7] M. Asai, A. M. Aly, Y. Sonoda, and Y. Sakai. A stabilized incompressible sph method by relaxing the density invariance condition. *Journal of Applied Mathematics*, 2012, 2012.

[8] B. Ataie-Ashtiani and L. Farhadi. A stable moving-particle semi-implicit method for free surface flows. *Fluid Dynamics Research*, 38(4):241–256, 2006.

[9] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982.

[10] J. Bonet and S. Kulasegaram. A simplified approach to enhance the performance of smooth particle hydrodynamics methods. *Applied Mathematics and Computation*, 126(2):133–155, 2002.

[11] J. Bonet and T.-S. Lok. Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computer Methods in applied mechanics and engineering*, 180(1):97–115, 1999.

[12] D. L. Brown, R. Cortez, and M. L. Minion. Accurate projection methods for the incompressible navier–stokes equations. *Journal of computational physics*, 168(2):464–499, 2001.

[13] J. Chen, K. Yang, and Y. Yuan. Sph-based visual simulation of fluid. In *Computer Science & Education, 2009. ICCSE'09. 4th International Conference on*, pp. 690–693. IEEE, 2009.

[14] Y. Chen, J. Lee, and A. Eskandarian. *Meshless methods in solid mechanics*. Springer Science & Business Media, 2006.

[15] Z. Chen, Z. Zong, M. Liu, L. Zou, H. Li, and C. Shu. An sph model for multiphase flows with complex interfaces and large density differences. *Journal of Computational Physics*, 283:169–188, 2015.

[16] G. Cirio, M. Marchal, S. Hillaire, and A. Lecuyer. Six degrees-of-freedom haptic interaction with fluids. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1714–1727, 2011.

[17] G. Cirio, M. Marchal, A. Le Gentil, and A. Lécuyer. tap, squeeze and stir the virtual world: Touching the different states of matter through 6dof haptic interaction. In *Virtual Reality Conference (VR), 2011 IEEE*, pp. 123–126. IEEE, 2011.

[18] G. Cirio, M. Marchal, M. A. Otaduy, and A. Lécuyer. Six-oof haptic interaction with fluids, solids, and their transitions. In *World Haptics Conference (WHC), 2013*, pp. 157–162. IEEE, 2013.

[19] A. Crespo, J. M. Dominguez, A. Barreiro, M. Gómez-Gesteira, and B. D. Rogers. Gpus, a new tool of acceleration in cfd: efficiency and reliability on smoothed particle hydrodynamics methods. *PLoS One*, 6(6):e20685, 2011.

[20] A. Crespo, J. Domnguez, B. Rogers, M. Gmez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. Garca-Feal. Dualsphysics: Open-source parallel {CFD} solver based on smoothed particle hydrodynamics (sph). *Computer Physics Communications*, 187:204 – 216, 2015. doi: 10.1016/j.cpc.2014.10.004

[21] S. J. Cummins and M. Rudman. An sph projection method. *Journal of computational physics*, 152(2):584–607, 1999.

[22] A. L. V. e Silva, M. W. Almeida, C. J. Brito, V. Teichrieb, J. M. Barbosa, and C. Salhua. A qualitative analysis of fluid simulation using a sph variation. In *Congresso de Mtodos Numricos em Engenharia*, 2015.

[23] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics*, 183(1):83–116, 2002.

[24] R. Fraedrich, S. Auer, and R. Westermann. Efficient high-quality volume rendering of sph data. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1533–1540, 2010.

[25] A. GHASEMI V, B. Firoozabadi, and M. Mahdinia. 2d numerical simulation of density currents using the sph projection method. *European journal of mechanics. B, Fluids*, 38:38–46, 2013.

[26] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[27] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on gpus. In *Computer Graphics International*, pp. 63–70. SBC Petropolis, 2007.

[28] C. Hori, H. Gotoh, H. Ikari, and A. Khayyer. Gpu-acceleration for moving particle semi-implicit method. *Computers & Fluids*, 51(1):174–183, 2011.

[29] C. J. Horvath and B. Solenthaler. Mass preserving multi-scale sph. *Pixar Technical Memo 13-04, Pixar Animation Studios*, 2013.

