Graduação em Ciência da Computação

Caio José dos Santos Brito

A RAY TRACED RENDERING SOLUTION FOR PARTICLE-BASED FLUID SIMULATION

B.Sc. Dissertation

Federal University of Pernambuco
secgrad@cin.ufpe.br
www.cin.ufpe.br/~graduacao

RECIFE
2015

Federal University of Pernambuco
Center for Informatics
B.Sc in Computer Science

Caio José dos Santos Brito

A RAY TRACED RENDERING SOLUTION FOR PARTICLE-BASED FLUID SIMULATION

A B.Sc. Dissertation presented to the Center for Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Bachelor in Computer Science.

Advisor: *Veronica Teichrieb*
Co-Advisor: *Mozart William Santos Almeida*

RECIFE
2015

I dedicate this thesis to all my family, friends and professors
who gave me the necessary support to get here.

Acknowledgements

To God and the Quinta de Luz for guidance in the right direction.

To my mother, for the support and care through all those years, I wouldn’t be anything without her.

To my friends, in particular to the "alopras", for all the good moments that helped me to relax and always smile.

To everyone at the Voxar Labs for all the help and good times together.

To my advisors Veronica and Mozart, who always trusted me that I could be reach my goals.

Resumo

Simulação de fluidos utilizando métodos sem a presença de malha tornou-se cada vez mais eficiente para resolver problemas de mecânica que lidam com grandes deformações e tornou-se popular em muitas aplicações, tais como engenharia naval, engenharia mecânica, filmes e jogos. Um dos principais métodos é o Smoothed Particle Hydrodynamics (SPH). Esse trabalho tem como objetivo investigar a aplicabilidade do método SPH e meios de reconstruir a superfície do fluido utilizando um algorithm de ray tracing. O SPH fracamente compressível (WCSPH) é implementado utilizando uma equação de estado para calcular a pressão do sistema e a abordagem XSPH para simular a viscosidade. Uma versão paralela do método é implementada usando a tecnologia CUDA para melhorar o desempenho de cálculo de tempo. Para reconstruir a superfície do fluido, utilizando os resultados do WCSPH, um sistema de traçado de raios chamado Real Time Ray Tracer ou simplesmente RT^2 é usado. Para criar uma visualização mais realista da superfície, dois métodos de alisamento foram testados e comparados: o Gaussian Blur e o Screen Space Curvature Flow. Ambos os algoritmos foram capazes de criar uma superfície lisa, mas a abordagem de Curvature Flow foi capaz de criar um resultado mais realista, porém com um baixo desempenho em fps, enquanto que a abordagem Gaussian criou um resultado menos realista, mas com um alto desempenho em fps.

Palavras-chave: Ray Tracing, Smoothed Particle Hydrodynamics, Reconstrução de Superfície, Suavização, Simulação de Fluidos, Renderização de Fluidos

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 aims to investigate the applicability of the SPH to fluid simulation and ways of reconstruct the fluid surface based on a ray tracer algorithm. A Weakly Compressible SPH (WCSPH) is implemented which uses a state equation to calculate the pressure of the system and the XSPH approach to simulate the viscosity. A parallel version of the method is implemented using CUDA technology to improve the time calculation performance. To reconstruct the fluid surface of the WCSPH results a ray tracing system named Real Time Ray Tracer or simply RT^2 is used. In order to create a more realistic surface visualization, two methods of smoothing are used to create a more realistic surface: the Gaussian Blur and the Screen Space Curvature Flow approach. Both algorithms were able to create a smooth surface but the Curvature Flow approach was able to create a more realistic result but with a low fps performance, while the Gaussian approach creates a less realistic result but with high fps performance.

Keywords: Ray Tracing, Smoothed Particle Hydrodynamics, Surface Reconstruction, Smoothing, Fluid Simulation, Fluid Rendering

List of Figures

List of Tables

Contents

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 (3).

To overcome such challenges, mesh-free methods may be used such as the Smoothed Particle Hydrodynamics (SPH) (4) and Moving Particles Semi-Implicit (MPS) methods (5). 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 (6).

In this work, the SPH method will be used in order to simulate the fluid behavior. This method was designed over three decades ago by Lucy (7) and Gingold and Monaghan (4) and intended to astrophysics applications. Since their conception, there have been continuous improvements and adaptations of the original method in order to simulate various kinds of physical phenomena.

One of the biggest problems for particle methods is the modeling of the interaction between solids and fluids (boundary condition). Several solutions have been presented, which may lead to the high degree of numerical precision or only the visual accuracy (3). However, given the high flexibility of implementation of this type of method, it has been used in the gaming industry (8) and in the film industry (9).

In those industries, another big challenge is to realistically render the simulated fluid. To visualize the simulation results, it is necessary to rebuild the surface using the particles identified as free surface (10). The rendering of the results can be made using several methods such as direct rendering (11), 3D scalar field (10), volume rendering (12) or a screen space approach (8).

Recently, 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 (13). The second approach renders the particles as spheres or point sprites and applies a smoothing filter to the depth map in order to create a better looking final surface render.

The 3D scalar field approach includes 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 (10).

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 a jam. To overcome this characteristic, a smoothing function is applied to the depth map of the scene and it 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 (1).

1.1 Goals

The goal of this work is to investigate the SPH method and ways of reconstructing the surface from the fluid simulation results based on a ray tracing algorithm. To achieve this goal, a compressible SPH method is implemented based on the work of (14) and (15) and, to improve the calculation time, a parallel implementation of the method is implemented using CUDA.

To reconstruct the fluid surface, a ray tracing system named Real Time Ray Tracer - RT^2 (16) is used, and, to obtain a more realistic surface, two smoothing algorithms are implemented and compared in regards to visual quality and rendering time. The smoothing algorithms used to smooth the surface are: Gaussian Blur (17) and Screen Space Curvature Flow (8).

1.2 Document Structure

The next chapter discusses the state of art of SPH-based fluid simulations and also how to render the results from the simulation. The Chapter 3 describes the SPH technique, its governing equations, main applications and different ways of calculation the fluid behavior. After that, The chapter 4 describes the ray tracing algorithm and the smoothing methods applied in this work. The Chapter 5 shows the methodology adopted throughout the development of this work and chapter 6 describes the test cases used in this work and how to analyze the results. The Chapter 7 discusses the results of this work in regarding numerical validation of the SPH method developed, GPU speedup, surface reconstruction, visual quality and performance. At last, in chapter 8, the conclusions are discussed and the contributions of this work are exposed, with future possibilities and enhancements being discussed.

2

SPH-based Fluid Simulation and Rendering: State of the Art

The SPH method was introduced by (7) and (4) in order to model astrophysical phenomena. Since then it has been vastly extended to model fluids (18), (19) and even solids behavior (20), mainly under those aspects which could limit the applications simulated by mesh-based approaches, such as high deformations, for instance.

A straightforward adaptation to the original SPH method is the application for weakly compressible fluids. In this kind of fluid, the pressure can be calculated by an equation of state. The works (21), (22), (23) and (24) show comparisons between implementations of weakly compressible and truly incompressible methods, in which are applied techniques such as the one introduced by (25).

In the truly incompressible methods, the density 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 requires more computation time as stated in the works of (26), (27), (28) and (29).

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 (3), pressure influence (27), density compensation (30) or a geometrical approach (31).

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 (24), (32), (33) and (34) show some existing options concerning viscosity improvements.

The most common way of simulating a turbulent flow is to incorporate the Reynolds-Averaged Navier-Stokes turbulence (RANS) model into the SPH method (35). It's worth noticing that the majority of the meshless works found in the literature deal with a low Reynolds' number such as the works of (36), (37), (38), (33), (39) and (40). This last work discusses the common usage of this value in the literature.

In order to explain the effects of the smoothing function on meshless simulations, the works (33), (41), (42), (43), (44) and (3) 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 processor (GPGPU) techniques, in order to decrease the time consumption, as shown in the works of (45), (46), (47), (48) and (49).

In the SPH literature, many methods have been presented in order 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 (13) is used to generate a triangular mesh of the isosurface of this 3D field. The choice of the scalar field formulation is the key to create a high quality surface; the simplest choice is to use a blobbies approach, also known as metaballs (50), 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 (51) (52) (53) , which can generate a better visual result in sharp features and edges (10).

A second approach to render the fluid surface is to use an explicit method, frequently used in an Eulerian context (54). In the SPH literature, a few works can be found using this approach, such as (55), which change the surface position using information of the particles simulation. Those methods have a high memory consumption and, in order to avoid this problem, methods such as point splatting (56) and ray-isosurface intersection with metaballs can be used (57).

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 and create a depth map of the scene, smooth this map in order to create a more coherent surface and render the scene using the smoothed depth map. Many algorithms can be used to smooth the depth map: a binomial filter (58), a Gaussian filter (8), a curvature flow (8) (59), and a post smoothing filter (1).

3

Smoothed Particle Hydrodynamics

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. (3.1).

