Pós-Graduação em Ciência da Computação

CAIO JOSÉ DOS SANTOS BRITO

LARGE SCALE AND INTERACTIVE FLUID SIMULATION AND RENDERING USING THE SMOOTHED PARTICLE HYDRODYNAMICS TECHNIQUE ON GPU

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2018

Caio José dos Santos Brito

Large Scale and Interactive Fluid Simulation and Rendering Using the Smoothed Particle Hydrodynamics Technique on GPU

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

ADVISOR: **Veronica Teichrieb**
CO-ADVISOR: **João Marcelo Xavier Natário Teixeira**

RECIFE
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

B862l Brito, Caio José dos Santos
Large scale and interactive fluid simulation and rendering using the smoothed particle hydrodynamics technique on GPU / Caio José dos Santos Brito. – 2018.
79 f.: il., fig., tab.

Orientadora: Veronica Teichrieb.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências.

1. Visão computacional. 2. Realidade aumentada. I. Teichrieb, Veronica (orientadora). II. Título.

006.37 CDD (23. ed.) UFPE- MEI 2018-084

Caio José dos Santos Brito

Large Scale and Interactive Fluid Simulation and Rendering using the Smoothed Particle Hydrodynamics Technique on GPU

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação

Aprovado em: 28/02/2018.

BANCA EXAMINADORA

_____

Prof. Dr. Abel Guilhermino da Silva Filho
Centro de Informática/UFPE

_____

Prof. Dr. . Pierre Poulin
Département d'Informatique et de Recherche Opérationnelle
Université de Montréal

_____

Profa. Dra. Veronica Teichrieb
Centro de Informática / UFPE
(Orientadora)

Acknowledgements

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

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

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

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

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) which has the challenge of simulating fluid with different behaviors (multiphase, viscoelastic, viscoplastic, turbulent), and to render high visual quality results in real time. The main goals of this work are to investigate the following subjects: (a) the simulation of weakly compressible fluids using SPH for different behaviors, (b) the simulation of fluids in large scale and at interactive frame rates and (c) the rendering of fluids with balance between performance and visual quality. The implemented tool can simulate multiphase flow and viscoelastic fluids up to 1 million particles on GPU in interactive rates of 15 fps for $100k$ particles and 2 fps for $1M$ particles. Also, two rendering approaches were proposed: the first based on local illumination for multiphase fluids in real time and, to accomplish a more realistic result, an interactive solution based on Ray Tracing was proposed.

Keywords: SPH. Rendering. GPU.

Resumo

Simulação de fluidos sem a presença de malha tem se tornado uma maneira robusta de solucionar problemas com grandes deformações e tem sido cada vez mais utilizada em aplicações nas áreas de engenharia naval, engenharia mecânica, indústria cinematográfica e jogos. Um dos principais métodos sem malha é o Smoothed Particle Hydrodynamics (SPH) no qual tem o desafio de simular fluidos com diferentes propriedades (multifásico, viscoelástico, viscoplástico, turbulentos) e também apresentar os resultados da simulação com alto realismo visual em tempo real. O objetivo desse trabalho é investigar os seguintes assuntos: (a) a simulação de fluidos fracamente compressíveis utilizando o método SPH para simular diversos comportamentos, (b) a simulação de fluidos em grande escala e em taxas computacionais interativas e (3) a renderização do fluido buscando o equilíbrio entre desempenho e qualidade visual. A ferramenta desenvolvida é capaz de simular fluidos viscoelásticos e fluxos multifásicos com até 1 milhão de partículas na GPU em taxas interativas de 15 fps para $100k$ partículas e 2 fps para $1M$ de partículas. E também, duas propostas de renderização foram desenvolvidas: a primeira baseada em iluminação local para fluidos multifásicos em tempo real e, para obtenção de resultados mais realistas, uma solução interativa baseada em Ray Tracing foi proposta.

Palavras-chave: SPH. Renderização. GPU.

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 boundary regions (LIU; LIU, 2010).

Although more accurate and well consolidated, the conventional methods (FEM and Finite Difference Method (FDM)) have some problems when dealing with such required deformations, as those methods rely on meshes, which cannot handle moving discontinuities efficiently. According to (BELYTSCHKO et al., 1996), the best approach for mesh-based methods dealing with this kind of problem is remeshing every iteration step of the simulation to keep the mesh discontinuities coincident through the entire simulation. The work of Franci and Cremonesi (FRANCI; CREMONESI, 2017) presents the relation between a coarse and a fine mesh as can be seen in Fig. 1.1 (left), which shows that a fine mesh provides more precise results, but the process of remeshing is still necessary to solve the volume loss as can be seen in Fig. 1.1 (right), which is quite time-consuming as it gets linearly more complex depending on the number of nodes.

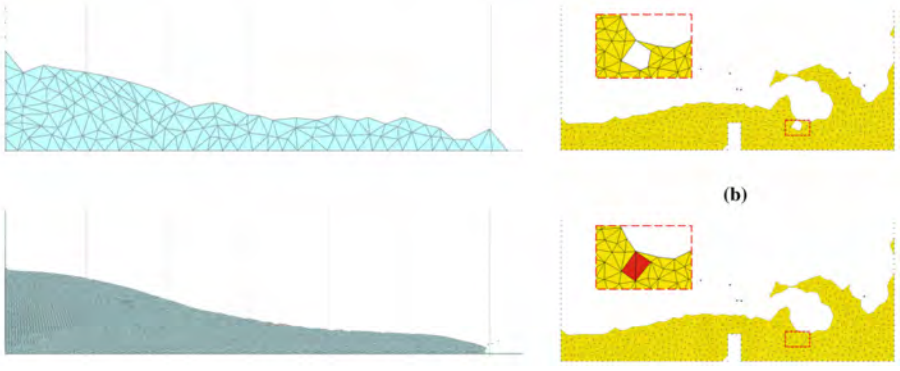

Figure 1.1: Comparison between coarse and fine meshes (FRANCI; CREMONESI, 2017).

1.1 Fluid Simulation

To overcome such challenges, mesh-free methods may be used such as the Smoothed Particle Hydrodynamics (SPH) (GINGOLD; MONAGHAN, 1977) and Moving Particles Semi-Implicit (MPS) methods (KOSHIZUKA; NOBE; OKA, 1998). 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 mesh-based methods become very complex (ROGERS et al., 2003).

As can be seen in the work of Marrone et al. (MARRONE et al., 2011), the SPH method can simulate a violent impact without volume loss and with no need to create any mesh, as illustrated in Fig. 1.2.

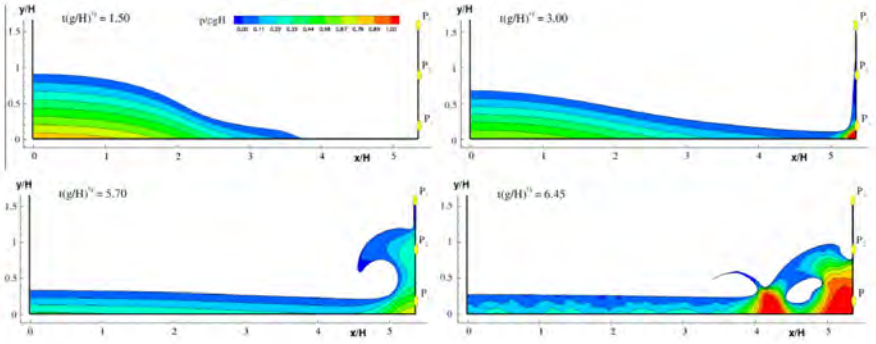

Figure 1.2: SPH simulation in a violent impact (MARRONE et al., 2011).

According to Violeau and Rogers (VIOLEAU; ROGERS, 2016), 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 visual accuracy (LIU; LIU, 2010).

Vieira-e-Silva, Brito et al. (SILVA et al., 2015) evaluated a weakly compressible SPH method that relies only on the XSPH formulation to simulate viscosity and prevents the particle penetration problem (boundary condition). This approach led to a small SPH formulation (pressure force calculation and XSPH) and to a relatively high numerical precision as can be seen in Fig. 1.3. So, this method can be used for interactive applications due to its small number of calculations and easy control of the viscosity with the XSPH method, leading to an easily tunable method with fast and small number of computations.

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 (VIOLEAU; ROGERS, 2016), 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 (SOLENTHALER; PAJAROLA, 2008) presented a multiphase SPH formulation which does not

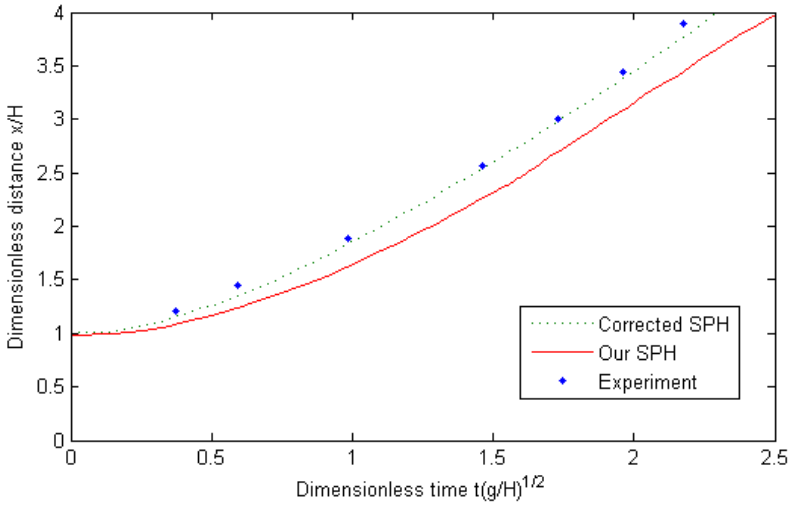

Figure 1.3: Evolution of the water wave front through dimensionless time using the WCSPH algorithm (SILVA et al., 2015).

affect the performance negatively and is easy to implement due to its simple modifications to the standard SPH method. An example of multiphase flow simulation can be seen in Fig. 1.4, in which the density ratio is 1, 10, and 100; as density ratio gets higher, the fluid with higher density tends to push the fluid with lower density.

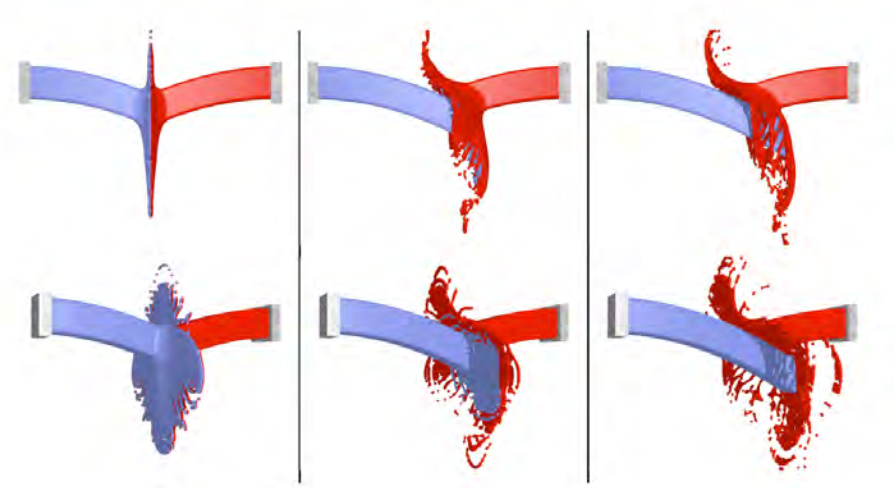

Figure 1.4: Multiphase flow with different density ratio: 1 (left), 10 (middle) and 100 (right) (SOLENTHALER; PAJAROLA, 2008).

Fluid simulation is quite common in the game industry, for instance, the Unity game engine has its own mesh-based water simulator (UNITY TECHNOLOGIES, 2018), and the Unreal engine uses a particle system to simulate dynamic fluids (EPIC GAMES, 2018a,b). There are also games that prefer to create their own fluid simulator as in PixelJunk Shooter 2, which has its own fluid engine to simulate fluids with different densities (water and lava) and thermal dynamics between fluids (KESSLER; CARABAICH; KINOSHITA, 2011). Those methods can generate high-quality results but are numerically far from reality, creating a non-realistic

experience for the user.

In the cinematography industry, particle-based methods are very common. The Houdini-FX (SIDE EFFECTS SOFTWARE, 2018a) software simulates water and viscous fluids using the Fluid Implicit Particle Method (FLIP) (SIDE EFFECTS SOFTWARE, 2018b), that is a faster method but not numerically accurate. The RealFlow (NEXT LIMIT TECHNOLOGIES, 2018a) provides two fluid simulation methods: the Position Based Dynamics Method (PBD), which is a faster solution (NEXT LIMIT TECHNOLOGIES, 2018b), and the SPH, which is slow but very accurate (NEXT LIMIT TECHNOLOGIES, 2018c).

The SPH method has been used in the gaming industry (LAAN; GREEN; SAINZ, 2009) and in the film industry (HORVATH; SOLENTHALER, 2013). The method has become one of the most popular particle-based methods in the animation industry (YAN et al., 2016) due to its high flexibility of implementation and numerical precision, which provides a high quality experience for the user but has the challenge of providing a solution in real time for a large number of particles (HORVATH; SOLENTHALER, 2013).

In this industry, it is very common to simulate viscoelastic materials, for example, egg white, gels, and slime. They produce appealing visual effects, so the video games and film industries quite often require accurate simulation of these viscoelastic properties. Exaggerated representations that real-world materials do not exhibit, like very large deformations, are frequently required by such industries to make certain characters or effects cause a greater impression on the audience. Several examples of viscoelastic simulations can be seen in Fig 1.5.