[30] X. Hu and N. Adams. A constant-density approach for incompressible multi-phase sph. *Journal of Computational Physics*, 228(6):2082–2091, 2009.

[31] X. Hu and N. A. Adams. A multi-phase sph method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, 213(2):844–861, 2006.

[32] M. Ihmsen, N. Akinci, G. Akinci, and M. Teschner. Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer*, 28(6-8):669–677, 2012.

[33] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner. Sph fluids in computer graphics. 2014.

[34] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):51, 2015.

[35] S. Koshizuka, A. Nobe, and Y. Oka. Numerical analysis of breaking waves using the moving particle semi-implicit method. *International Journal for Numerical Methods in Fluids*, 26(7):751–769, 1998.

[36] S. Koshizuka and Y. Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear science and engineering*, 123(3):421–434, 1996.

[37] Ø. E. Krog and A. C. Elster. Fast gpu-based fluid simulations using sph. In *Applied Parallel and Scientific Computing*, pp. 98–109. Springer,

2012.

[38] E.-S. Lee, C. Moulinec, R. Xu, D. Violeau, D. Laurence, and P. Stansby. Comparisons of weakly compressible and truly incompressible algorithms for the sph mesh free particle method. *Journal of computational physics*, 227(18):8417–8436, 2008.

[39] E.-S. Lee, D. Violeau, R. Issa, and S. Ploix. Application of weakly compressible and truly incompressible sph to 3-d water collapse in waterworks. *Journal of Hydraulic Research*, 48(S1):50–60, 2010.

[40] M. Liu and G. Liu. Smoothed particle hydrodynamics (sph): an overview and recent developments. *Archives of computational methods in engineering*, 17(1):25–76, 2010.

[41] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, vol. 21, pp. 163–169. ACM, 1987.

[42] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977.

[43] J. Monaghan and A. Kocharyan. Sph simulation of multi-phase flow. *Computer Physics Communications*, 87(1-2):225–235, 1995.

[44] J. Monaghan and A. Rafiee. A simple sph algorithm for multi-fluid flow with high density ratios. *International Journal for Numerical Methods in Fluids*, 71(5):537–561, 2013.

[45] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005.

[46] J. J. Monaghan and J. B. Kajtar. Sph particle boundary forces for arbitrary boundaries. *Computer Physics Communications*, 180(10):1811–1820, 2009.

[47] M. Müller, S. Schirm, and S. Duthaler. Screen space meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 9–15. Eurographics Association, 2007.

[48] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 237–244. ACM, 2005.

[49] J. Orthmann, H. Hochstetter, J. Bader, S. Bayraktar, and A. Kolb. Consistent surface model for sph-based fluid transport. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 95–103. ACM, 2013.

[50] W.-M. Pang, J. Qin, Y.-P. Chui, and P.-A. Heng. Fast prototyping of virtual reality based surgical simulators with physx-enabled gpu. *Transactions on edutainment IV*, pp. 176–188, 2010.

[51] C. Peng, G. Xu, W. Wu, H.-s. Yu, and C. Wang. Multiphase sph modeling of free surface flow in porous media with variable porosity. *Computers and Geotechnics*, 81:239–248, 2017.

[52] S. Premžoe, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. In *Computer Graphics Forum*, vol. 22, pp. 401–410. Wiley Online Library, 2003.

[53] A. Rafiee, M. Manzari, and M. Hosseini. An incompressible sph method for simulation of unsteady viscoelastic free-surface flows. *International Journal of Non-Linear Mechanics*, 42(10):1210–1223, 2007.

[54] F. Reichl, M. G. Chajdas, J. Schneider, and R. Westermann. Interactive rendering of giga-particle fluid simulations. In *Eurographics/ACM SIGGRAPH Symposium on High Performance Graphics*, pp. 105–116. The Eurographics Association, 2014.

[55] B. D. Rogers, R. A. Dalrymple, M. Gesteira, and O. Knio. Smoothed particle hydrodynamics for naval hydrodynamics. In *Proc 18th Int Workshop on Water Waves and Floating Bodies. In: Clermont AH, Ferrant P, editors. Division Hydrodynamique Navale du Laboratoire de Mécanique des Fluides. France: Ecole Centrale de Nantes*, 2003.