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla P + \frac{1}{\rho}\nabla \cdot \boldsymbol{\tau} + Fext \tag{3.1}$$

where $\mathbf{u}$ is the velocity of the fluid, t is the time, ρ is the density of the fluid, P is the pressure of the fluid system, $\boldsymbol{\tau}$ is the deviatoric stress tensor, and $Fext$ is the external forces in the fluid system. This approach can be described in two parts: the kernel approximation and the problem discretization. In the kernel approximation step, a function $f(x)$ can be represented by an integral interpolation as in Eq. (3.2) with second order accuracy (3):

$$f(x) = \int_{\Omega} f(x^{'})\delta(x - x^{'})dx^{'} \tag{3.2}$$

where f is a function of the position vector x, and $\delta(x - x^{'})$ is the Dirac delta function given by Eq. (3.3).

$$\delta(x - x^{'}) = \begin{cases} 1, x = x^{'} \\ 0, x \neq x^{'} \end{cases} \tag{3.3}$$

The delta function can be approximated by a kernel function $W(x - x^{'}, h)$, where h is the smoothing length, so the function f can be expressed by Eq. (3.4) and its derivative can be calculated by Eq. (3.5) and Eq. (3.6).

$$\langle f(x) \rangle = \int_{\Omega} f(x^{'})W(x - x^{'}, h)dx^{'} \tag{3.4}$$

$$\langle \nabla \cdot f(x) \rangle = -\int_{\Omega} f(x^{'}) \cdot \nabla W(x - x^{'}, h)dx^{'} \tag{3.5}$$

$$\langle \nabla f(x) \rangle = \int_{\Omega} f(x^{'})\nabla W(x - x^{'}, h)dx^{'} \tag{3.6}$$

The kernel function W is a symmetric smooth function which defines the influence area of a particle and should satisfy some conditions (3):

1. Normalization condition, that can be expressed as in Eq. (3.7):

$$\int_{\Omega} W(x-x^{'},h)dx^{'} = 1 \tag{3.7}$$

2. Limit condition, which can be expressed by Eq. (3.8):

$$\lim_{h\to 0} W(x-x^{'},h) = \delta(x-x^{'},h) \tag{3.8}$$

3. Compact domain condition defined by Eq. (3.9), which limits the domain of the problem to a local solution:

$$\lim_{h\to 0} W(x-x^{'},h) = 0 \;\; when|(x-x^{'},h)| > kh \tag{3.9}$$

There are many possible choices for a kernel function. Most of SPH simulations use a cubic spline kernel as in Eq. (3.10), which resembles a Gaussian function but its second derivative has some results close to a linear function and may cause transverse mode instability (37). Another possibility is to use a quintic kernel function as in Eq. (3.11). This kernel is more stable because it doesn't lead to a transverse mode instability (37). There are some applications that don't need high accuracy, for instance, in games, so simpler kernels can be used, for instance, the poly6 kernel expressed by Eq. (3.12) or the spiky kernel shown in Eq. (3.13) (60).

$$W(r,h) = \alpha_d \begin{cases} \frac{2}{3} - r^2 + \frac{1}{2}r^5, & if \;\; 0 \le r \le 1 \\ \frac{1}{6}(2-r)^3, & if \;\; 1 < r \le 2 \\ 0, & otherwise \end{cases} \tag{3.10}$$

where $\alpha_d = 1/h$, $5/7\pi h^2$ and $3/2\pi h^3$, for one, two and three dimensions, respectively.

$$W(r,h) = \alpha_d \begin{cases} (3-r)^5 - 6(2-r)^2 + 15(1-r)^5, & if \;\; 0 \le r \le 1 \\ (3-r)^5 - 6(2-r)^5, & if \;\; 1 < r \le 2 \\ (3-r)^5, & if \;\; 2 < r \le 3 \\ 0, & otherwise \end{cases} \tag{3.11}$$

where $\alpha_d = 120/h$, $7/478\pi h^2$ and $3/359\pi h^3$, for one, two and three dimensions, respectively.

$$W(r,h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3, & if \;\; 0 \le r \ge h \\ 0, & othewise \end{cases} \tag{3.12}$$

$$W(r,h) = \frac{15}{\pi h^6} \begin{cases} (h-r)^3, & if \;\; 0 \leq r \geq h \\ 0, & otherwise \end{cases} \tag{3.13}$$

The second part of the SPH method is to approximate the continuous hydrodynamics problem into a series of particles. An amount of fluid is described as a finite number of particles. Each particle in a position x has velocity u, mass m, density ρ, viscosity μ and an influence radius h which describes the interaction of a particle over its neighbors (61), as illustrated in Figure 3.1.

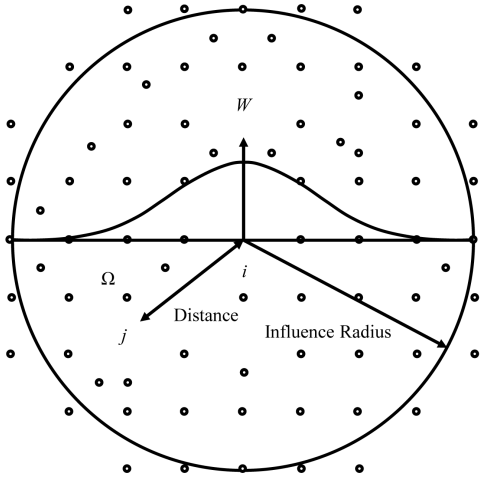


Figure 3.1: Particle approximation for a two-dimensional problem.

The influence radius of a particle defines a domain, being and area (in 2D) or a volume (in 3D) of influence (3). Given that there are two different particles inside the domain of a particle a, the one closer to the particle a suffers more influence than the other. Different influence radius can be assigned to each particle in the system and the domain can have different shapes as suggested by (3).

To get the neighborhood from a single particle, a geometric comparison is used between the distances of two particles. If a couple of particles are within a distance less than the influence domain, those particles are neighbors of each other and the radius of influence of a particle can be calculated as $1.3dx$, where dx is the initial spacing of the particles, according to the work of (31).

The neighborhood search is a potentially time-consuming step and is usually optimized using an accelerated spatial access structure like a uniform grid or an octree, instead of a naive brute-force search (62).

In the discrete formulation, the interpolation (3.4) can be defined as Eq. (3.14):

$$f(x_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) W(x - x_j, h) \tag{3.14}$$

where N is the number of neighbors of a particle and j is the index of the neighbor particle.

Using the same approach, the divergent and the gradient operators can be calculated as in Eq. (3.15) and Eq. (3.16):

$$\nabla \cdot f(x_i) = -\sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) \cdot \nabla W(x - x_j, h) \tag{3.15}$$

$$\nabla f(x_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) \nabla W(x - x_j, h) \tag{3.16}$$

Those derivatives may lead to a large numerical error. So, to overcome those limitations, some algebraic operations are done and stable forms of the derivatives can be found, as in Eq. (3.17) (63):

$$\nabla \cdot f(x_i) = \rho_i \sum_{j=1}^{N} m_j [\frac{f(x_j)}{\rho_j^2} + \frac{f(x_i)}{\rho_i^2}] \cdot \nabla W(x - x_j, h) \tag{3.17}$$

3.1 Weakly Compressible SPH (WCSPH) Method

The Navier-Stokes equation describes the fluid movement in three main components: pressure, viscosity and external forces. The WCSPH solves the fluid movement by considering the fluid as a weakly compressible system, which is based on the fact that every incompressible fluid is actually a little compressible, and because of that the method simulates a quasi-incompressible equation to model the simulation (15). In order to calculate those components, the first step is to calculate the particle densities, which can be calculated using the density summation equation as expressed in Eq. (3.18) (15):

$$\rho_i = \sum_j m_j W_{ij} \tag{3.18}$$

This approach can make the simulation unstable for particles near the boundary or at the free surface, caused by insufficient number of particles inside the kernel. Two common ways of solving this instability is to normalize the kernel so the density will be calculated by Eq. (3.19) or it can be calculated using the continuity equation as in Eq.(3.20) (3):

$$\rho_i = \frac{\sum_j m_j W_{ij}}{\sum_j \frac{m_j}{\rho_j} W_{ij}} \tag{3.19}$$

$$\frac{d\rho_i}{dt} = \sum_j m_j (\mathbf{u}_i - \mathbf{u}_j) \nabla W_{ij} \tag{3.20}$$

After calculating the density of the particles in the system, the next step is to calculate their pressures. For a weakly compressible system, there are two main options: 1) For higher compressibility, an ideal gas equation such as (3.21) can be used; 2) In cases where the low

density variation must be enforced, the Tait's equation (3.22) can be used (2):

$$P_i = k_p(\rho_i - \rho_0) \tag{3.21}$$

$$P_i = B((\frac{\rho_i}{\rho_0})^\gamma - 1) \tag{3.22}$$

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 7 (64). The pressure force is commonly calculated using the derivative expression by Eq. (3.17), which results in the Eq. (3.23). This approach ensures a modular equality between two particles and conserves linear and angular momentum, leading to a more stable simulation (61):

$$\frac{1}{\rho_i}\nabla P_i = \sum_j m_j(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2})\nabla W_{ij} \tag{3.23}$$

To calculate the viscosity term of the governing equations, there are some approaches that can be used. For cases which need to model strong shocks, an artificial viscosity (3.24) can be used (37):

$$\prod_{ij} = \begin{cases} \dfrac{-\alpha \bar{c_{ij}}\theta_{ij} + \beta\theta_{ij}^2}{\bar{\rho_{ij}}}, \mathbf{u}_{ij}\cdot\mathbf{x}_{ij} < 0 \\ 0, \mathbf{u}_{ij}\cdot\mathbf{x}_{ij} \geq 0 \end{cases} \tag{3.24}$$

where the parameters found in the equation above are given by:

$$\theta_{ij} = \frac{h_{ij}\mathbf{u}_{ij}\cdot\mathbf{x}_{ij}}{r_{ij}^2 + \varepsilon^2} \tag{3.25}$$

$$\bar{c_{ij}} = \frac{1}{2}(c_i + c_j) \tag{3.26}$$

$$\bar{\rho_{ij}} = \frac{1}{2}(\rho_i + \rho_j) \tag{3.27}$$

$$\bar{h_{ij}} = \frac{1}{2}(h_i + h_j) \tag{3.28}$$

$$\mathbf{x}_{ij} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j) \tag{3.29}$$

$$\mathbf{u}_{ij} = \frac{1}{2}(\mathbf{u}_i + \mathbf{u}_j) \tag{3.30}$$

where α, β and ε are constants set around 1, 1 and $0.1h_{ij}$, respectively, and c_i and c_j are the respective speeds of sound for particles i and j.