Figure 1.5: Examples of viscoelastic fluids in games: Starcraft II (top left), Phineas and Ferb (top right), Goop (bottom left), and World of Goo (bottom right).

The work of Takahashi et al. (TAKAHASHI et al., 2016) proposes a particle-based hybrid method for simulating volume preserving viscoelastic fluids with large deformations. It combines SPH and Position-based Dynamics, the latter proposed by Müller et al. (MÜLLER et al., 2007), to approximate the dynamics of viscoelastic fluids, where the idea of adaptive connections between particles is used to correct particle velocities, which are carefully calculated

to not negatively affect volume preservation of materials. The authors claim that examples show the proposed hybrid method can sufficiently preserve fluid volumes and robustly generate a variety of viscoelastic behaviors, such as splitting and merging large deformations, and Barus effect as can be seen in Fig. 1.6. Despite the visual quality of Takahashi et al.'s work, the method takes an average of $10s/\text{step}$ in a simulation with 111k particles.

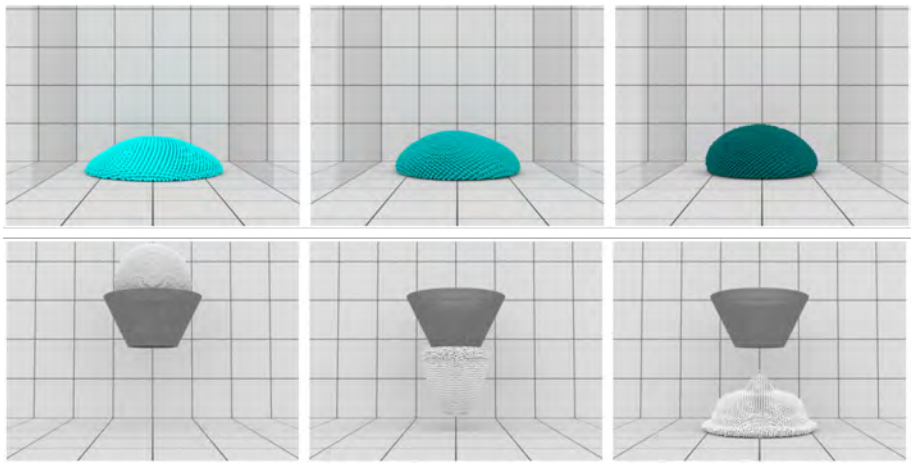

Figure 1.6: Results from the work of (TAKAHASHI et al., 2016).

1.2 Fluid Rendering

In the gaming, film, and animation industries, another big challenge is to realistically render the simulated fluid. To visualize simulation results, it is necessary to rebuild the surface using the particles identified as free surface (YU; TURK, 2013). The rendering of the results can be made using several methods such as Direct Rendering (YU et al., 2012), 3D Scalar Field (YU; TURK, 2013), Volume Rendering (FRAEDRICH; AUER; WESTERMANN, 2010) or a Screen Space Approach (LAAN; GREEN; SAINZ, 2009).

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 (LORENSEN; CLINE, 1987). The second approach renders the particles as spheres or point sprites, and applies a smoothing filter to the depth map to create a better looking final surface render.

The 3D Scalar Field approach faces 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 (YU; TURK, 2013). The results from the rendering can be seen in Fig. 1.7.

The screen space option may be more suitable for real-time applications because each particle is rendered individually, without the need to apply a function on neighboring particles.

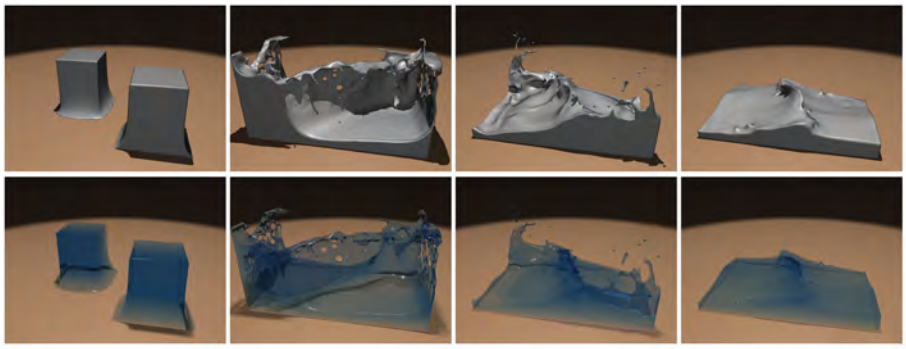

Figure 1.7: Rendering of a double dam break using the 3D Scalar Field technique proposed by Yu and Turk (YU; TURK, 2013).

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. An overview of the method proposed by Reichl et al. (REICHL et al., 2014) can be seen in the Fig. 1.8.

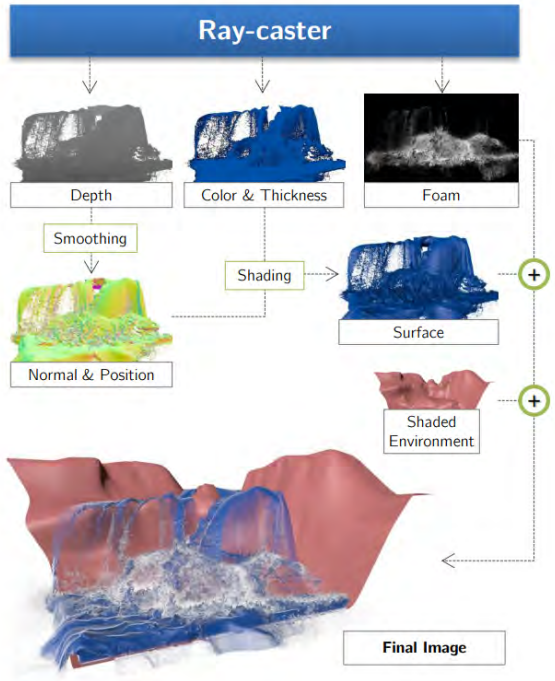

Figure 1.8: Overview of the method proposed by Reichl et al. (REICHL et al., 2014).

This type of technique has two main challenges: finding the best function to create a smooth surface (REICHL et al., 2014) and creating a realistic illumination on the fluid surface (XIAO; ZHANG; YANG, 2017). Despite the recent efforts to create a high-quality rendering in real-time, those methods are still unable to provide a solution that is suitable for any method. For instance, the method of Xiao et al. (XIAO; ZHANG; YANG, 2017) relies on the particles mass to reconstruct the surface and needs to be modified to work on massless methods such as the Moving Particle Semi-implicit and the screen space methods still are not integrated into a global

illumination algorithm.

1.3 Goals

The main goals of this work are to investigate the following subjects: (a) the simulation of weakly compressible fluids using the SPH for different behaviors, (b) the simulation of large-scale fluids in interactive rates, and (c) the rendering of fluids with a balance between computational performance and visual quality.

The SPH method used in this work was the one developed by Vieira-e-Silva, Brito et al. (SILVA et al., 2015) that relies only on XSPH formulation to simulate viscosity and prevent the 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.

The SPH method developed by Vieira-e-Silva et al. was extended to simulate multiphase flow using the density formulation proposed by Solenthaler and Pajarola (SOLENTHALER; PAJAROLA, 2008) and depends only on the XSPH to handle viscosity and boundary conditions. The SPH method was also extended to simulate viscoelastic fluids using the formulation proposed in the work of Takahashi et al. (TAKAHASHI et al., 2016).

To achieve even higher frame rates, aiming real-time simulation, NVIDIA's CUDA (NVIDIA, 2013) was used to accelerate the simulation generation, as well as OpenMP (OPENMP ARCHITECTURE REVIEW BOARD, 2017) to explore the parallelism provided by multiple CPU cores. A parallelized CPU version using OpenMP and a parallelized GPU version using CUDA were developed so the simulation could achieve its maximum performance with a large number of particles, reaching interactive rates. Both versions developed were compared regarding performance.

To visualize the fluid simulation two rendering approaches were proposed. First, to be able to visualize the multiphase flow, a graphical pipeline for real-time multiphase particle-based simulation based on the work of van der Laan et al. (LAAN; GREEN; SAINZ, 2009) was implemented and, looking for high-visual quality, a Ray Tracing based rendering solution was proposed using the NVIDIA OptiX Ray Tracing engine (PARKER et al., 2010).

1.4 Organization

The next chapter discusses the state of the art of SPH-based fluid simulations focusing on multiphase flow and viscoelastic flow and also on how to render results from the simulation. Chapter 3 describes the SPH technique, its governing equations and modifications on the method to simulate multiphase and viscoelastic flow, also, this chapter describes the methods used to render the simulation results. After that, Chapter 4 details the implementation and the tools used in this work. Chapter 5 describes the test cases used in this work and how to analyze

the results. The Chapter 6 discusses the results regarding numerical validations of the SPH method developed, GPU speedup, the visual quality of the rendering and performance. At last, in Chapter 7, conclusions are discussed, and the contributions of this work are exposed, with future possibilities and enhancements being explored.

2

STATE OF THE ART

In this chapter, the state of the art on the SPH method is exposed focusing on viscoelastic flow, multiphase flow, and fluid rendering for particle-based simulations.

2.1 SPH Method

The SPH method was introduced by (LUCY, 1977) and (GINGOLD; MONAGHAN, 1977) to model astrophysical phenomena. Since then it has been vastly extended to model fluids (CHEN; YANG; YUAN, 2009), (ANDREA, 2005) and even solids behavior (CHEN; LEE; ESKANDARIAN, 2006), mainly focusing on those aspects that could limit the applications simulated by mesh-based approaches, such as large deformations, for instance, as can be seen in Fig 1.1.

A straightforward adaptation of the original SPH method is the application for weakly compressible fluids. In this kind of fluid, pressure can be calculated by an equation of state. The works (SZEWC; POZORSKI; MINIER, 2012), (SHADLOO et al., 2012), (LEE et al., 2010) and (LEE et al., 2008) show comparisons between implementations of weakly compressible (WCSPH) and truly incompressible methods (ISPH), in which are applied techniques such as the one introduced by (CUMMINS; RUDMAN, 1999).

Shadloo et al. (SHADLOO et al., 2012) state that, in comparison with the ISPH, the WCSPH is easier to program and has a better-ordered particle distribution. Due to these reasons WCSPH has become the most common method to solve the linear momentum balance equation using SPH. However, ISPH has a better stability to solve turbulent fluid flows with high Reynolds number, while WCSPH suffers from large density variations being more suitable for flows with low turbulence. The ISPH method also provides a more accurate pressure field calculation but with the high cost of solving a linear system, which can be quite difficult to solve for a massive number of particles. For that reason, Violeau and Rogers (VIOLEAU; ROGERS, 2016) state that in the future, WCSPH will need to be supplemented with a more robust formulation to be able to achieve the precision of the ISPH.

In truly incompressible methods, density is calculated by the Poisson equation. This equation can be represented by a sparse linear system. Those methods can generate a more accurate

solution but require more computation time as stated in the works of (GHASEMI V; FIROOZ-ABADI; MAHDINIA, 2013), (XU; STANSBY; LAURENCE, 2009), (BROWN; CORTEZ; MINION, 2001), and (ASAI et al., 2012).

The main difference between the original (astrophysical) SPH method and more recent particle-based fluid simulations is the inclusion of boundary conditions. Works like (HARADA; KOSHIZUKA; KAWAGUCHI, 2007a), (TANAKA; MASUNAGA, 2010), (TSURUTA; KHAYYER; GOTOH, 2013), (LASTIWKA; BASA; QUINLAN, 2009), and (MONAGHAN; KAJTAR, 2009) propose efficient ways for dealing with them. Another kind of boundary that may suffer from instability problems is the interface between two or more different fluids in multiphase scenarios. The works from (HU; ADAMS, 2006) and (ZAINALI et al., 2013) illustrate how this issue could be handled.

Besides the compressibility factor present in fluids and boundary conditions, some other elements can be introduced to the fluid behavior, depending on the kind of problems being studied, like viscosity, presence or absence of turbulence, type of smoothing function, among others. Some works present results for adjustments in viscosity according to the problem being focused on (POZORSKI; WAWREŃCZUK, 2002), (WATKINS et al., 1996), (RAFIEE; MANZARI; HOSSEINI, 2007), (SIGALOTTI et al., 2003), (YANG; LIU; PENG, 2014).

The most common way of simulating a turbulent flow is to incorporate the Reynolds-averaged Navier-Stokes turbulence (RANS) model into the SPH method (WILCOX et al., 1998). It is worth noticing that the majority of the meshless works found in the literature deal with a low Reynolds' number such as the works of (PAN; ZHANG; SUN, 2012), (MORRIS; FOX; ZHU, 1997), (CHANTASIRIWAN, 2006), (SIGALOTTI et al., 2003), (PRICE, 2011) and (MEISTER; BURGER; RAUCH, 2014). This latter work discusses the common usage of this value in the literature.

The smoothing function is an important choice on the SPH method because it models how particles interact with each other depending on how close they are. To explain the effects of the smoothing function on meshless simulations, the works (SIGALOTTI et al., 2003), (ATAIE-ASHTIANI; FARHADI, 2006), (SWEGLE; HICKS; ATTAWAY, 1995), (BONET; KULASEGARAM, 2002), (BONET; LOK, 1999), and (LIU; LIU, 2010) discuss the changes in behavior when varying the smoothing functions, evaluating the accuracy and stability of the methods.