[56] H. Schechter and R. Bridson. Ghost sph for animating water. *ACM Transactions on Graphics (TOG)*, 31(4):61, 2012.

[57] M. S. Shadloo, A. Zainali, M. Yildiz, and A. Suleman. A robust weakly compressible sph method and its comparison with an incompressible sph. *International Journal for Numerical Methods in Engineering*, 89(8):939–956, 2012.

[58] L. D. G. Sigalotti, J. Klapp, E. Sira, Y. Meleán, and A. Hasmy. Sph simulations of time-dependent poiseuille flow at low reynolds numbers. *Journal of computational physics*, 191(2):622–638, 2003.

[59] B. Solenthaler and R. Pajarola. Density contrast sph interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 211–218. Eurographics Association, 2008.

[60] B. Solenthaler, J. Schläfli, and R. Pajarola. A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds*, 18(1):69–82, 2007.

[61] J. Swegle, D. Hicks, and S. Attaway. Smoothed particle hydrodynamics stability analysis. *Journal of computational physics*, 116(1):123–134, 1995.

[62] K. Szewc, J. Pozorski, and J.-P. Minier. Analysis of the incompressibility constraint in the smoothed particle hydrodynamics method. *International Journal for Numerical Methods in Engineering*, 92(4):343–369, 2012.

[63] A. M. Tartakovsky and P. Meakin. A smoothed particle hydrodynamics model for miscible flow in three-dimensional fractures and the two-dimensional rayleigh–taylor instability. *Journal of Computational Physics*, 207(2):610–624, 2005.

[64] A. M. Tartakovsky and P. Meakin. Pore scale modeling of immiscible and miscible fluid flows using smoothed particle hydrodynamics. *Advances in Water Resources*, 29(10):1464–1478, 2006.

[65] W. J. van der Laan, S. Green, and M. Sainz. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pp. 91–98. ACM, 2009.

[66] D. Violeau and B. D. Rogers. Smoothed particle hydrodynamics (sph) for free-surface flows: past, present and future. *Journal of Hydraulic Research*, 54(1):1–26, 2016.

[67] Z. Wang and Y. Wang. Haptic interaction with fluid based on smooth particles and finite elements. In *International Conference on Computational Science and Its Applications*, pp. 808–823. Springer, 2014.

[68] X. Xiao, S. Zhang, and X. Yang. Real-time high-quality surface rendering for large scale particle-based fluids. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, p. 12. ACM, 2017.

[69] R. Xu, P. Stansby, and D. Laurence. Accuracy and stability in incompressible sph (isph) based on the projection method and a new approach. *Journal of Computational Physics*, 228(18):6703–6725, 2009.

[70] X. Yan, Y.-T. Jiang, C.-F. Li, R. R. Martin, and S.-M. Hu. Multiphase sph simulation for interactive fluids and solids. *ACM Transactions on Graphics (TOG)*, 35(4):79, 2016.

[71] X. Yang, M. Liu, and S. Peng. Smoothed particle hydrodynamics modeling of viscous liquid drop without tensile instability. *Computers & Fluids*, 92:199–208, 2014.

[72] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics (TOG)*, 32(1):5, 2013.

[73] J. Yu, C. Wojtan, G. Turk, and C. Yap. Explicit mesh surfaces for particle based fluids. In *Computer Graphics Forum*, vol. 31, pp. 815–824. Wiley Online Library, 2012.

[74] Y. Zhang, X. Ban, X. Liu, and X. Wang. Adaptiving time steps for sph cloth-fluid coupling. In *Cyberworlds (CW), 2016 International Conference on*, pp. 143–146. IEEE, 2016.

[75] Y. Zhang, B. Solenthaler, and R. Pajarola. Adaptive sampling and rendering of fluids on the gpu. In *Proceedings of the Fifth Eurographics/IEEE VGTC conference on Point-Based Graphics*, pp. 137–146. Eurographics Association, 2008.

[76] X. Zhu, L. Cheng, L. Lu, and B. Teng. Implementation of the moving particle semi-implicit method on gpu. *SCIENCE CHINA Physics, Mechanics & Astronomy*, 54(3):523–532, 2011.

[77] T. Zirr and C. Dachsbacher. Memory-efficient on-the-fly voxelization of particle data. In *EGPGV*, pp. 11–18, 2015.