Despite the fact that this approach is used to model real viscosity, the results are not the most accurate. But this formulation guarantees the conservation of the angular momentum, which is crucial for simulation cases that may have a large fluid velocity or a big free surface, and simulate a shear and bulk viscosity (37). Another option is to use the SPH formalism to calculate the viscosity force acting on a particle, which can be expressed as (3.31). This formulation gives a good visual result but numerical accuracy is not guaranteed (60):

$$f_i^{viscosity} = \mu \sum_j \frac{m_j(\mathbf{u}_j - \mathbf{u}_i)}{\rho_j} \nabla^2 W_{ij} \tag{3.31}$$

where μ is the dynamic viscosity.

In order to achieve a better numerical accuracy of the simulation, the viscous diffusion estimation defined in Eq. (3.32) can be used. This approach combines the standard SPH first derivative with the first derivative of a finite difference and conserves linear momentum exactly while the angular momentum is approximately conserved (37):

$$(\frac{1}{\rho}\nabla \cdot \mu\nabla)\mathbf{u}_i = \sum_j \frac{(m_j(\mu_i + \mu_j)\mathbf{x}_{ij} \cdot \nabla W)}{\rho_i \rho_j (x_{ij}^2 + 0.01h^2)} u_{ij} \tag{3.32}$$

A simple way to simulate the viscosity in the system is to use the XSPH approach. This formulation is computationally cheaper than the other methods and is easier to tune because it only uses one tunable parameter (64). This method forces particles near each other to move with close velocity and conserves angular and linear momentum approximately by dumping the particle velocity using Eq. (3.33) (64):

$$\mathbf{u}_i = \mathbf{u}_i + \varepsilon \sum_j m_b \frac{(\mathbf{u}_i - \mathbf{u}_j)}{\bar{\rho}_j} W_{ij} \tag{3.33}$$

where 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 acting upon the system which is commonly represented by the gravity. The particle new velocities and positions are calculated using a simple first order Euler time integration described by Eq. (3.34) and Eq. (3.35), respectively (64):

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t + \mathbf{a}_i^t t \tag{3.34}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{u}_i^{t+1} t \tag{3.35}$$

where $\mathbf{a}_i$ is the particle i acceleration.

4

Rendering the Fluid Surface

4.1 Ray Tracing

A ray tracing algorithm is used in order to reconstruct the fluid surface. The ray tracer is a technique which has been used for over three decades to synthesize images based on natural physical phenomena. In the computer graphics field, the ray tracing was originated in 1968 with the work of Arthur Appel (65) to solve the visibility problem. The recursive ray tracing was introduced by Turner Whitted in 1980 (66); using his algorithm, images were synthesized by simulating reflection, refraction and shadow effects. Using a physical phenomenon based model, instead of using a raster algorithm, the algorithm traces rays on the scene from the camera, being possible to render images with more details and more realistic.

Unlike the raster algorithm, ray tracing can print objects without the need of transforming each object into a polygonal mesh, giving more precision to the method. For instance, a sphere can be represented as a algebraic entity and the spatial equation is able to get all the information to a proper rendering.

In the beginning, the ray tracing algorithm was used for off-line rendering purposes due to the high computational demand. But with the increase of computational performance of the CPU and GPU, many works described real-time approaches for the algorithm (67) (68) (69) (70).

The ray tracer initializes its processing by emitting rays from the camera position into the 3D scene. Those rays come out from the camera to the cut screen space, which represents the final image. The emitted rays directed from the camera are called primary rays. In a simple ray tracer, a ray comes out of the camera to each pixel of the image; this process is called ray casting and the ray can be described as Eq. 4.1.

$$\mathbf{R}(t) = \mathbf{E} + t\mathbf{D} \tag{4.1}$$

where $\mathbf{R}$ is the ray position, $\mathbf{E}$ is the eye position, $\mathbf{D}$ is the direction of the ray and t is the parameter [1...+∞]

When a ray intersects an object of the scene, a new ray is emitted as reflected, refracted, shadow or illumination rays and those rays are called secondary rays. A reflected ray is a

secondary ray which bounces off the surface after a hit by a ray. The angle of reflection is equal to the incident angle and the new ray direction is calculated by Eq. 4.2.

$$\mathbf{R}_{out} = 2\mathbf{N}(\mathbf{N} \cdot \mathbf{R}_{in}) - \mathbf{R}_{in} \tag{4.2}$$

A refracted ray is a secondary ray which is conducted by the Snell's law expressing that the products of the refractive indices and the sines of the angle of incidence and refraction must be equal, as in Eq. 4.3

$$n_1 \cdot sin(\theta_i) = n_2 \cdot sin(\theta_r) \tag{4.3}$$

where n_1 and n_2 are the normals for each medium and θ_i and θ_r are the angles of incidence and refraction, respectively.

In order to render the results from the SPH simulation, each particle of the scene is rendered as a sphere. In the ray tracer, a sphere can be expressed by Eq. 4.4 and any point of the spherical surface must satisfy this equation.

$$(P_x - C_x)^2 + (P_y - C_y)^2 + (P_z - C_z)^2 = R^2 \tag{4.4}$$

where **P** = (P_x,P_y,P_z) is any point on the sphere, **C** = (C_x,C_y,C_z) is the center of it and R is the radius of the sphere.

To reconstruct the surface of the fluid, the ray tracing intersects the primary ray with the sphere of the scene and the intersection between a ray and a sphere can be calculated by an algebraic approach.

The first step is to substitute the parametric ray equation into the surface equation(s). This means solving the intersection for all points P that are both on the ray and on one of the solid's surfaces. So after substituting Eq. 4.1 in Eq. 4.4, the intersection between a ray and a sphere is expressed by Eq. 4.5.

$$((E_x + tD_x) - C_x)^2 + ((E_y + tD_y) - C_y)^2 + ((E_z + tD_z) - C_z)^2 = R^2 \tag{4.5}$$

After expanding this equation, the same can be expressed by a quadratic equation 4.6 which can be solved by the Bhaskara equation.

$$at^2 + bt + c \tag{4.6}$$

where

$$a = D_x^2 + D_y^2 + D_z^2 \tag{4.7}$$

$$b = 2(D_x(E_x - C_x)) + D_y(E_y - C_y) + D_z(E_z - C_z) \tag{4.8}$$

$$c = (E_x - C_x)^2 + (E_y - C_y)^2 + (E_z - C_z)^2 - R^2 \tag{4.9}$$

The unit normal of the sphere can be calculated using equation 4.10:

$$\widehat{\mathbf{n}} = \frac{\mathbf{P} - \mathbf{C}}{|\mathbf{P} - \mathbf{C}|} \tag{4.10}$$

where **P** is a point of the sphere.

In this work, to render the results from the SPH simulation and reconstruct the fluid surface, a ray tracing system named Real Time Ray Tracer or simply RT^2 (16) was used. It does the ray tracing process entirely on GPU using the CUDA programming model (71). The RT^2 gives support to primary and secondary rays as explained in Whitted's simple model (66) and also supports visibility tests, reflections, refractions and shadows. Moreover, the ray tracer gives support to geometric primitive types, such as triangles, cylinders, spheres and parametric surfaces.

In the tracing system, an iterative version of Whitted's method is implemented, which leads to higher performance on CUDA and every stage of the rendering is integrated into a single CUDA kernel and uses the concept of persistent threads to avoid idle threads inside a block (72). Due to this approach, instead of defining thousands of blocks to be processed, only the necessary ones are created to fully occupy the stream multiprocessors of the GPU.

To obtain the rendering results with high performance, the RT^2 supports 5 acceleration structures: Bounding Volume Hierarchy (BVH), Octree, Uniform Grid, KD-Tree and Bounding Interval Hierarchy (BIH) (16). The KD-Tree ray traversal achieved better performance if compared to the other structures, improving the performance on a 12-39% approximate range. However, this structure consumes more memory than the others. Because of that, in cases of high complex geometric scene or if the architecture has a limited amount of memory, the BVH and the BIH are more suitable options (16).

4.2 Surface Reconstruction

After rendering the spheres using the ray tracer, the result is a surface with a "blobby" or jelly-like look. In order to create a more realistic surface without the need of creating any other structure besides the fluid particles, a blur algorithm is applied to the surface to minimize the difference between points of the surface close to each other.