Unfortunately, similarly to the other meshfree methods, the SPH technique suffers from instability problems. Some (the main problems with particle-based methods) are related to numerical errors at the boundaries, i.e., at free-surfaces or when interacting with solid boundaries as can be seen in Fig. 2.1, which illustrates the pressure profile of the SPH method in comparison to the theoretical result. The works of (MONAGHAN, 1994), (KONDO et al., 2008), (LEE et al., 2011), (CRESPO, 2008), (FANG et al., 2009) and (KIARA; HENDRICKSON; YUE, 2013) describe why those instability problems arise for each specific particle method. Some adaptations, which help to attenuate the effects of instability or even increase the accuracy of

the simulation have been implemented in the works of (BØCKMANN; SHIPILOVA; SKEIE, 2012), (BØCKMANN; SHIPILOVA; SKEIE, 2012), (XU; STANSBY; LAURENCE, 2009), (HOSSEINI; FENG, 2011), (DEHNEN; ALY, 2012), and (TANAKA; MASUNAGA, 2010).

Another common problem is the one related to the interaction between particles, more precisely, when they get too close to each other, generating repulsive stress, which results in an instability known as tensile instability (MONAGHAN, 2000). Works like (YANG; LIU; PENG, 2014) try to overcome this problem using other adaptations to the methods.

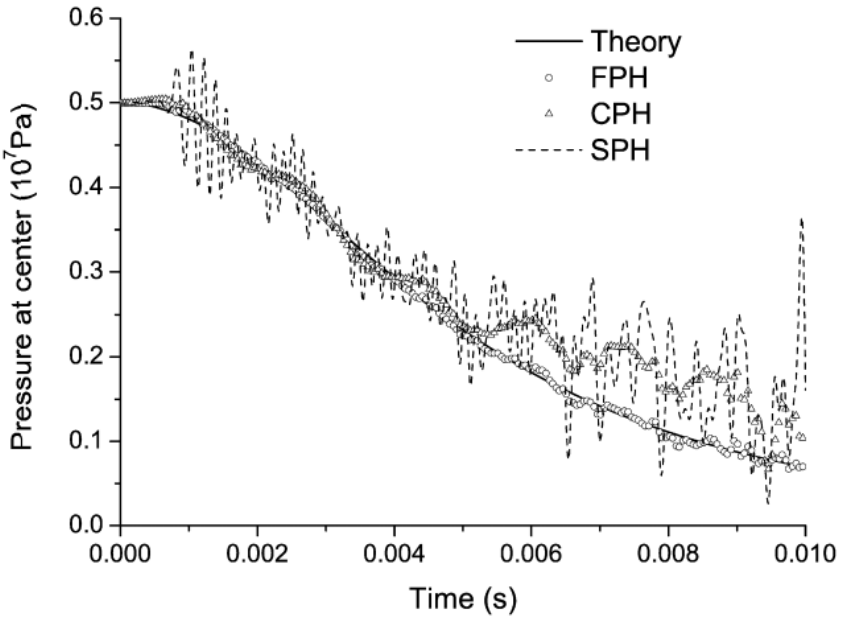

Figure 2.1: Comparison between the pressure profile of SPH with the theoretical expected result.

Given the meshless characteristic of the simulation, it is possible to create a parallel solution, using cluster technology or general purpose programming for graphics processor (GPGPU) techniques, to decrease time consumption, as shown in the works of (HORI et al., 2011), (ZHU et al., 2011), (CRESPO et al., 2011), (HARADA; KOSHIZUKA; KAWAGUCHI, 2007b), and (KROG; ELSTER, 2012).

In the virtual reality (VR) community, SPH simulation has been used in many applications. Cirio et al. (CIRIO et al., 2011) proposed a six Degrees of Freedom (DoF) haptic interaction with fluid simulated using WCSPH, providing a force-based feedback and being able to achieve real-time performance (90 fps) for 32,768 particles, which is sufficient to simulate the behavior of a pancake dough or a fluid on a small bowl. This work was extended to deal with melting and freezing phenomena (CIRIO et al., 2013) and to interact with deformable and rigid bodies (CIRIO et al., 2011). An example of application can be seen in Fig. 2.2.

Another example addressed to the VR community is the work of Pang et al. (PANG et al., 2010), which uses a PhysX built-in SPH-based fluid solver to simulate bleeding effects in a VR-based surgical simulator, being able to achieve a performance of 49 fps with 5,000 particles.

Figure 2.2: Result from the work of Cirio et al. (CIRIO et al., 2011), which applies SPH simulation to cook a pancake.

Also, Wang and Wang (WANG; WANG, 2014) proposed a haptic interaction with the fluid using SPH and FEM to operate a canoe with two paddles rowing in the fluid.

As for the gaming industry, several works that use the SPH method focus on simulating fluids for interactive applications and real-time ones, starting with the work of Müller et al. (MÜLLER; CHARYPAR; GROSS, 2003). Other works focus on simulating fluids with different properties and features, so it is possible to represent most fluid types and behaviors. Performance needs to be the focus if the application will be utilized for gaming purposes, so the aim should always be a simulation running at least near to real time; however, the fluid also needs to preserve its physical properties and present them as coherently as possible (JUNIOR et al., 2010) (JUNIOR et al., 2012).

2.1.1 Viscoelastic Flow

Works involving the SPH method augmented with viscoelastic formulations are quite recent. Still, lots of works are already benefiting from each other. The visual appeal and the high range of applicability of this type of simulation in fields such as medicine, biology and the entertainment industry (XU; YU, 2016; YEH; FALOUTSOS; REINMAN, 2006) may be the reason of its instant popularity. In this section, some similar works to this one are presented, showing its importance to the community.

Clavet et al. (CLAVET; BEAUDOIN; POULIN, 2005) took advantage of the method proposed by Müller and Pearce (MILLER; PEARCE, 1989) and Terzopoulos et al. (TERZOPOULOS; PLATT; FLEISCHER, 1991), which is a spring-based method, and combined it with SPH to simulate materials with elasticity, plasticity, and viscosity, adopting a prediction-relaxation scheme. This spring-based model computes attraction and repulsion forces between particles

to successfully simulate the viscoelastic properties of certain materials. Another similar spring-based method was also proposed by Takamatsu and Kanai (TAKAMATSU; KANAI, 2011), who used Position Based Dynamics to simulate fluids with viscosity and elasticity in a unified framework that can be seen in Fig. 1.6.

Müller et al. (MÜLLER et al., 2004) proposed the addition of an elasticity term to the formulations that use Moving Least Square (MLS) to simulate elastoplastic objects. Solenthaler et al. (SOLENTHALER; SCHLÄFLI; PAJAROLA, 2007) adopted the formulation of this elasticity term and computed it using SPH instead of MLS to allow for robustly simulating fluid with various properties under some conditions. The method that Solenthaler et al. (SOLENTHALER; SCHLÄFLI; PAJAROLA, 2007) proposed was extended to handle rotational motions of elastic materials (BECKER; IHMSEN; TESCHNER, 2009). Mao and Yang (MAO; YANG, 2006) introduced a viscoelastic force term into the Navier-Stokes equations to simulate viscoelastic fluids.

Xu and Yu (XU; YU, 2016) proposed a viscoelastic SPH to be used in biological applications, more specifically a multiscale SPH method to simulate transient viscoelastic flows by using a bead-spring chain description of the polymer molecule. The authors came up with a methodology that coupled macroscopic conservation equations for mass and momentum with a differential equation for bead-spring chain dynamics, which, when solved, obtains the polymeric stress.

Xu et al. (XU; DENG, 2016) proposed an improved WCSPH method to simulate transient-free surface flows of viscous and viscoelastic fluids. The improvement to the WCSPH formulations includes a greater accuracy and stability due to a correction in the kernel gradient calculation and an enhanced computation of pressure distribution in the dynamics of the fluid due to corrections in the continuity equation. The effectiveness of the method is successfully proved through a series of test scenarios common in the literature, like a dam breaking flow, stretching of a water drop, and a viscoelastic fluid dropped against a wall, which can be seen in Fig 2.3.

Heck et al. (HECK et al., 2017) also proposed a viscoelastic SPH applied to biology, although this time to model extracellular matrix viscoelasticity for an extracellular matrix in contact with a migrating cell. This method improves contact mechanics by modeling it based on an existing boundary method in SPH, which is extended to allow the modeling of moving boundaries in contact with a viscoelastic solid. This result should enable the field researchers to model and understand realistic cell-matrix interactions in the future.

2.1.2 Multiphase Flow

The multiphase flow was firstly introduced by Monaghan and Kocharyan (MONAGHAN; KOCHARYAN, 1995), which provided a general and easily extended SPH method to handle multiphase air simulation. Müller et al. (MÜLLER et al., 2005) were able to simulate multiple fluids with small density ratios by changing the mass and the rest density. Hu and Adams

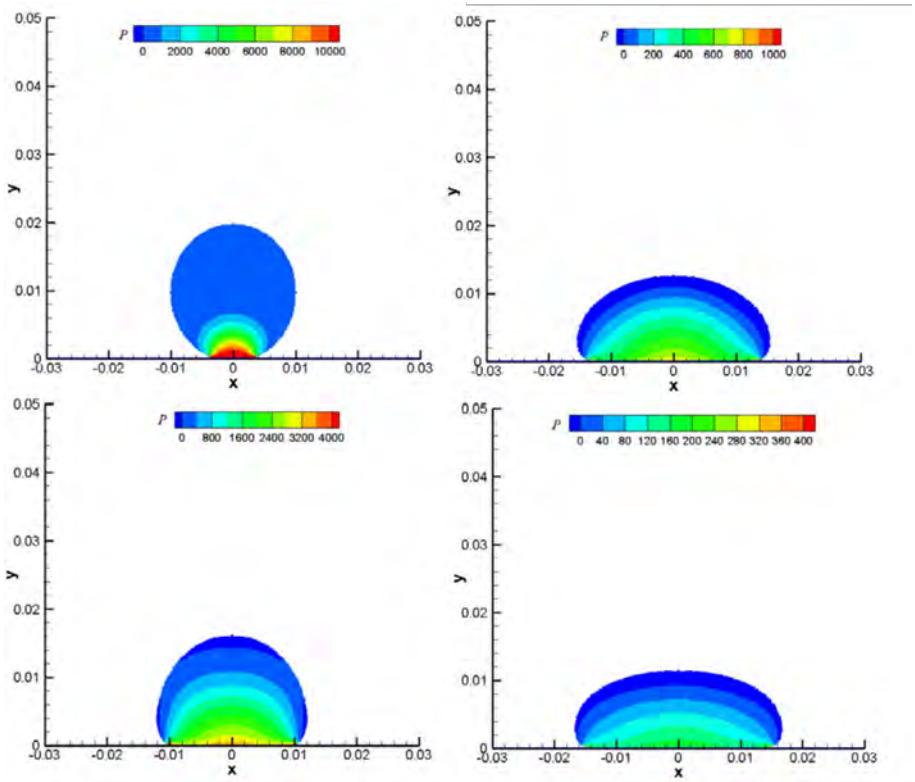

Figure 2.3: Results from the work of Xu and Deng (XU; DENG, 2016) for a viscoelastic fluid dropped against a wall in time.

(HU; ADAMS, 2009) presented a constant-density approach, which corrects the intermediate density errors by adjusting the half time step velocity with exact projection, allowing to simulate incompressible flows with high-density ratios by the projection SPH method. Recently, Yan et al. (YAN et al., 2016) extended the SPH method to cover solid phases, including deformable bodies and granular materials using the concept of volume fraction, establishing a new way of modeling fluid-solid interaction, as can be seen in Fig. 2.4.

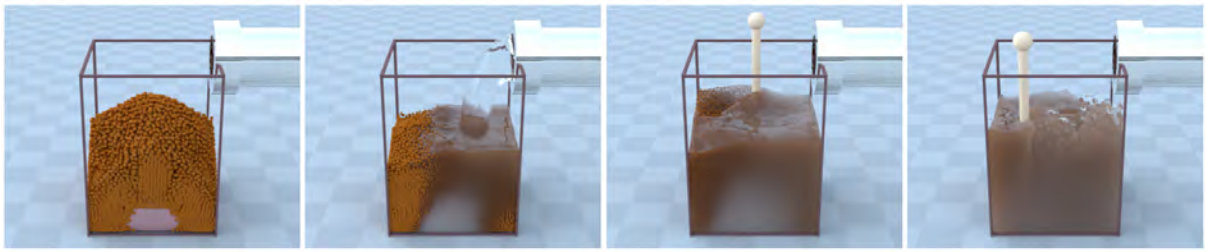

Figure 2.4: Results from the work of Yan et al. (YAN et al., 2016) simulating an instant coffee and a soft candy dissolving in water.

Many works benefited from the introduction of multiphase fluid flow in SPH. One example is the work of Tartakovsky and Meakin (TARTAKOVSKY; MEAKIN, 2005), which simulates miscible and immiscible fluid flows. 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 (TARTAKOVSKY; MEAKIN, 2006) as illustrated in Fig. 2.5 that shows visual results of multiphase flow with different capillary numbers that is larger as velocity increases.

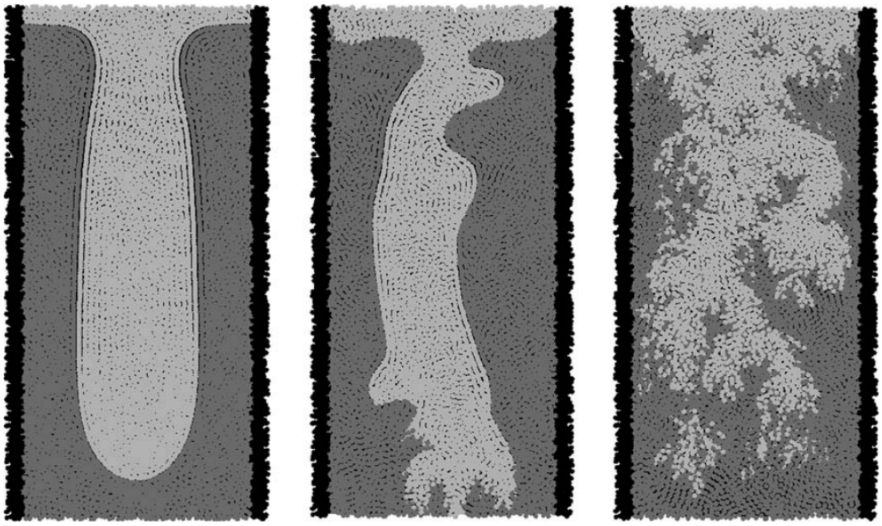

Figure 2.5: Results from the work of Tartakovsky and Meakin (TARTAKOVSKY; MEAKIN, 2006) for different values of capillary number.

Another example is the work of Hu and Adams (HU; ADAMS, 2006), where a multiphase SPH for the 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.

2.1.3 Fluid Rendering

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 (LORENSEN; CLINE, 1987) 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 (BLINN, 1982), but this method can create surface bumps, 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 (AKINCI et al., 2012) (AKINCI

et al., 2012) (ORTHMANN et al., 2013) or an anisotropic kernel (YU; TURK, 2013), which can generate a better visual result in sharp features and edges but it takes minutes to render. An example of 3D scalar field rendering can be seen in Figure 1.7.

A second approach to render the fluid surface is to use an explicit method, frequently used in an Eulerian context (ENRIGHT et al., 2002). In the SPH literature, a few works can be found using this approach, such as (PREMŽOE et al., 2003), which change the surface position using information of the particles simulation as can be seen in Fig. 2.6. Those methods have a high memory consumption and, to avoid this problem, methods such as the point splatting (MONAGHAN; RAFIEE, 2013) and ray-isosurface intersection with metaballs can be used (ZHANG; SOLENTHALER; PAJAROLA, 2008).

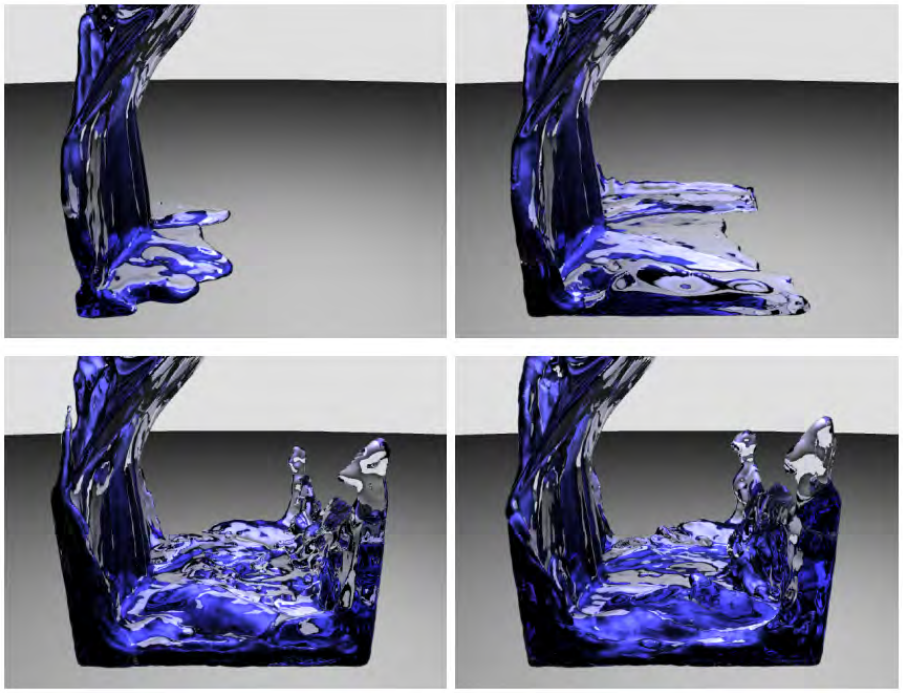

Figure 2.6: Results from the work of Premzoe et al. (PREMŽOE et al., 2003).

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 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 (MÜLLER; SCHIRM; DUTHALER, 2007), a Gaussian filter (LAAN; GREEN; SAINZ, 2009), a curvature flow (LAAN; GREEN; SAINZ, 2009) (AKINCI et al., 2013), and a post-smoothing filter (REICHL et al., 2014). The results from the work of van der Laan et al. (LAAN; GREEN; SAINZ, 2009) can be seen in Fig 2.7, which uses a curvature flow to smooth the fluid surface.

Recently, instead of computing light refraction on the fluid by blending fluid color with the background color, a ray tracing refraction is being calculated to provide a more realistic fluid rendering. Zirr and Dachsbacher (ZIRR; DACHSBACHER, 2015) proposed a view-adaptive

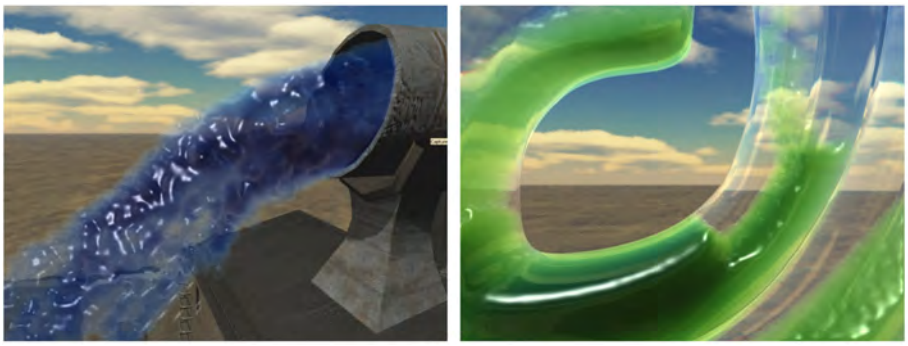

Figure 2.7: Results from the work of van der Laan et al. (LAAN; GREEN; SAINZ, 2009).

high-resolution voxelization of SPH particle data using ray casting refraction and reflection being able to provide a high-quality rendering solution at interactive rates.

Xiao et al. (XIAO; ZHANG; YANG, 2017) proposed an approach that combines particle splatting, ray-casting, and surface normal estimation. To reconstruct the fluid surface, an iso-surface is constructed based on ray-casting and a mass dependent function is used to calculate the scalar field, principal component analysis (PCA) is used to compute the surface normal and calculates the final fluid color using the method of van der Laan et al. (LAAN; GREEN; SAINZ, 2009) with raycast-based refraction as can be seen in Fig 2.8. The method can render 2 million fluid particles in full HD resolution (1920 x 1080) at 10 fps but, due to its mass dependence, the methods are not able to render massless methods like MPS.

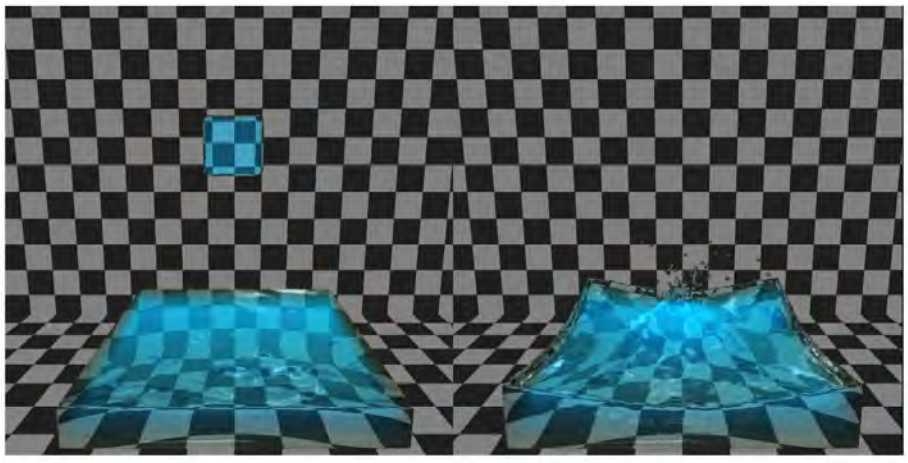

Figure 2.8: Results from the work of Xiao et al. (XIAO; ZHANG; YANG, 2017).

For multiphase fluids, as most of the works found in the literature are done using 2D simulation, there are not many works which render these fluids (CHEN et al., 2015; PENG et al., 2017). But, when it is necessary, most works use a Ray Tracing solution as in (SOLENTHALER; PAJAROLA, 2008; YAN et al., 2016). This approach results in a high-quality visual but, in both works, the rendering is done offline or with a performance of 1 fps.

3

PROPOSED METHODS

This chapter explains the SPH method developed by myself and colleagues from Voxar Labs (SILVA et al., 2015), and its modifications to support multiphase fluids (BRITO et al., 2017) and viscoelastic flows (BRITO et al., 2017), proposed in this work, are explained.

Three rendering solutions used to visualize the simulation results are explained: the screen space rendering solution based on the work of Laan et al. (LAAN; GREEN; SAINZ, 2009), the multiphase rendering based on the previous method (BRITO et al., 2017), and a Ray Tracing based solution to render single-phase fluids.

3.1 SPH Method

The SPH is a Lagrangian method that has been used mainly to simulate hydrodynamics problems solving the Navier-Stokes equation, defined by Eq. (3.1):

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

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 F_{ext} is the external forces function 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 (LIU; LIU, 2010):

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

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 distance, 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 distance of a particle and should satisfy some conditions (LIU; LIU, 2010):

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), that 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 SPH formulations 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, that may cause transverse mode instability (MORRIS; FOX; ZHU, 1997). Another possibility is to use a quintic kernel function as in Eq. (3.11). This kernel is more stable because it does not lead to a transverse mode instability (MORRIS; FOX; ZHU, 1997). There are some applications that do not need high accuracy, as 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) (MÜLLER et al., 2005).

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

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

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

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

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

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

The second part of the SPH method is to approximate the continuous hydrodynamics problem into a series of particles. A volume of fluid is described as a finite number of particles. Each particle in position x has velocity u, mass m, density ρ, viscosity μ, and influence radius h that describes the interaction of particle over its neighbors (MONAGHAN, 2005), as illustrated in Figure 3.1.

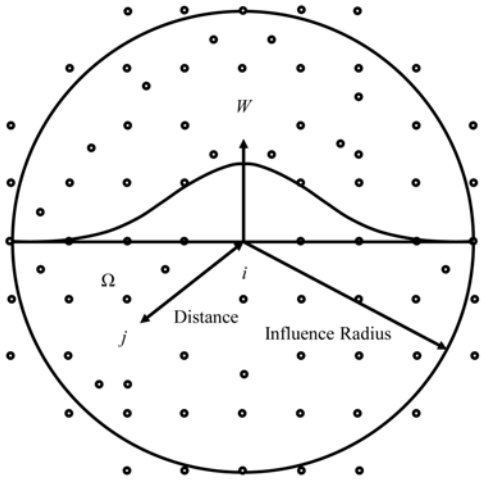

Figure 3.1: Particle approximation for a two-dimensional problem.

The influence radius of a particle defines a domain, being an area (in 2D) or a volume (in 3D) of influence (LIU; LIU, 2010). Given that two different particles are inside the domain of a particle a, the one closer to 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 (LIU; LIU, 2010).

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 smaller 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 distance between the particles, according to the work of (MONAGHAN; KAJTAR, 2009).

The neighborhood search is a potentially time-consuming step and is usually optimized with an accelerated spatial access structure like a uniform grid or an octree, instead of a naive brute-force search (IHMSEN et al., 2014).

In the discrete formulation, the interpolation in Eq. (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 large numerical error. To overcome those limitations, some algebraic operations are done and stable forms of the derivatives can be found, as in Eq. (3.17) (MONAGHAN, 1994):

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

The Navier-Stokes equation describes the fluid movement in three main components: pressure, viscosity, and external forces. 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 a little compressible, and because of that, the method simulates a quasi-incompressible equation to model the simulation (MONAGHAN, 2000).

In order to calculate those components, the first step is to calculate the particle densities, that can be calculated using the density summation equation as expressed in Eq. (3.18) (MONAGHAN, 2000):

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

This approach can make the simulation unstable for particles near the boundary or at the free surface, caused by an 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) (LIU; LIU, 2010):

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

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

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 Eq. (3.21) can be used; 2) in cases where the low-density variation must be enforced, the Tait equation Eq. (3.22) can be used (SILVA et al., 2015):

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

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

where k_p and B are pressure constants, ρ_0 is the rest density of the fluid and γ is a constant that usually has a value of 7 (SCHECHTER; BRIDSON, 2012).