To achieve this purpose, a screen space approach is used (8). The first step is to create the depth map of the scene; this depth map is created using the ray tracer procedure which retains the closest value at each pixel. After creating the depth map of the scene, a smoothing algorithm is used to create a more realistic look for the fluid surface. With the new depth map, the normals of the surface are calculated and with the new surface, the render process can continue.

To smooth the depth map, two methods were used: the Gaussian Blur and Curvature Flow process.

4.2.1 Gaussian Blur

In the Gaussian Blur (17) method, the new value of a depth from the map is calculated by averaging its value with the ones in the neighborhood. The neighborhood is determined by a quadratic window and the weights of each value of the window are calculated by the Gaussian function of Eq. 4.11.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}} \tag{4.11}$$

where x and y are the position index of the window and σ is the cross product.

This approach can cause blur over silhouette edges and can cause plateaus of equal depth when using a large kernel.

4.2.2 Screen Space Curvature Flow

In order to achieve a better looking result, (8) propose a curvature flow approach, based on (73), which smoothes sudden changes in curvature between particles. As the viewpoint is constant, the smoothing effect can be applied by moving the depth value z proportionally to the curvature, as defined by equation 4.12.

$$\frac{\partial z}{\partial t} = H \tag{4.12}$$

where t is the smoothing time step and H is the mean curvature.

The mean curvature is defined as the divergence of the unit normal of a surface, as in equation 4.12.

$$2H = \nabla \cdot \widehat{\mathbf{n}} \tag{4.13}$$

A point P in view space V_x and V_y is mapped into a value in the depth map by inverting the projection transformation, as defined by equation 4.14.

$$\mathbf{P}(x,y) = \begin{pmatrix} (\frac{2x}{V_x} - 1)/F_x \\ (\frac{2y}{V_y} - 1)/F_y \\ 1 \end{pmatrix} z(x,y) = \begin{pmatrix} W_x \\ W_y \\ 1 \end{pmatrix} z(x,y) \tag{4.14}$$

where V_x and V_y are the dimensions of the viewport, F_x and F_y is the focal lenght in the x and y direction.

The normal of a point of the surface is calculated by the cross product between the derivatives of $\mathbf{P}$ in the x and y direction, as expressed by Eq. 4.15.

$$\mathbf{n}(x,y) = \frac{\partial \mathbf{P}}{\partial x} \times \frac{\partial \mathbf{P}}{\partial y} = \begin{pmatrix} C_x z + W_x \frac{\partial z}{\partial x} \\ W_y \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial x} \end{pmatrix} \times \begin{pmatrix} W_y \frac{\partial z}{\partial y} \\ C_y z + W_y \frac{\partial z}{\partial y} \\ \frac{\partial z}{\partial y} \end{pmatrix} \approx \begin{pmatrix} -C_y \frac{\partial z}{\partial x} \\ -C_x \frac{\partial z}{\partial y} \\ C_x C_y z \end{pmatrix} z \tag{4.15}$$

where $C_x = \frac{2}{V_x F_x}$, $C_y = \frac{2}{V_y F_y}$ and the terms W_x and W_y are ignored in order to simplify the computation as they have a small contribution on the calculation.

The unit normal is calculated by the expression of equation 4.16.

$$\widehat{\mathbf{n}} = \frac{\mathbf{n}(x,y)}{|\mathbf{n}(x,y)|} = \frac{(-C_y \frac{\partial z}{\partial x}, -C_x \frac{\partial z}{\partial x}, C_x C_y z)^T}{\sqrt{D}} \tag{4.16}$$

where $D = C_y^2 (\frac{\partial z}{\partial x})^2 + C_x^2 (\frac{\partial z}{\partial y})^2 + C_x^2 C_y^2 z^2$ and is substituted in equation 4.12 to express H in a way that can be derived.

The z component of the divergence is always zero, due the fact that z is a function of$x$ and y is kept constant when x and y are also maintained constant. So

$$2H = \frac{\partial \widehat{n}_x}{\partial x} + \frac{\partial \widehat{n}_y}{\partial y} = \frac{C_y E_x + C_x E_y}{D^{3/2}} \tag{4.17}$$

in which

$$E_x = \frac{1}{2} \frac{\partial z}{\partial x} \frac{\partial D}{\partial x} - \frac{\partial^2 z}{\partial x^2} D \tag{4.18}$$

,

$$E_y = \frac{1}{2} \frac{\partial z}{\partial y} \frac{\partial D}{\partial y} - \frac{\partial^2 z}{\partial y^2} D \tag{4.19}$$

A simple Euler integration of equation 4.12 in time is used to change the depth value in each iteration, and the derivatives of z are computed using finite differencing. To create a smoother surface, the number of iterations can be high, leading to a high computation time.

5

Implementation Methodology

The first step of this work was to implement a WCSPH simulation method in C++ language. Each particle of the system is a set of information into a struct such as: position, velocity, intermediate velocity, acceleration, mass, pressure, density, type of the particle, number of neighbors, index of the neighbors, kernel calculated and the kernel gradient calculated. The code of the particle struct can be seen below.

```
struct Particle3D{

        Vector3D <double> position;
        Vector3D <double> velocity;
        Vector3D <double> intVelocity;
        Vector3D <double> acceleration;

        double mass;
        double pressure;
        double density;

        int type; // 0 for fluid, 1 for board

        int neighbors[200]; //200 is the maximum number of
                                        //neighbors
        int numberNeighbors;
        double kernel[200];
        Vector3D<double> kernelGrad[200];

};
```

The continuity equation 3.20 was used to calculate the density of the particles, the Tait's equation 3.22 to calculate the pressure and the XSPH approach 3.33 to simulate the viscosity (2). In order to validate the method implementation, the numerical result will be compared with

the one found in the literature and will be illustrated in chapter 7 and to visualize the result the ParaView software was used, which can render the fluid particles as points in space as can be seen in Figure 5.1.

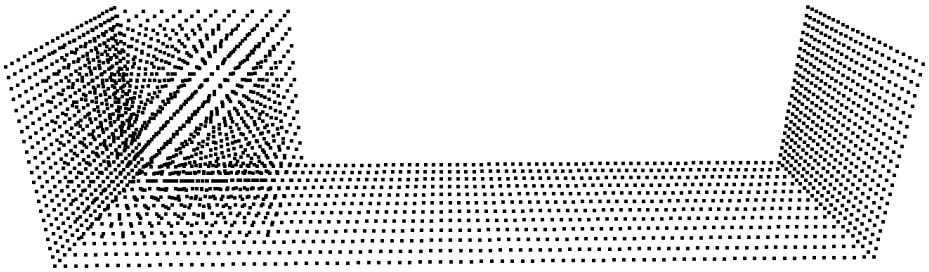


Figure 5.1: Visualization of the particles using the software ParaView.

In order to speedup the calculation, a parallel implementation was developed using CUDA technology without losing the numerical accuracy. In this solution, the calculations regarding each particle are done by a different thread in parallel.

The number of blocks used in a cuda kernel is a multiple of 512, which is the number of threads in a block.

In order to render the results from the simulation, the RT^2 was used to render each particle as a sphere to reconstruct the fluid surface. The RT^2 was also used to implement the smoothing algorithms, and even though the RT^2 is integrated into a single kernel to increase the performance, the implementation of the smoothing algorithms demanded a change of the integration into a multi-kernel solution.

The multi-kernel solution has 4 steps: (1) calculate the depth map of the scene, (2) smooth the depth map, (3) calculate the normal of the surface using the depth map information and (4) illuminate the scene using phong algorithm, as illustrated in Figure 5.2.

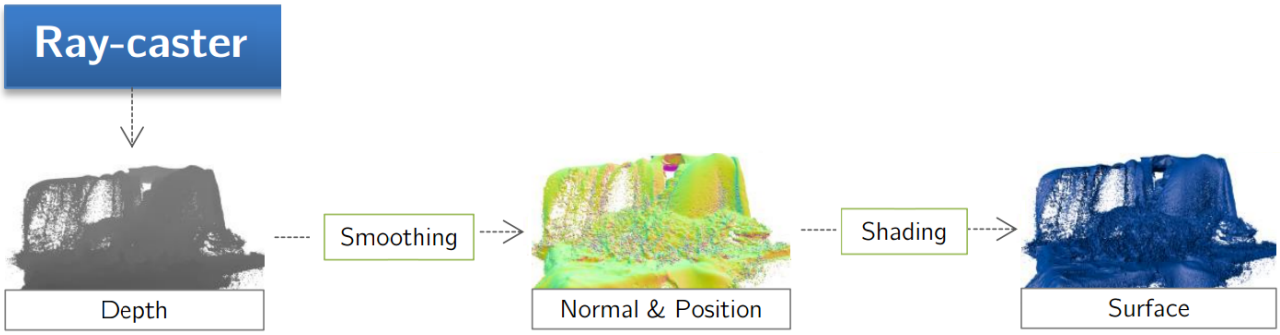


Figure 5.2: Multi-kernel solution steps. Adapted from (1).

And the final steps were to implement both smoothing algorithms discussed in the previous chapter and to analyze the visual results and the performances of them.

6

Case Studies

To analyze the results from the algorithms implemented, two different scenarios were used: the dam break and drop of water falling into an amount of fluid, which will be named water drop from now on.