The pressure force is commonly calculated using the derivative expression by Eq. (3.17), which results in Eq. (3.23). This approach ensures a modular equality between two particles and conserves linear and angular momentums, leading to a more stable simulation (MONAGHAN, 2005):

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

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 (MORRIS; FOX; ZHU, 1997):

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

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 large free surface, and simulate a shear and bulk viscosity (MORRIS; FOX; ZHU, 1997). Another option is to use the SPH formalism to calculate the viscosity force acting on a particle, which can be expressed in Eq. (3.31). This formulation gives a good visual result but numerical accuracy is not guaranteed (MÜLLER et al., 2005):

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

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 (MORRIS; FOX; ZHU, 1997):

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

A simple way to simulate 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 (SCHECHTER; BRIDSON, 2012). This method forces particles near each other to move with close velocity and conserves angular and linear momentums approximately by damping the particle velocity using Eq. (3.33) (SCHECHTER; BRIDSON, 2012):

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

where ε is the tunable parameter of the XSPH method.

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 (SCHECHTER; BRIDSON, 2012):

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

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

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

In this work, the WCSPH presented by Vieira-e-Silva et al. (SILVA et al., 2015) was used, and it calculates the pressure using the Tait equation Eq. (3.22). The pressure force is calculated by Eq. (3.23) and XSPH as viscosity factor and boundary condition.

The SPH method flow can be found in Fig. 3.2.

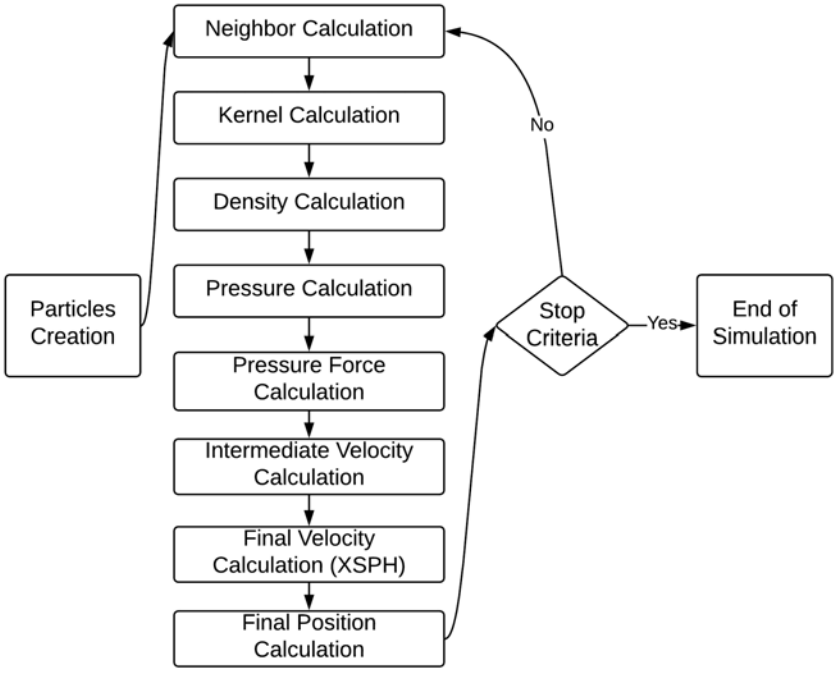

Figure 3.2: SPH simulation flow.

3.1.1 Handling Multiphase Simulation

To handle multiphase simulation, Solenthaler and Pajarola (SOLENTHALER; PAJAROLA, 2008) 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. (3.36).

$$\frac{d\rho_i}{dt} = \sum_j m_i(\mathbf{u}_i - \mathbf{u}_j)\nabla W_{ij} \tag{3.36}$$

In the method proposed by Vieira-e-Silva et al. (SILVA et al., 2015), the XSPH contribution is calculated for every neighbor of a fluid particle. But, to create a more realistic visual behavior, only the fluid particle contribution is used to handle multiphase flows. If the full neighborhood is considered, the fluid becomes too viscous.

3.1.2 Viscoelastic Scheme

To handle viscoelastic simulations, (TAKAHASHI et al., 2016) proposed a velocity correction $\triangle \mathbf{v}$ that is based on a set of pairwise connections that is created in the beginning of the simulation with distance r_{ij}, which is the initial particle distance. The velocity correction is calculated as Eq. (3.37):

$$\triangle \mathbf{v} = -\frac{1}{\triangle t}\sum_{j}^{C_f}\frac{c_i + c_j}{2}\frac{m_i}{m_i + m_j}D_{ij}\frac{\mathbf{x}_{ij}}{\left\|\mathbf{x}_{ij}\right\|} \tag{3.37}$$

where C_f is the number of connected fluid particles to particle i, c is the correction coefficient and D is a function defined as $D_{ij} = max(\left\|\mathbf{x}_{ij}\right\| - r_{ij}, 0)$.

Function D_{ij} expresses that velocity correction is only performed when the particle distance is larger than the initial particle distance r_{ij}, which means that the fluid is in expansion and the correction coefficient controls the stiffness of viscoelastic materials.

3.1.2.1 One Way Solid-Fluid Coupling

In order to compute a one way solid-fluid coupling and create a sticking behavior on the boundary, Eq. (3.37) must also be computed using the boundary particles as described in Eq. (3.38), which assumes $m_k = \infty$ (TAKAHASHI et al., 2016):

$$\triangle \mathbf{v} = -\frac{1}{\triangle t}\sum_{j}^{C_f}\frac{c_i + c_j}{2}\frac{m_i}{m_i + m_j}D_{ij}\frac{\mathbf{x}_{ij}}{\left\|\mathbf{x}_{ij}\right\|} - \frac{1}{\triangle t}\sum_{k}^{C_b}\frac{c_i + c_k}{2}D_{ik}\frac{\mathbf{x}_{ik}}{\left\|\mathbf{x}_{ik}\right\|} \tag{3.38}$$

where C_b is the number of connected boundary particles to particle i, and k is boundary particle.

Using this coupling approach, when the fluid hits a boundary, it should stick to the boundary instead of bouncing back.

The viscoelastic SPH method flow can be found in Fig. 3.3, where the red box is the modification done to simulate the viscoelastic flow.

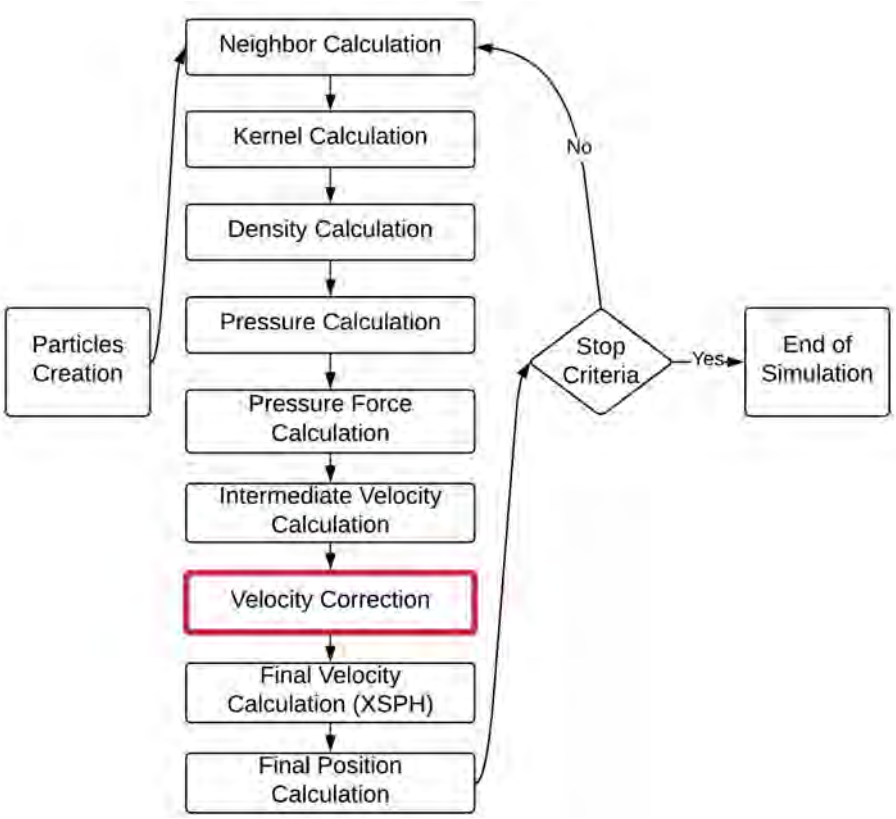

Figure 3.3: Viscoelastic SPH simulation flow.

3.2 Fluid Rendering

In this section, the screen-space rendering solution and the proposed extension to render multiphase flows is explained and also the proposed Ray Tracing based rendering solution is exposed.

3.2.1 Screen-Space Rendering Solution

To render a single phase fluid van der Laan et al. (LAAN; GREEN; SAINZ, 2009) proposed a screen-space solution, which uses the particle's position input to the rendering pipeline. The approach can be summarized into four steps: using the fluid particle's position, the surface depth (Section 3.2.1.1) and thickness (Section 3.2.1.2) are computed into different buffers. Then, the surface depth is smoothed (Section 3.2.1.3) using a Bilateral Filter and a final pass is done to combine depth, thickness and the scene behind the fluid into the final image (Section 3.2.1.4).

3.2.1.1 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 (INC, 2015). The point sprite size is calculated inversely proportional to the viewer's distance to the fluid surface. In other words, the sprite 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.

3.2.1.2 Surface Thickness

To give a more reliable impression on the fluid, Beer's law states that a fluid becomes less visible depending on the quantity of fluid in front of the observer, which will be referred to as thickness (LAAN; GREEN; SAINZ, 2009).

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.

3.2.1.3 Surface Smoothing Method

After rendering the particles as spheres, the surface will resemble a jam being able to visualize each particle individually. So, to avoid this behavior, the depth map is smoothed using a Bilateral Filter, which provides a good visual quality preserving silhouette edges and presenting a better performance if compared with other methods (LAAN; GREEN; SAINZ, 2009).

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 (depth map), which decreases the weight of pixels with large intensity differences.

3.2.1.4 Final Color

The fluid color can be calculated by Eq. (3.39), where F is the Fresnel function, a is the refracted fluid color, b is the reflected scene color, k_s and α are constants for the specular highlight, $\mathbf{n}$ is the surface normal, and $\mathbf{h}$ is the half-angle between camera and light, and $\mathbf{v}$ is the camera vector.

$$C_{out} = a(1 - F(\mathbf{n}\cdot\mathbf{v})) + bF(\mathbf{n}\cdot\mathbf{v}) + k_s(\mathbf{n}\cdot\mathbf{h})^{\alpha} \tag{3.39}$$

The view-space normal of a fluid point is determined by finite differences of the depth map. This approach may result in artifacts close to fluid silhouettes, so, in that case, the difference is calculated in the opposite direction which can be detected by the smallest finite difference (LAAN; GREEN; SAINZ, 2009).

In addition, the thickness $T(x,y)$ is used to attenuate the refracted color (LAAN; GREEN; SAINZ, 2009). So, the thicker the fluid, the less background should be visible as expressed in Eq (3.40). To create the illusion of a refraction, the thickness is also used to linearly perturb the background pixel color as seen in Eq. (3.41):

$$a = lerp(C_{fluid}, B(x + \beta \boldsymbol{n}_x + \beta \boldsymbol{n}_y), e^{-T(x,y)}) \tag{3.40}$$

$$\beta = T(x,y)\gamma \tag{3.41}$$

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.

3.2.2 Multiphase Rendering Solution

To render a multiphase fluid a shader solution based on the work of van der Laan et al. (LAAN; GREEN; SAINZ, 2009) is proposed, which is different from the original method in two aspects. The illustration of the rendering method can be found in Eq. (3.4).

The first difference is the number of buffers. Instead of using one buffer, for each fluid, the surface depth and thickness are calculated into different maps. 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 with Eq. (3.39).

Finally, the final color is the sum of both fluid colors and can be calculated as Eq. (3.42).

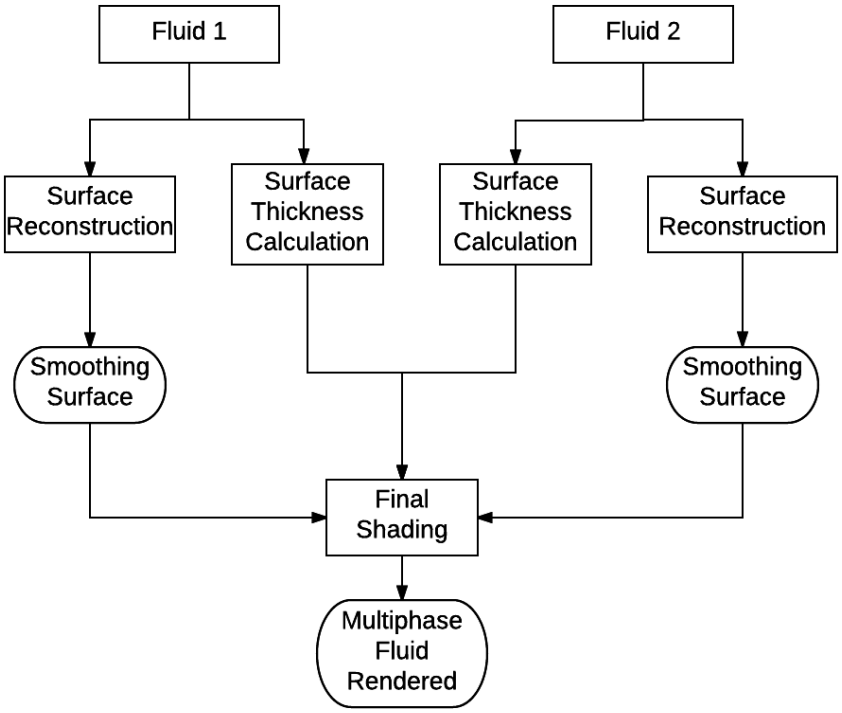

Figure 3.4: Render solution flow diagram.

$$C_{final} = k(C_{f_1} + C_{f_2}) \tag{3.42}$$

where k is a saturation coefficient, that avoids the image to become oversaturated, f_1 and f_2 are the first and second fluids, respectively.

3.2.3 Ray Tracing Rendering Solution

To obtain a more realistic fluid rendering, a Ray Tracing based rendering solution was proposed, which can be summarized into four steps: first, each particle of the scene is rendered as a sphere, and the depth map is stored into a buffer. Then, the depth map is smoothed using the method proposed by van der Laan et al. (LAAN; GREEN; SAINZ, 2009). The normal of each point of the surface is calculated using the depth difference. Finally, the final color is calculated using the fluid color and the result from the ray-cast based reflection and refraction are computed. Each step of the rendering solution will be explained in the following sections.

3.2.3.1 Ray Tracing

A Ray Tracing based algorithm is used to reconstruct the fluid surface. The ray tracer is a technique that 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 Appel (APPEL, 1968) to solve the visibility problem.

The recursive Ray Tracing was introduced by Whitted in 1980 (WHITTED, 2005); 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 the scene from the camera, being possible to render images with more details and more realism.

Unlike the raster algorithm, Ray Tracing can render objects without the need of transforming each of them into a polygonal mesh, giving more precision to the method. For instance, a sphere can be represented as an algebraic entity and the spatial equation can get all the information to a proper rendering.

In the beginning, the Ray Tracing algorithm was used for off line rendering purposes due to high computational demand. But with the increase of computational performance of the CPU and GPU, many works described real-time approaches for the algorithm (NAH et al., 2015) (SINGH; NARAYANAN, 2010) (LEE et al., 2013) (LIU et al., 2015).

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 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. (3.43):

$$\mathbf{R}(t) = \mathbf{E} + t\mathbf{D} \tag{3.43}$$

where $\mathbf{R}(t)$ 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 ray, and this ray is called a secondary ray. 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. (3.44):

$$\mathbf{R}_{out} = 2\mathbf{N}(\mathbf{N} \cdot \mathbf{R}_{in}) - \mathbf{R}_{in} \tag{3.44}$$

where $\mathbf{R}_{in}$ is the incident ray, $\mathbf{R}_{in}$ is the reflected ray and $\mathbf{N}$ is the surface normal.

A refracted ray is a secondary ray 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. (3.45):

$$n_1 \cdot sin(\theta_i) = n_2 \cdot sin(\theta_r) \tag{3.45}$$

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

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. (3.46), and any point of the spherical surface must satisfy this equation.

$$(P_x - C_x)^2 + (P_y - C_y)^2 + (P_z - C_z)^2 = R^2 \tag{3.46}$$

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

To reconstruct the surface of the fluid, 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. This way, after substituting Eq. (3.43) by Eq. (3.46), the intersection between a ray and a sphere is expressed by Eq. (3.47):

$$((E_x + tD_x) - C_x)^2 + ((E_y + tD_y) - C_y)^2 + ((E_z + tD_z) - C_z)^2 = R^2 \tag{3.47}$$

After expanding this equation, the same can be expressed by a quadratic equation (Eq. 3.48), which can be solved by the Bhaskara equation.

$$at^2 + bt + c \tag{3.48}$$

where

$$a = D_x^2 + D_y^2 + D_z^2 \tag{3.49}$$

$$b = 2(D_x(E_x - C_x)) + D_y(E_y - C_y) + D_z(E_z - C_z) \tag{3.50}$$

$$c = (E_x - C_x)^2 + (E_y - C_y)^2 + (E_z - C_z)^2 - R^2 \tag{3.51}$$

The unit normal of the sphere can be calculated using Eq. 3.52:

$$\widehat{\mathbf{n}} = \frac{\mathbf{P} - \mathbf{C}}{|\mathbf{P} - \mathbf{C}|} \tag{3.52}$$

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

3.2.3.2 Surface Reconstruction

After rendering the spheres using the ray tracer, the result is a surface with a "blobby" or jelly-like look. To create a more realistic surface without the need of creating any other structure besides 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 (LAAN; GREEN; SAINZ, 2009). The first step is to create the depth map of the scene; this depth map is created using the ray casting 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

rendering process can continue.

3.2.3.3 Screen Space Curvature Flow

To achieve a better-looking result, van der Laan et al. (LAAN; GREEN; SAINZ, 2009) proposed a curvature flow approach, based on (MALLADI; SETHIAN, 1997), 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 Eq. (3.53):

$$\frac{\partial z}{\partial t} = H \tag{3.53}$$

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 $\mathbf{n}$, as in Eq. (3.54):

$$2H = \nabla \cdot \widehat{\mathbf{n}} \tag{3.54}$$

A point $\mathbf{P}$ 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 Eq. (3.55):

$$\mathbf{P}(x,y) = \begin{pmatrix} (\frac{2x}{V_x} - 1)/F_x \\ (\frac{2y}{V_y} - 1)/F_y \\ 1 \end{pmatrix} z(x,y) = \begin{pmatrix} W_x \\ W_y \\ 1 \end{pmatrix} z(x,y) \tag{3.55}$$

where V_x and V_y are the dimensions of the viewport, and F_x and F_y are the focal length in the x and y directions.

The normal of a point on the surface is calculated by the cross product between the derivatives of $\mathbf{P}$ in the x and y directions, as expressed by Eq. (3.56):

$$\mathbf{n}(x,y) = \frac{\partial \mathbf{P}}{\partial x} \times \frac{\partial \mathbf{P}}{\partial y} = \begin{pmatrix} C_x z + W_x \frac{\partial z}{\partial x} \\ W_y \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial x} \end{pmatrix} \times \begin{pmatrix} W_y \frac{\partial z}{\partial y} \\ C_y z + W_y \frac{\partial z}{\partial y} \\ \frac{\partial z}{\partial y} \end{pmatrix} \approx \begin{pmatrix} -C_y \frac{\partial z}{\partial x} \\ -C_x \frac{\partial z}{\partial y} \\ C_x C_y z \end{pmatrix} z \tag{3.56}$$

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 computations as they have a small contribution to the calculation.

The unit normal is calculated by the expression of Eq. 3.57:

$$\widehat{\mathbf{n}} = \frac{\mathbf{n}(x,y)}{|\mathbf{n}(x,y)|} = \frac{(-C_y \frac{\partial z}{\partial x}, -C_x \frac{\partial z}{\partial x}, C_x C_y z)^T}{\sqrt{D}} \tag{3.57}$$

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 Eq. (3.53) to express H in a way

that can be derived.

The z component of the divergence is always zero, because z is a function of x and y, being kept constant when x and y are also maintained constant. So

$$2H = \frac{\partial \widehat{n}_x}{\partial x} + \frac{\partial \widehat{n}_y}{\partial y} = \frac{C_y E_x + C_x E_y}{D^{3/2}} \tag{3.58}$$

in which

$$E_x = \frac{1}{2}\frac{\partial z}{\partial x}\frac{\partial D}{\partial x} - \frac{\partial^2 z}{\partial x^2} D \tag{3.59}$$

$$E_y = \frac{1}{2}\frac{\partial z}{\partial y}\frac{\partial D}{\partial y} - \frac{\partial^2 z}{\partial y^2} D \tag{3.60}$$

A simple Euler integration of Eq. (3.53) 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 higher computation time.

3.2.3.4 Rendering

After the reconstruction of the fluid surface, the fluid is rendered as a transparent surface, and the ray refraction and reflection are calculated by a Schlick approximation as express by Eq. (3.61) and Eq. (3.62). To achieve higher performance, a single layer refraction is calculated, which is not a physically accurate model but can provide a high-quality visualization (XIAO; ZHANG; YANG, 2017).

$$\mathbf{r} = 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n} - \mathbf{l} \tag{3.61}$$

$$R(\theta) = R_0 + (1 - R_0)(1 - cos\theta)^5 \tag{3.62}$$

where $R_0 = (\frac{\mathbf{n}-1}{\mathbf{n}+1})^2$.

4

IMPLEMENTATION

To achieve high performance, the SPH tool was implemented on top of the open source DualSPHysics code (CRESPO et al., 2015), and several modifications were developed to simulate WCSPH with support to multiphase flow and viscoelastic fluids.

To implement the Ray Tracing based fluid rendering, the NVIDIA OptiX ray tracing engine (PARKER et al., 2010) was used, which it does the Ray Tracing process entirely on GPU using the CUDA programming model (NVIDIA, 2018). OptiX gives support to primary and secondary rays as explained in Whitted's simple model (WHITTED, 2005) and also supports reflections, refractions, and shadows.

4.1 DualSPHysics

DualSPHysics is an open source project created with the purpose of encouraging other researchers to study SPH, and it has GNU General Public License as published by the Free Software Foundation (CRESPO et al., 2015). The code is available for CPU and GPU based SPH simulation, being able to compute the fluid behavior with numerical stability and accuracy. It has been used for many applications (MOKOS; ROGERS; STANSBY, 2017) (VACONDIO et al., 2016) (MOKOS et al., 2015). The code is written in C++ using OpenMP (OPENMP ARCHITECTURE REVIEW BOARD, 2017) for the parallel CPU implementation and CUDA (NVIDIA, 2018) for the GPU implementation.

The SPH code implemented on the DualSPHysics can be divided into three main phases (CRESPO et al., 2015): (1) neighborhood organization, (2) particle interaction, and (3) time integration.

In phase one, the particles neighborhood is computed using a cell-linked list (CLL) approach (DOMÍNGUEZ et al., 2011). In the CLL, the domain is divided into square cells (2D) or cube cells (3D) with twice the influence radius (h) and the particles are stored into a list depending on the cell they belong to.

To create the neighborhood of a particle, only particles of adjacent cells are considered potential neighbors. It is worth noticing that a neighbor list is not created, but a list of particles is reordered depending on the cell they belong to. This approach is faster and consumes less

memory than creating a real list of neighbor particles (DOMÍNGUEZ et al., 2011). To conclude the first step, every array is reordered using the list of particles.

The second phase calculates the interaction between neighbor particles by solving the momentum and continuity equations. The interaction between two particles occurs if the distance is less than $2h$.

Using the results from the second phase, the time integration is calculated. In this phase, the new particles density, velocity, and position are calculated, a new time step can be computed, particle information is stored in the memory, and the arrays are ordered so that particles inside the same cell can be close to each other.

The input data are read by a tool named GenCase, which reads a .xml file that contains the geometry of the simulation (fluid and boundary) and several constants used for the simulation, such as the reference density of the fluid, the gamma value used by the Tait equation, and the gravity value. Several geometries can be created using the GenCase, such as spheres, pyramids, and boxes, as can be seen in Codes 4.1, 4.2, and 4.3, that show the XML code to create a sphere, a box, and a pyramid, respectively. The initial particle spacing is also defined in the input .xml file.

Code 4.1: Sphere defined in the input file by its radius and center point.

```
<drawsphere radius="0.2">

<point x="0.2" y="0.5" z="0.6" />

</drawsphere>
```

Code 4.2: Pyramid defined in the input file by its four vertices.

```
<drawpyramid>

   <point x="2" y="2" z="4" />
   <point x="1" y="1" z="2" />
   <point x="4" y="1" z="2" />
   <point x="1" y="5" z="2" />

</drawpyramid>
```

Code 4.3: Box defined in the input file by a point and the size of the box in the x, y and z directions.

```
<drawbox>

<point x="0.2" y="0.5" z="0.2" />
<size x="0.3" y="0.3" z="0.3" />

</drawbox>
```

After each time step, the particle positions are stored into a .xyz file that contains the list of particle positions or a .vtk file that also stores the density, velocity, and pressure of each particle.

4.1.1 DualSPHysics Modifications

In this section, the modifications done to the DualSPHysics code will be explained to implement the WCSPH proposed by Vieira-e-Silva et al. (SILVA et al., 2015), the multiphase flow and, the viscoelastic behavior.

4.1.1.1 WCSPH

The WCSPH method was implemented in the open source code of DualSPHysics v4.0 (CRESPO et al., 2015), and uses a grid to calculate the neighborhood of a particle and CUDA to accelerate the calculations being able to simulate up to millions of particles.

Two parts of the DualSPHysics code were modified to implement WCSPH: (1) Euler integration, and (2) XSPH calculation.

4.1.1.1.1 Euler Integration The Euler integration is used to calculate the new position of a fluid particle as shown in Eq. (3.35). The particles position and velocity are stored in the `Posxyg`, `Poszg` and `Velrhopg` arrays for the GPU solution, which are already used by the original DualSPHysics implementation. After each time step, the particle grid position `Dcellg` is updated as can be seen in Code 4.4.