A 2D dam break test case with 1600 fluid particles was used in order to validate the SPH implementation. This scenario is useful for testing numerical methods and solving flows that have a constant free-surface variation. In the test scenario, a dam was built as a 2D rectangular domain which burst right at the beginning of the simulation. The region on the right side of the dam does not contain any fluid. The fluid column has an initial height H of 0.6m and an initial width L of 0.6m. The bottom side of the domain has size of 2.4m. Figure 6.1 shows a 2D version of the dam break test scenario.

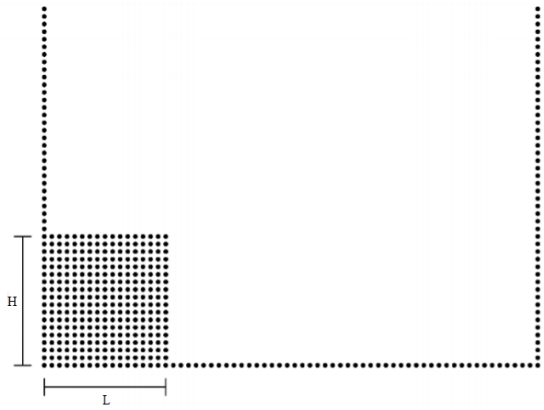
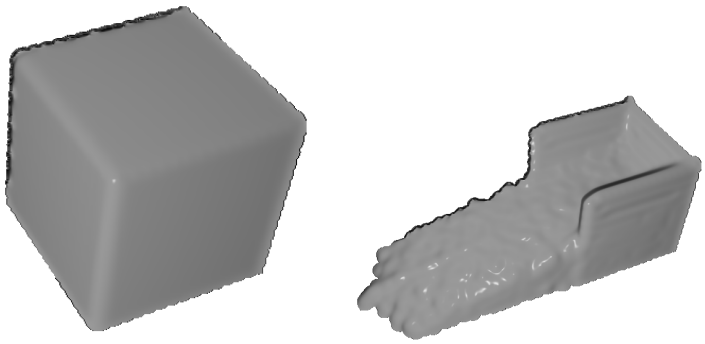


Figure 6.1: 2D dam break scheme (2).

A 3D dam break was also constructed. The dam was built as a 3D rectangular domain which burst left at the beginning of the simulation. The fluid column has H of 0.3m, L of 0.3m and an initial depth D of 0.3m. The bottom side of the domain has width size of 1.2m. Keyframes of this test case can be visualized in Figure 6.2.



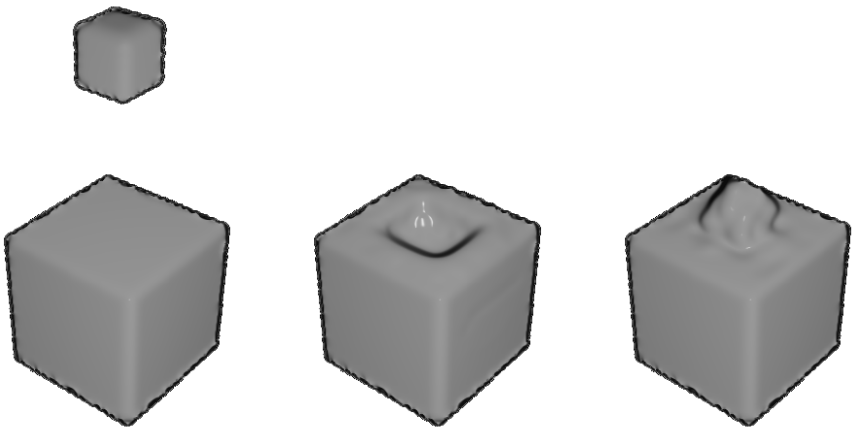
(a) Initial configuration of the 3D dam break

(b) Configuration of the dam break halfway the simulation

Figure 6.2: Keyframes of the 3D dam break test case.

Two different configurations were constructed for the 3D dam break, one with 1000 fluid particles and another with 3775 fluid particles, in order to analyze the influence of the particle number in the surface reconstruction result.

In the water drop scene, a 3D rectangular with 125 particles, initial height h of 0.1m, an initial width l of 0.1m and an initial depth d of 0.1m falls from a height of 0.6m and initial velocity of (0, -10, 0) m/s into a confined 3D rectangular with H of 0.3m, L of 0.3m and D of 0.3m. Keyframes of this test case can be visualized in Figure 6.3.



(a) Initial configuration of the water drop

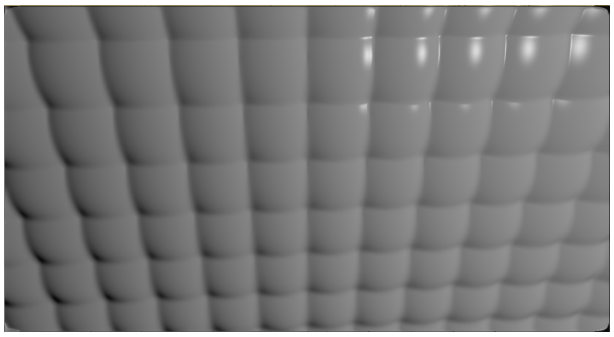
(b) First keyframe of the simulation

(c) Last keyframe of the simulation

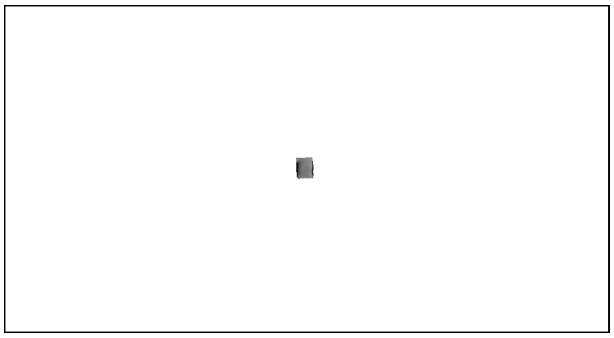
Figure 6.3: Keyframes of the water drop test case.

To analyze the performance of the algorithm, the camera moves through the scene from a location where the screen is completely filled with spheres and to another location where the scene cannot be distinguished, those positions can be seen in Figure 6.4 . While the camera moves through the scene, the frame rate in frames per second (fps) is tracked in order to get the maximum and minimum fps.

Since a ray tracing approach is being used is expected that the frame rate decreases as the camera goes away from the scene, given that there is less objects to be intersected.



(a) Screen is completely filled with spheres



(b) Far away distante view from the scene

Figure 6.4: Initial and final position of the test case.

7

Results and Analysis

7.1 Hardware and Software Infrastructure

The SPH method and the surface reconstruction algorithm were tested on a Intel(R) Core(TM) i7-4790K 4GHz with 32GB of memory and a 64-bit operating system. The GPU used was a GeForce GTX 960 with 2GB of memory and CUDA 7.5 installed.

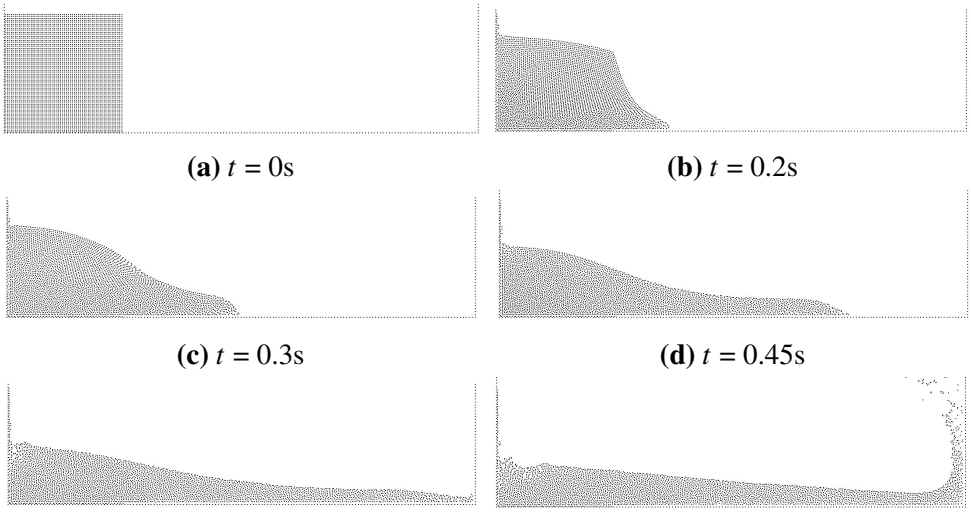
7.2 SPH Simulation Validation

The weakly compressible method WCSPH was implemented using a cubic kernel as smoothing function with kernel support of $1.3l_0$, where l_0 is the initial particles distance and using a time step equals to $5x10^{-4}$.

To evaluate the solutions obtained with the WCSPH, the 2D dam break scenario was used and the wave position right after when the dam bursts is tracked. The time is represented as a dimensionless value calculated by $t\sqrt{\frac{g}{H}}$, where t is the time in seconds and g is the gravity. The wave front position is expressed by the dimensionless expression as well. In order to validate the results from the simulations, the particle evolution chart is compared with the one found in (74).

The WCSPH method implemented and shown here has a close result to the referenced results, as it can be seen in Figure 7.2. The results found can be improved by using a more accurate viscosity approach than the artificial one and a different boundary condition, because the approach used in the simulation sticks the particles with low velocity to the boundary.

To simulate the fluid with high numerical precision, a small time step was used, leading to a very time consuming calculation.