Code 4.4: Euler integration GPU code.

```
template<bool floating, bool shift> __global__ void
   KerComputeStepPosition
(unsigned n, unsigned npb, const float4 *velrhop1, double dt,
   double2 *posxy, double *posz, unsigned *dcell, word *code)
{
   unsigned p = blockIdx.y*gridDim.x*blockDim.x +
      blockIdx.x*blockDim.x + threadIdx.x;
   if (p<n){
      if (p<npb){// -Boundary particle do not move

      }
      else{ // -Fluid particle

         float3 rpos = make_float3(posxy[p].x, posxy[p].y, posz[p]);

         rpos.x += velrhop1[p].x*dt;
         rpos.y += velrhop1[p].y*dt;
         rpos.z += velrhop1[p].z*dt;

         double dx = rpos.x - CTE.maprealposminx;
         double dy = rpos.y - CTE.maprealposminy;
         double dz = rpos.z - CTE.maprealposminz;

         posxy[p] = make_double2(rpos.x, rpos.y);
         posz[p] = rpos.z;
         //-Guarda celda y check.
         //-Stores cell and checks.

         unsigned cx = unsigned(dx / CTE.scell), cy = unsigned(dy /
            CTE.scell), cz = unsigned(dz / CTE.scell);
         dcell[p] = PC__Cell(CTE.cellcode, cx, cy, cz);


      }
   }
}
```

4.1.1.1.2 XSPH Calculation To calculate XSPH, the summation from Eq. (3.33) is computed using a CUDA kernel which interacts through the particles neighborhood and stores the

result into a `float3`. In sequence, another CUDA kernel computes the final velocity as Eq. (3.34) and stores the result into the velocity array as shown in Code 4.5. Line 4 calculates the particle index using the CUDA kernel information and, if it is a fluid particle, the velocity `velrhopnew` is updated using the XSPH result.

Code 4.5: XSPH GPU code.

```
template<bool floating, bool shift> __global__ void
   KerComputeStepVelocity
(unsigned n, unsigned npb, const float4 *velrhop1, const float3
   *xsph, double dt, word *code, float4 *velrhopnew, double eps)
{
unsigned p = blockIdx.y*gridDim.x*blockDim.x + blockIdx.x*blockDim.x
   + threadIdx.x;
 if (p<n){
   if (p<npb){// -Boundary particle
      velrhopnew[p] = make_float4(0, 0, 0, velrhop1[p].w);
   }
   else{ // -Fluid particle

      float4 rvelrhop = velrhop1[p];
      float3 xsphp = xsph[p];

      //velocity update using the xsph
      rvelrhop.x = float(double(rvelrhop.x) - double(xsphp.x)*eps);
      rvelrhop.y = float(double(rvelrhop.y) - double(xsphp.y)*eps);
      rvelrhop.z = float(double(rvelrhop.z) - double(xsphp.z)*eps);
      velrhopnew[p] = rvelrhop;

   }
 }
}
```

4.1.1.2 Velocity Correction for Viscoelastic Behavior

To simulate the fluid viscoelastic behavior, first it is necessary to create the connections from the fluid particles, so the initial fluid position is kept into a `float4` array that is used for creating the connections using a CLL. To calculate the one-way solid-fluid coupling, the connections are updated to the point where the fluid has the highest number of boundary neighbor particles. In sequence, two CUDA kernels are created. The first one interacts through the particles connections and calculates the velocity correction as Eq. (3.37). Then, the second kernel uses the velocity correction and updates the fluid velocity as shown in Code 4.6, which updates the

fluid velocity `velrhopnew` using the velocity correction result.

Code 4.6: Velocity correction GPU code.

```
template<bool floating, bool shift> __global__ void
   KerComputeStepVelocityCorr
(unsigned n, unsigned npb, const float4 *velrhop1, const float3
   *velcorr, double dt, word *code, float4 *velrhopnew)
{
   unsigned p = blockIdx.y*gridDim.x*blockDim.x +
      blockIdx.x*blockDim.x + threadIdx.x;
  if (p<n){
   if (p<npb){ //-Boundary particle
      velrhopnew[p] = make_float4(0, 0, 0, velrhop1[p].w);
   }
   else{ // -Fluid particle

      float4 rvelrhop = velrhop1[p];
      float3 velcorrp = velcorr[p];
         /* Velocity update using the
               viscoelastic correction */
      rvelrhop.x = float(double(rvelrhop.x) - double(velcorrp.x)*(1
         / dt));
      rvelrhop.y = float(double(rvelrhop.y) - double(velcorrp.y)*(1
         / dt));
      rvelrhop.z = float(double(rvelrhop.z) - double(velcorrp.z)*(1
         / dt));
            velrhopnew[p] = rvelrhop;

   }
  }
}
```

4.1.1.3 Multiphase Flow

Three modifications were made to the already implemented WCSPH method.

First, the reference density of each particle is kept in a float array named `Rho0`. This density will be used in Eq. 3.22 for the pressure calculation and also to compute the mass of the particles. The particle is initialized with a mass value corresponding to the water reference density ($1000 kg/m^3$), so, to calculate the correct mass of each particle, its mass is multiplied by the ratio between the reference density and the water reference density.

Two methods were also modified to solve the multiphase fluid equation: the density

calculation and the XSPH method, as explained in Sec. 3.1.1.

4.2 OptiX

OptiX (PARKER et al., 2010) is a general purpose ray tracing engine developed by NVIDIA, which provides a programmable Ray Tracing pipeline with a lightweight scene representation, and can be used in areas like rendering, artificial intelligence, animation, and scientific visualization.

The OptiX main characteristics are: (a) it is focused exclusively on Ray Tracing algorithm, and it is not embedded in some specific pipeline, (b) it can be used for rendering and non-rendering based applications, (c) the engine abstracts batching, reordering of rays and the creation of acceleration structures, (d) the engine tunes the execution of the used hardware and, (e) it provides a flexible node graph that allows the scene to be organized to reach a higher performance.

An application using OptiX is organized into a series of programs that can be of seven types: ray generation, intersection, bounding box, closest hit, any hit, miss, exception, selector visit. The ray generation programs are the entry of the Ray Tracing pipeline, that performs a pixel operation that can be tracing a pixel from the camera, computing baked lighting, or even performing a pixel operation on a buffer. The intersection programs are created to perform the ray-geometry intersection tests and can also compute any operation based on the hit position and, to improve the performance of the intersection test, the test is done using the bounding box of the geometry which is calculated in the bounding box program.

The closest hit programs perform operations when the closest intersection is calculated. Those operations usually are shading and storing the result on a buffer or casting new rays into the scene. If it is necessary to do some calculation for each ray-object intersection, the any hit programs are used, and for cases where there is no intersection, the miss programs are invoked. Finally, the exception programs are invoked when the system finds an exception condition such as no memory available or an index out of range, and the selector visit programs perform operations on the node graph.

A scene is represented as a lightweight graph that controls efficiently the traversal of rays through the scene. The scene graph is composed of four main nodes: the group nodes cluster the other nodes and have an acceleration structure associated with it, the geometry group nodes contain the geometry and the material of an object, the transform nodes have a 4x3 matrix which performs an affine transformation on the geometry, and the selector nodes are used to perform any operation on their children.

An example of scene graph can be seen in Fig. 4.1.

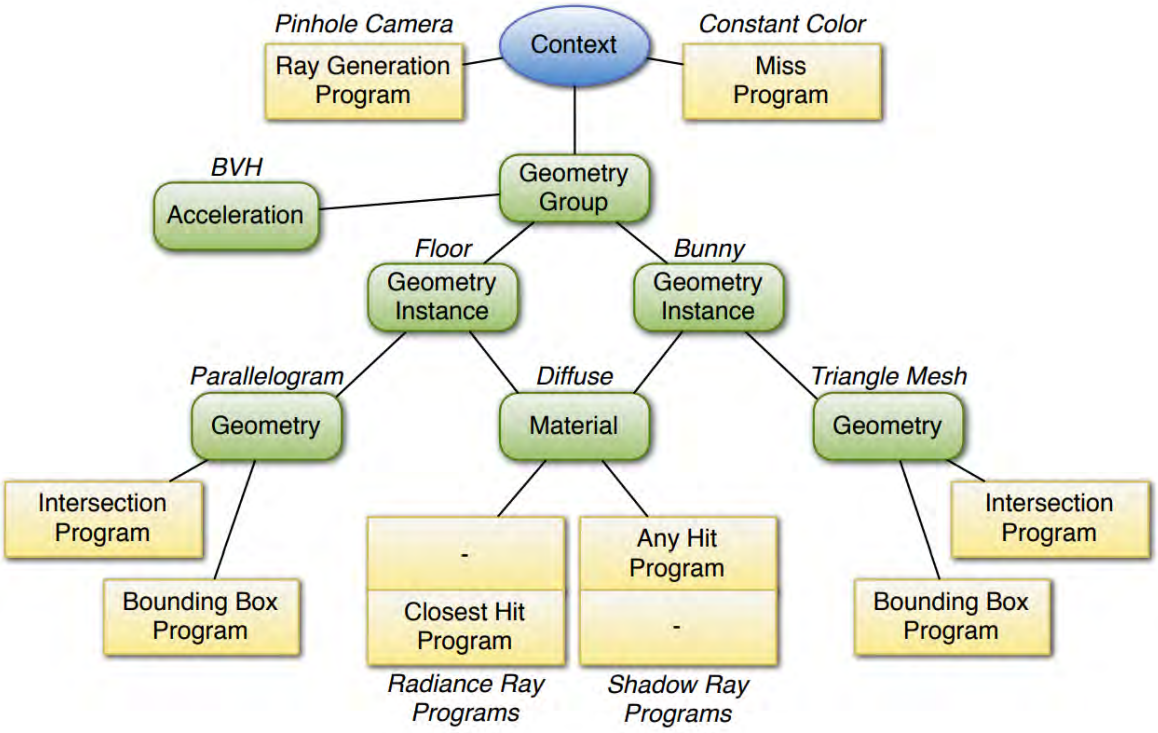

Figure 4.1: OptiX scene graph example.

4.2.1 Rendering Implementation

The ray traced based rendering implementation is composed of three main ray generation programs. The first one is the depth map creation, which is done by tracing rays from a pinhole camera into the scene that contains a series of spheres that are the fluid particles. The value of the closest hit of each ray is stored into a `rtBuffer`.

To create a more realistic fluid surface, the depth map is smoothed using the technique explained in Sec. 3.2.3.3 and, to achieve a high-quality result, the smoothing is performed with 50 iterations. Despite the use of a ray generation program, that program does not cast any ray in the scene, but it does a processing on each pixel.

The final ray generation program uses the smoothed depth map to calculate the normal of those points, calculate the final color of the fluid and it also traces secondary rays to calculate the refraction, and reflection colors. To achieve high performance and more realistic visual quality, the secondary rays do not hit the fluid, so the particles within the fluid can not be visualized.

Three geometry groups are used in the code, one with all the elements of the scene (background + fluid), the second one with only the spheres which are the fluid particles and the last one with background elements. The first group is used to shade the fluid and the background, the fluid group is used for the depth map creation, and the background group is used to trace the secondary rays of the scene after a ray hits a fluid. This division is necessary to render the fluid correctly, but it overloads the memory that is only possible because the node representation of the OptiX is lightweight.

The Linear Bounding Volume Hierarchy (LBVH) (LAUTERBACH et al., 2009) was used as acceleration structure which focuses on the construction speed and, since the fluid particles are always moving, this structure is the best option available.

5

TEST CASES

In this chapter the test cases used to validate the SPH method for multiphase flow, viscoelastic behavior, and its rendering will be explained.

5.1 Multiphase Flow

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

5.1.1 Density Equilibrium

In the first test, two fluids with different densities, $1000kg/m^3$ and $3000kg/m^3$, are inside a container. This test validates the behavior of two fluids with different densities that after some time behave in a way that the heavier fluid must be under the lighter 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 Figure 5.1.

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

5.1.2 3D Double Dam Break

The second test case is a 3D double dam break simulation, where the fluid on the left has a density of $1000kg/m^3$, and the other fluid has a density of $3000kg/m^3$. This scenario is useful for solving flows that have a great 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 L_b size of $12m$, as can be seen in Fig. 5.2.

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

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

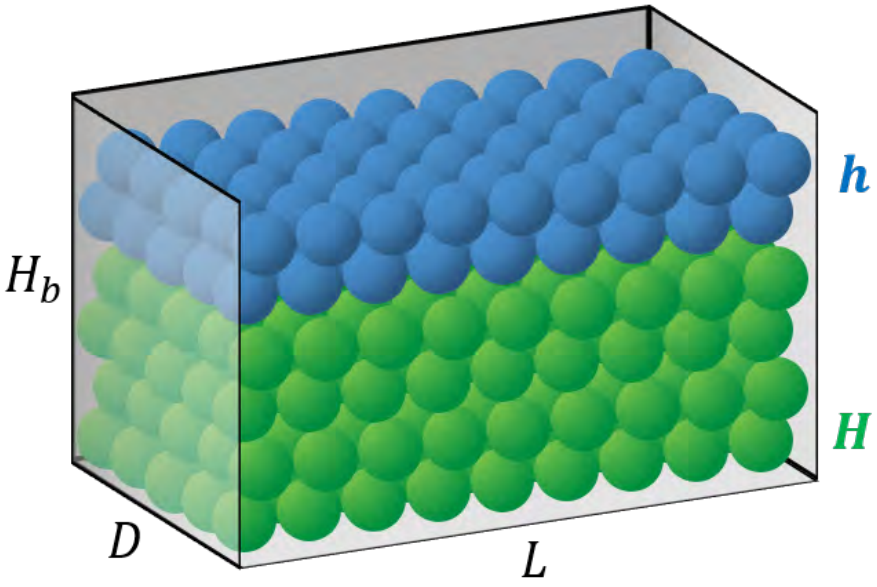

Figure 5.1: Density equilibrium test case. The denser fluid is illustrated by blue particles and the lighter one is represented by green particles.

surface reconstruction result. And finally, to understand how the particles number influences performance, three scenarios were used: $100k$, $500k$, and $1M$ fluid particles.