(a) $t = 0$s **(b)** $t = 0.2$s

(c) $t = 0.3$s **(d)** $t = 0.45$s

(e) $t = 0.6$s **(f)** $t = 0.8$s

Figure 7.1: WCSPH visual result for different times t.

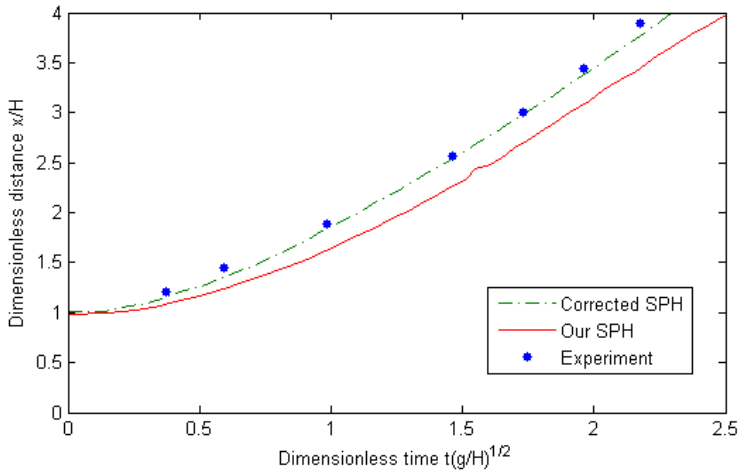


Figure 7.2: Evolution of the water wave front through dimensionless time using the WSCPH algorithm (2).

In the CPU based simulation, a single time step is calculated on average in 2.29 seconds and standard deviation of 0.39 seconds, whereas in the GPU based simulation, the calculation can be done on average in 0.67 seconds and standard deviation of 0.04 seconds. Leading to a speedup of 3.4 times, approximately.

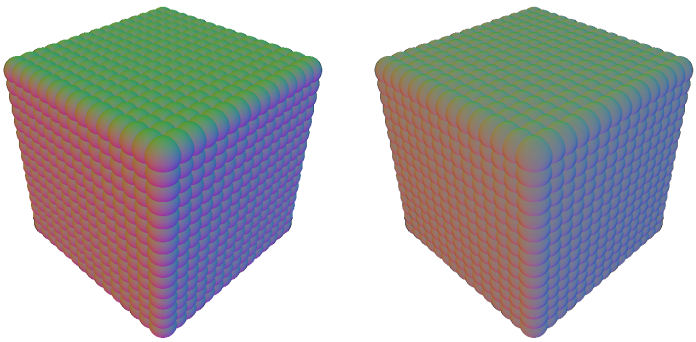
7.3 Surface Reconstruction

The surface reconstruction can be divided in three steps: (1) depth map creation, (2) smoothing the normal map and (3) calculate the surface normal. With the new surface, phong algorithm is used to illuminate the scene.

The normal map is created by taking the distance which is used to lead the ray from

the camera to the scene. After applying the smoothing algorithm, the map is more equally distributed.

The normal of a point of the smoothed surface is calculated by Eq. 4.16. To validate the calculation, the normal map of a scene with spheres in it, which can be calculated using the Eq. 4.10, is compared to the normal map using the approximated approach. The comparison proved that the approximated approach can reproduce correctly the normal behavior, as can be seen in Figure 7.3.



(a) Spherical normal calculation

(b) Approximated normal calculation

Figure 7.3: Normal calculation results comparison.

7.3.1 Dam break with 1000 particles

This test case, without the smooth surface, was rendered between 83 and 142 fps and the rendering result can be found in Figure 7.4.

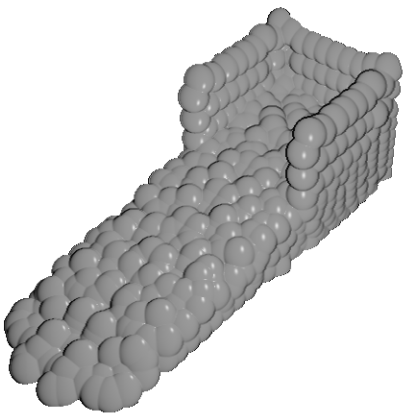
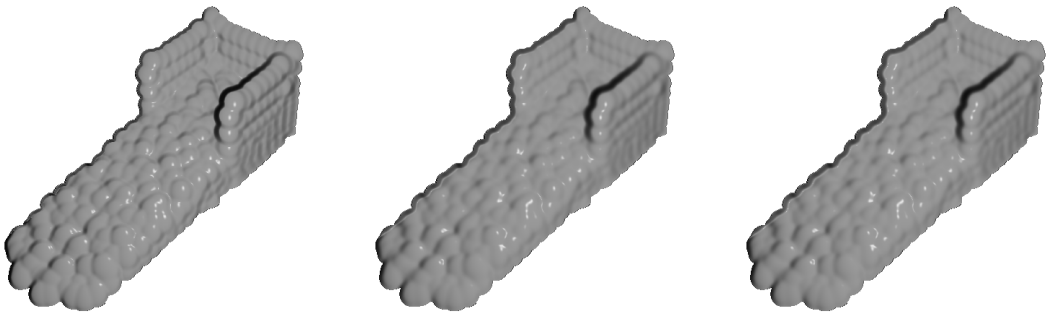


Figure 7.4: Dam break with 1000 particles without smoothing the surface.

If the Gaussian Blur is used to smooth the fluid surface, as the mask size increases, better is the visual result from the rendering but bigger is the time for rendering a single frame, as can been seen in Table 7.1 and the rendering result can be seen in Figure 7.5.



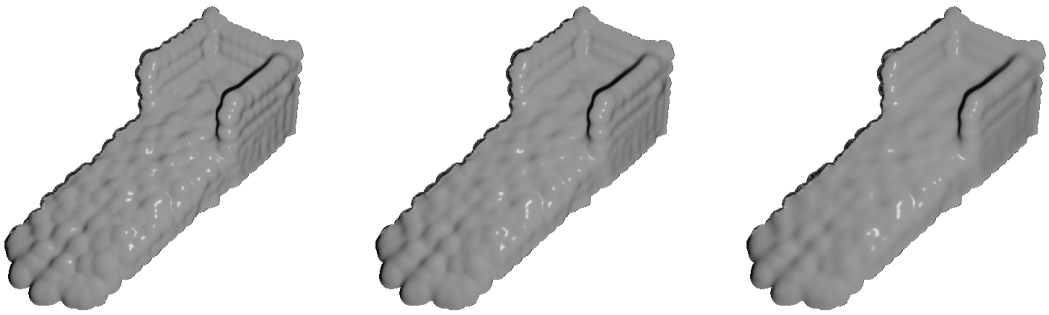
(a) Mask size = 7 **(b)** Mask size = 15 **(c)** Mask size = 31

Figure 7.5: Dam break with 1000 particles using Gaussian smoothing with different mask size.

Table 7.1: Performance of the Gaussian Blur method in frames per second (fps) in the Dam break with 1000 particles.

Mask size = 7	Mask size = 15	Mask size = 31
60 to 130 fps	25 to 50 fps	5 to 17 fps

Using the Screen Space Curvature Flow approach, it was possible to achieve a more realistic reconstruction, but with a small fps, if compared with the Gaussian approach, as can be seen in Figure 7.6.



(a) 40 iterations **(b)** 60 iterations **(c)** 100 iterations

Figure 7.6: Dam break with 1000 particles using Curvature Flow smoothing with different iterations number.

In order to achieve a better visual result, a bigger number of iterations is needed, which lead to a small fps, as can be seen in Table 7.2.

Table 7.2: Performance of the Screen Space Curvature Flow method in frames per second (fps) in the Dam break with 1000 particles.

40 iterations	60 iterations	100 iterations
12 to 15 fps	9 to 10 fps	5 to 7 fps

7.3.2 Dam break with 3375 particles

Without any blur in the surface, the ray tracer was able to reconstruct the fluid surface between 87 and 130 fps and the visual result can be found in Figure 7.7.

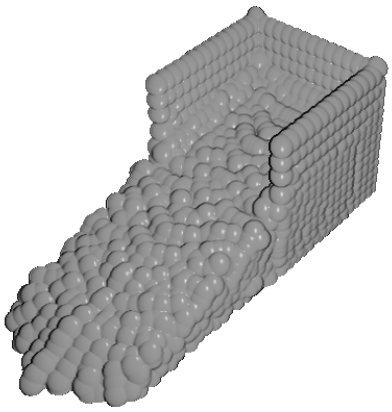


Figure 7.7: Dam break with 3375 particles without smoothing the surface.

By using both smoothing approaches, the relation between visual quality and performance was similar to the one found in the previous test case. The higher the reconstruction visual quality, the smaller is the performance of the approach, as can be seen in Table 7.3 for the Gaussian approach and in Table 7.4 for the Curvature Flow approach.

Table 7.3: Performance of the Gaussian Blur method in frames per second (fps) in the Dam break with 3375 particles.

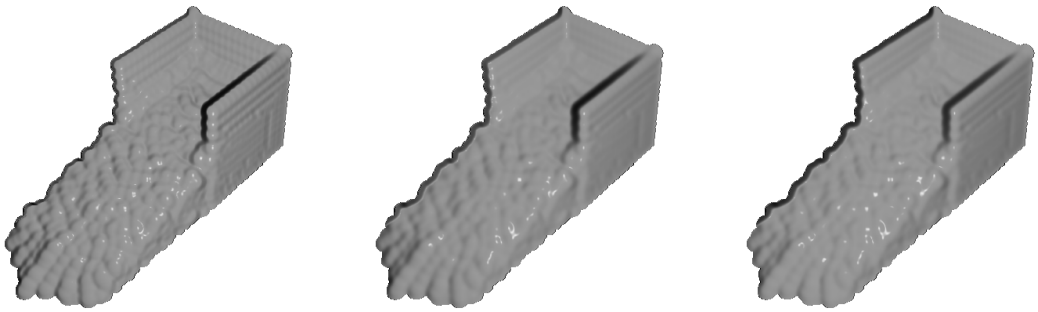
Mask size = 7	Mask size = 15	Mask size = 31
72 to 115 fps	28 to 45 fps	6 to 14 fps

Table 7.4: Performance of the Screen Space Curvature Flow method in frames per second (fps) in the Dam break with 3375 particles.

40 iterations	60 iterations	100 iterations
13 to 14 fps	9 to 10 fps	6 to 7 fps

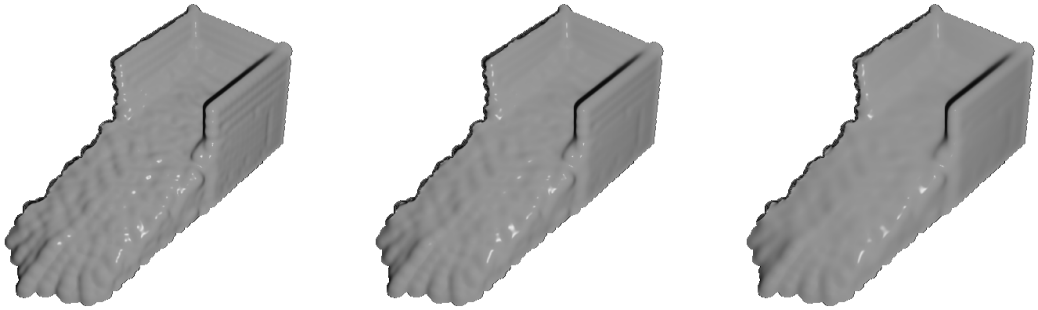
Comparing the visual quality of the results, with the results from the previous test, it was possible to achieve a better reconstruction quality due to the bigger number of particles in the same scene. But, just like the earlier test, the Curvature Flow approach had a better visual quality compared to the Gaussian approach, while the Gaussian is more suitable for applications that require better time performance.

The reconstruction results can be seen in Figure 7.8 and 7.9 using the Gaussian approach and the Curvature flow approach, respectively.



(a) Mask size = 7 **(b)** Mask size = 15 **(c)** Mask size = 31

Figure 7.8: Dam break with 3375 particles using Gaussian smoothing with different mask size.



(a) 40 iterations **(b)** 60 iterations **(c)** 100 iterations

Figure 7.9: Dam break with 3375 particles using Curvature Flow smoothing with different iterations number.

7.3.3 Water drop with 3500 particles

For the water drop, without the smooth surface, was possible to render with performance between 90 and 150 fps and the rendering result can be found in Figure 7.10.

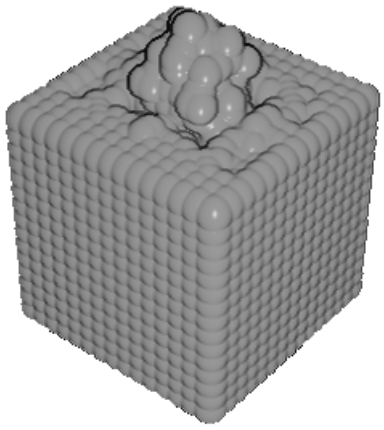


Figure 7.10: Drop water with 3500 particles without smoothing the surface.

By comparing the performance of both approaches in this test case, it was possible to visualize the same behavior of the two couple of tests done before, as can be seen in Table 7.5 and Table 7.6.

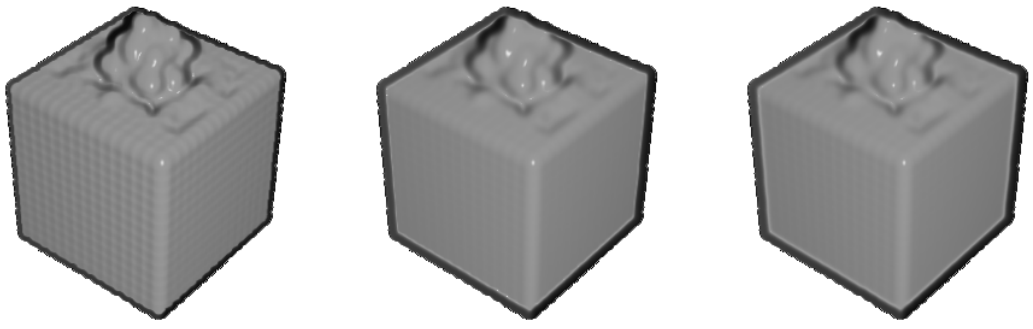
Table 7.5: Performance of the Gaussian Blur method in frames per second (fps) in the water drop with 3500 particles.

Mask size = 7	Mask size = 15	Mask size = 31
32 to 101 fps	30 to 50 fps	7 to 9 fps

Table 7.6: Performance of the Screen Space Curvature Flow method in frames per second (fps) in the water drop with 3500 particles.

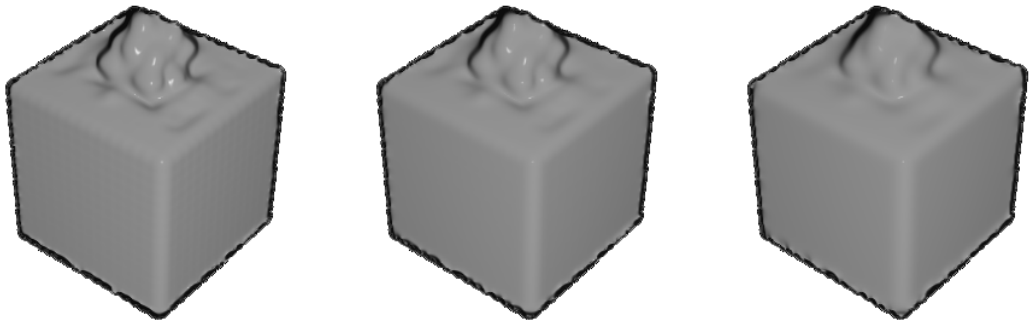
40 iterations	60 iterations	100 iterations
13 to 16 fps	9 to 11 fps	5 to 7 fps

By comparing the reconstruction result, the Curvature Flow approach had a better visual result compared to the Gaussian Blur. The Gaussian Blur is able to smooth the surface but still appear some imperfections in the surface, which are almost vanished using the Curvature Flow approach. The reconstruction result can be seen in Figure 7.11 and Figure 7.12 for the Gaussian and the Curvature Flow approaches, respectively.



(a) Mask size = 7 **(b)** Mask size = 15 **(c)** Mask size = 31

Figure 7.11: Drop Water 3500 particles using Gaussian smoothing with different mask size.



(a) 40 iterations **(b)** 60 iterations **(c)** 100 iterations

Figure 7.12: Drop Water 3500 particles using Curvature Flow smoothing with different iterations number.

In general, both methods were able to reconstruct the surface with a smoother look compared to the initial surface. Comparing the visual result, the Gaussian Blur method produces a less quality result because even with a large number of mask size it is still possible to differentiate the spheres in the scene and creates a dark region around the silhouettes of the surface. Whereas, the Screen Space Curvature Flow approach is able to decrease the dark region error and, by using a high number of iterations, the jelly look of the surface almost disappears.

In terms of performance, as expected, the frame rate increase as the scene occupies a bigger amount of the screen space. The Gaussian approach had a better result, being recommended to be used in real time applications. On the other hand, the Curvature Flow method is more suitable for applications which require a more realistic reconstruction without the necessity of being in real time.

8

Conclusion

This work has investigated the SPH method and ways of reconstructing the surface from the fluid simulation results using the Real Time Ray Tracer system. A WCSPH method was implemented being able to simulate the fluid behavior with numerical precision and a GPU version of the code was implemented achieving a speedup of 3x.

To reconstruct the fluid surface, two smoothing algorithms were implemented: the Gaussian Blur and the Screen Space Curvature Flow. It was possible to observe a trade-off between visual quality and performance. While the Gaussian Blur is more focused in performance, the Curvature Flow method can reconstruct the fluid surface with higher visual quality but is not recommended for real time applications.

Developing this work, one paper was published in the Congress on Numerical Methods in Engineering (CMN 2015) (2) and a book chapter regarding point based fluid simulation, to be published in 2016, was written while working alongside with the University of São Paulo (USP) entitled Applied Topics in Marine Hydrodynamics (75).

8.1 Future Work

As next steps for this work we intend to implement acceleration structures, like grid and octree, to speed up the neighborhood search of the SPH method in order to achieve a real time simulation and apply different methods of solid fluid coupling in order to create a more accurate simulation, such as (76) and (30). Also, improve the visual quality of the reconstruction by enforcing the boundary condition to prevent error around the silhouettes (8). To give a more realistic look to the fluid surface, there are several ways that can be followed, for instance, a thickness approach can be found in (8) and a volume based one can be found in (77).

References

[1] 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.

[2] 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," 2015.

[3] M. Liu and G. Liu, "Smoothed particle hydrodynamics (sph): an overview and recent developments," *Archives of computational methods in engineering*, vol. 17, no. 1, pp. 25–76, 2010.

[4] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics: theory and application to non-spherical stars," *Monthly notices of the royal astronomical society*, vol. 181, no. 3, pp. 375–389, 1977.

[5] 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*, vol. 26, no. 7, pp. 751–769, 1998.

[6] 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.

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

[8] 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.

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

[10] J. Yu and G. Turk, "Reconstructing surfaces of particle-based fluids using anisotropic kernels," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, p. 5, 2013.

[11] 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.

[12] R. Fraedrich, S. Auer, and R. Westermann, "Efficient high-quality volume rendering of sph data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 6, pp. 1533–1540, 2010.

[13] 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.

[14] J. Monaghan, "On the problem of penetration in particle methods," *Journal of Computational physics*, vol. 82, no. 1, pp. 1–15, 1989.

[15] J. J. Monaghan, "Sph without a tensile instability," *Journal of Computational Physics*, vol. 159, no. 2, pp. 290–311, 2000.

[16] A. L. dos Santos, V. Teichrieb, and J. Lindoso, "Review and comparative study of ray traversal algorithms on a modern gpu architecture," in *Proceedings of the 22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS Association*, pp. 203–212, Václav Skala-UNION Agency, 2014.

[17] V. Aurich and J. Weule, "Non-linear gaussian filters performing edge preserving diffusion," in *Mustererkennung 1995*, pp. 538–545, Springer, 1995.

[18] 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.

[19] 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.

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

[21] 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*, vol. 92, no. 4, pp. 343–369, 2012.

[22] 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*, vol. 89, no. 8, pp. 939–956, 2012.

[23] 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*, vol. 48, no. S1, pp. 50–60, 2010.

[24] 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*, vol. 227, no. 18, pp. 8417–8436, 2008.

[25] S. J. Cummins and M. Rudman, "An sph projection method," *Journal of computational physics*, vol. 152, no. 2, pp. 584–607, 1999.

[26] 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*, vol. 38, pp. 38–46, 2013.

[27] 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*, vol. 228, no. 18, pp. 6703–6725, 2009.

[28] D. L. Brown, R. Cortez, and M. L. Minion, "Accurate projection methods for the incompressible navier–stokes equations," *Journal of computational physics*, vol. 168, no. 2, pp. 464–499, 2001.

[29] 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*, vol. 2012, 2012.

[30] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner, "Versatile rigid-fluid coupling for incompressible sph," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 62, 2012.

[31] J. J. Monaghan and J. B. Kajtar, "Sph particle boundary forces for arbitrary boundaries," *Computer Physics Communications*, vol. 180, no. 10, pp. 1811–1820, 2009.

[32] 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*, vol. 42, no. 10, pp. 1210–1223, 2007.

[33] 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*, vol. 191, no. 2, pp. 622–638, 2003.

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

[35] D. C. Wilcox *et al.*, *Turbulence modeling for CFD*, vol. 2. DCW industries La Canada, CA, 1998.

[36] X.-j. Pan, H.-x. Zhang, and X.-y. Sun, "Numerical simulation of sloshing with large deforming free surface by mps-les method," *China Ocean Engineering*, vol. 26, pp. 653–668, 2012.

[37] J. P. Morris, P. J. Fox, and Y. Zhu, "Modeling low reynolds number incompressible flows using sph," *Journal of computational physics*, vol. 136, no. 1, pp. 214–226, 1997.

[38] S. Chantasiriwan, "Performance of multiquadric collocation method in solving lid-driven cavity flow problem with low reynolds number," *COMPUTER MODELING IN ENGINEERING AND SCIENCES*, vol. 15, no. 3, p. 137, 2006.

[39] D. J. Price, "Resolving high reynolds numbers in sph simulations of subsonic turbulence," *arXiv preprint arXiv:1111.1255*, 2011.

[40] M. Meister, G. Burger, and W. Rauch, "On the reynolds number sensitivity of smoothed particle hydrodynamics," *Journal of Hydraulic Research*, vol. 52, no. 6, pp. 824–835, 2014.

[41] B. Ataie-Ashtiani and L. Farhadi, "A stable moving-particle semi-implicit method for free surface flows," *Fluid Dynamics Research*, vol. 38, no. 4, pp. 241–256, 2006.

[42] J. Swegle, D. Hicks, and S. Attaway, "Smoothed particle hydrodynamics stability analysis," *Journal of computational physics*, vol. 116, no. 1, pp. 123–134, 1995.

[43] J. Bonet and S. Kulasegaram, "A simplified approach to enhance the performance of smooth particle hydrodynamics methods," *Applied Mathematics and Computation*, vol. 126, no. 2, pp. 133–155, 2002.

[44] J. Bonet and T.-S. Lok, "Variational and momentum preservation aspects of smooth particle hydrodynamic formulations," *Computer Methods in applied mechanics and engineering*, vol. 180, no. 1, pp. 97–115, 1999.

[45] C. Hori, H. Gotoh, H. Ikari, and A. Khayyer, "Gpu-acceleration for moving particle semi-implicit method," *Computers & Fluids*, vol. 51, no. 1, pp. 174–183, 2011.

[46] X. Zhu, L. Cheng, L. Lu, and B. Teng, "Implementation of the moving particle semi-implicit method on gpu," *SCIENCE CHINA Physics, Mechanics & Astronomy*, vol. 54, no. 3, pp. 523–532, 2011.

[47] 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*, vol. 6, no. 6, p. e20685, 2011.

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

[49] 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.

[50] J. F. Blinn, "A generalization of algebraic surface drawing," *ACM Transactions on Graphics (TOG)*, vol. 1, no. 3, pp. 235–256, 1982.

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

[52] 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.

[53] 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.

[54] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A hybrid particle level set method for improved interface capturing," *Journal of Computational physics*, vol. 183, no. 1, pp. 83–116, 2002.

[55] 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.

[56] J. Monaghan and A. Rafiee, "A simple sph algorithm for multi-fluid flow with high density ratios," *International Journal for Numerical Methods in Fluids*, vol. 71, no. 5, pp. 537–561, 2013.

[57] 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.

[58] 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.

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

[60] 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.

[61] J. J. Monaghan, "Smoothed particle hydrodynamics," *Reports on progress in physics*, vol. 68, no. 8, p. 1703, 2005.

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

[63] J. J. Monaghan, "Simulating free surface flows with sph," *Journal of computational physics*, vol. 110, no. 2, pp. 399–406, 1994.

[64] H. Schechter and R. Bridson, "Ghost sph for animating water," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 61, 2012.

[65] A. Appel, "Some techniques for shading machine renderings of solids," in *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pp. 37–45, ACM, 1968.

[66] T. Whitted, "An improved illumination model for shaded display," in *ACM Siggraph 2005 Courses*, p. 4, ACM, 2005.

[67] J.-H. Nah, J.-W. Kim, J. Park, W.-J. Lee, J.-S. Park, S.-Y. Jung, W.-C. Park, D. Manocha, and T.-D. Han, "Hart: A hybrid architecture for ray tracing animated scenes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 21, no. 3, pp. 389–401, 2015.

[68] J. Singh and P. Narayanan, "Real-time ray tracing of implicit surfaces on the gpu," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, pp. 261–272, March 2010.

[69] W.-J. Lee, Y. Shin, J. Lee, J.-W. Kim, J.-H. Nah, S. Jung, S. Lee, H.-S. Park, and T.-D. Han, "Sgrt: A mobile gpu architecture for real-time ray tracing," in *Proceedings of the 5th high-performance graphics conference*, pp. 109–119, ACM, 2013.

[70] X. Liu, Y. Deng, Y. Ni, and Z. Li, "Fasttree: a hardware kd-tree construction acceleration engine for real-time ray tracing," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1595–1598, EDA Consortium, 2015.

[71] NVIDIA, "Cuda zone | nvidia developer." https://developer.nvidia.com/cuda-zone. Accessed: 2016-01-09.

[72] T. Aila and S. Laine, "Understanding the efficiency of ray traversal on gpus," in *Proceedings of the conference on high performance graphics 2009*, pp. 145–149, ACM, 2009.

[73] R. Malladi and J. A. Sethian, "Level set methods for curvature flow, image enchancement, and shape recovery in medical images," in *Visualization and mathematics*, pp. 329–345, Springer, 1997.

[74] R. Staroszczyk, "Simulation of dam-break flow by a corrected smoothed particle hydrodynamics method," *Archives of Hydro-Engineering and Environmental Mechanics*, vol. 57, no. 1, pp. 61–79, 2010.

[75] "Meshless methods," in *Applied Topics in Marine Hydrodynamics* (G. Assi, H. Brinati, M. Conti, and M. Szajnbok, eds.), 2016.

[76] N. Akinci, G. Akinci, and M. Teschner, "Versatile surface tension and adhesion for sph fluids," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 182, 2013.

[77] J. Orthmann, M. Keller, and A. Kolb, "Topology-caching for dynamic particle volume raycasting.," in *VMV*, pp. 147–154, 2010.