5.2 Viscoelastic Fluids

To validate the visual results from the SPH method, three different objects were released towards the ground with no initial velocity: a sphere, a cube, and a triangular base pyramid.

The sphere is centered in the point $(0.3, 0.3, 0.3)$ with a $0.2m$ radius. The cube is centered in the point $(0.45, 0.45, 0.45)$ with lateral size equal to $0.3m$. Both sphere and cube have initial particle spacing of $0.01m$, and are composed by 40k particles. The pyramid is composed by the points $(1,1,2)$, $(4,1,2)$, $(1,5,2)$ and, $(2,2,4)$ and has initial particle spacing of 0.05, resulting in 60k particles.

The sphere test case is also repeated using three correction coefficients: 0.01, 0.001, and 0.0005 to understand the influence of the parameters on the fluid behavior. This case is also used to compare the performance of the GPU solution with the OpenMP implementation, by using $100k$, $300k$, $500k$, $750k$ and $1M$ fluid particles with 0.01 correction coefficient value.

To validate the one-way solid-fluid coupling, the sphere test and the pyramid test are used with the same aforementioned parameters but the pyramid has an initial velocity $v_0 = (10, 0, 0)$, that makes the object to be thrown against the wall on $x = 5$.

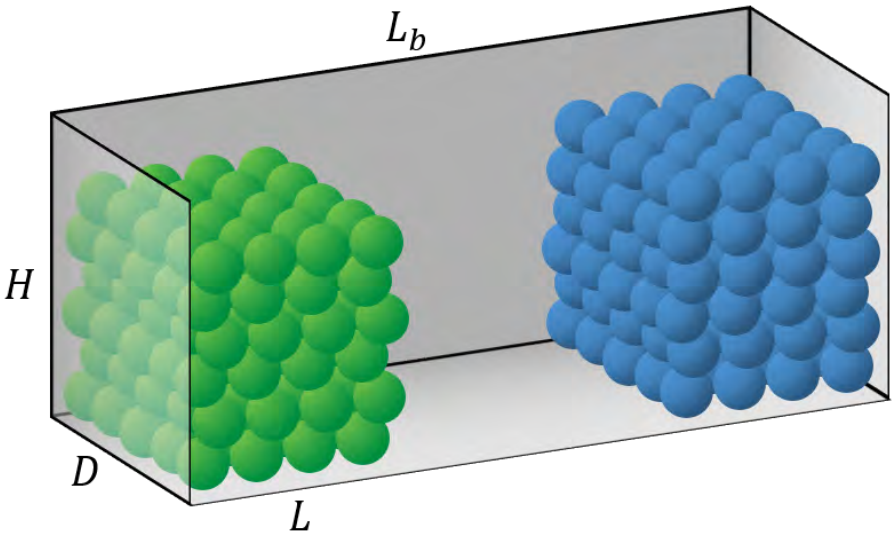

Figure 5.2: 3D double dam break test case. The denser fluid is illustrated by the blue particles while the lighter is represented by the green particles.

5.3 Ray Tracing Based Rendering Solution

To test rendering performance and visual quality, two test cases were performed. First, a 3D dam break was constructed. The dam was built as a 3D rectangular domain that bursts right 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 L_b of $1.2m$. The initial configuration of this test case can be visualized in Figure 5.3.

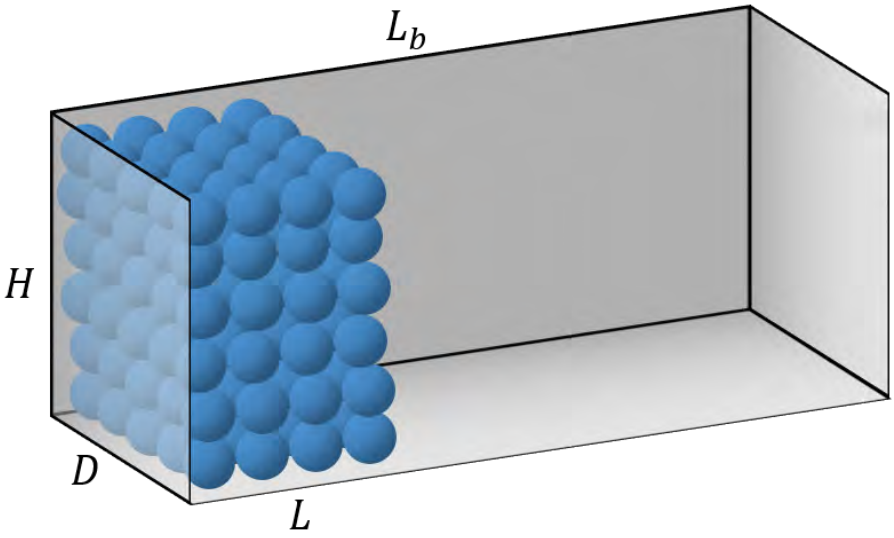

Figure 5.3: Initial configuration of the 3D dam break.

A water drop scene was constructed in which a 3D box configuration of fluid with initial height d of 0.2m, an initial width l of $0.2m$, and an initial depth d of $0.2m$ falls from a height H_b of 1m and zero initial velocity into a confined 3D rectangular configuration of fluid with H of

0.5m, L of 0.5m, and D of 0.5m. The initial configuration of this test case can be visualized in Fig. 5.4.

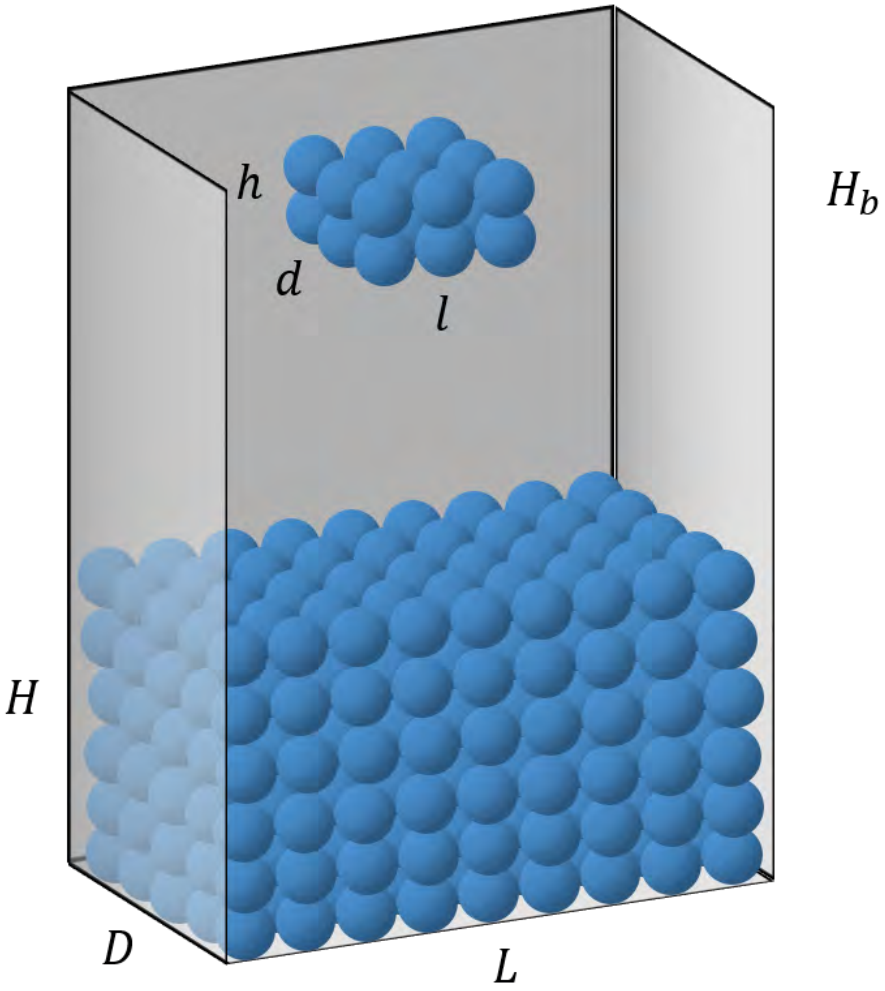

Figure 5.4: Initial configuration of the 3D drop case.

The 3D dam break is composed by 1M fluid particles and the water drop scene contains 500k fluid particles.

Similar to the work of Xiao et al. (XIAO; ZHANG; YANG, 2017), in order to analyze the performance of the algorithm, several tests were done changing the screen resolution values (1024 x 1024, 1440 x 1440 and 1920 x 1920), the number of particles (750k, 1M, and 2M) and also static and dynamic scenes were tested. In the dynamic scenes, the fluid particles change their positions and, for each frame, the acceleration structure must be reconstructed.

6

RESULTS

In this chapter, the results are presented regarding the simulation of multiphase flow and viscoelastic simulation using the SPH method. The results from the multiphase rendering solution and the Ray Tracing based rendering solution are presented. As this work tackles large scale simulations, the results will be evaluated with up to $1M$ particles.

6.1 Hardware and Software Infrastructure

The computer used to run the test cases had 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 GTX 960 with 4 GB of RAM and CUDA 7.5.

6.2 Multiphase Flow

This section will present the results from the multiphase flow simulation for the two test cases explained in Sec. 5.1, and will analyze the rendering results regarding visual quality and performance (fps).

The rendering was done at a 1024 x 768 resolution, which is the resolution used in (LAAN; GREEN; SAINZ, 2009), and the reference paper was able to achieve a performance between 44 and 55 fps for $64k$ particles. Although this article results cannot be directly compared with ours, it is important to notice that this technique reaches a real-time performance.

6.2.1 Density Equilibrium

The simulation proved to have a realistic behavior as the denser liquid goes to the bottom of the container over time and XSPH was able to create realistic boundary conditions, as can be seen in the right part of Figure 6.1. However, the simulation took a lot of time to converge, because the XSPH method has a big influence on particles with a few neighbors of the same fluid.

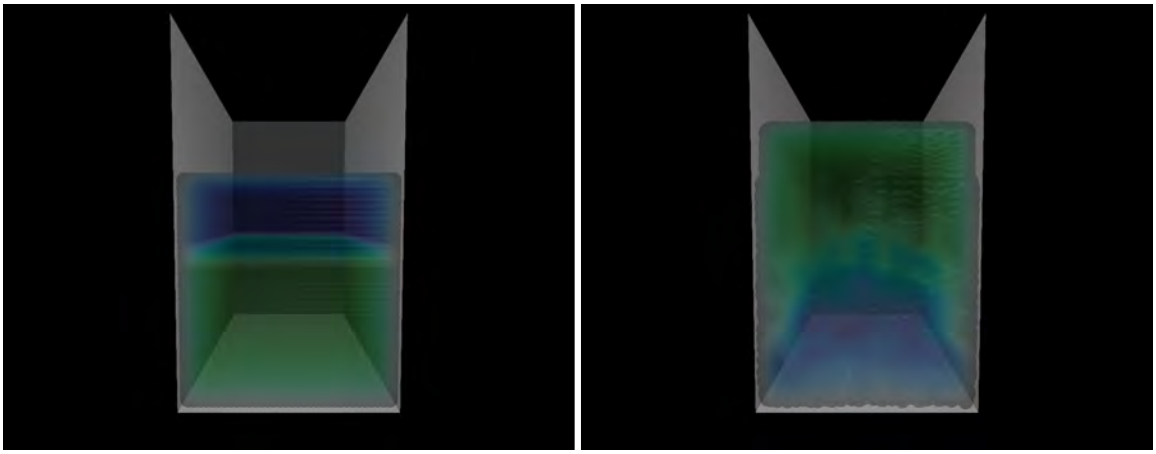

Figure 6.1: Density equilibrium results. Left: initial state (t = 0s); Right: t = 10s.

6.2.2 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 Figure 6.2. It is also possible to notice the XSPH contribution as the fluid sticks to the wall due to its viscosity and also the boundary condition is well established, as visualized in the right part of Figure 6.2.

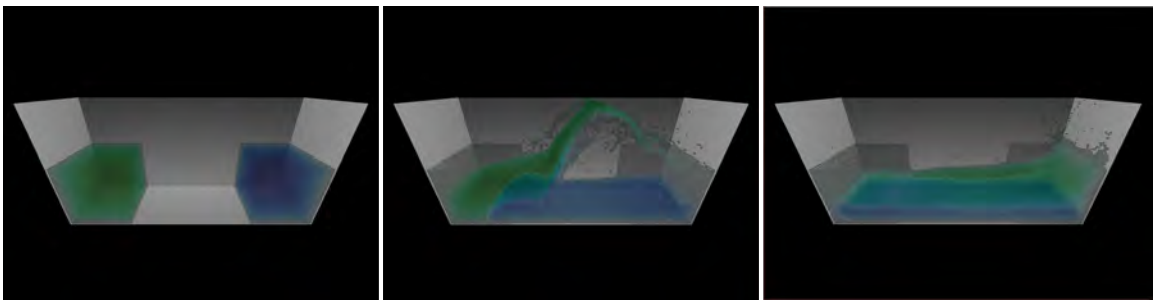

Figure 6.2: 3D double dam break results. Left: initial state (t = 0s); Middle: t = 2s; Right: t = 5s.

6.2.3 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 the left part of Figure 6.1. Also, as shown in the work of van der Laan et al. (LAAN; GREEN; SAINZ, 2009), 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 6.1, which compared the performance (fps) according to the percentage of screen filled. The performance drop happens because, if the camera is too close to the fluid, more pixels will be smoothed and more processing will happen.

Table 6.1: Comparison of performance (fps) varying the percentage of screen filled.

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

Table 6.2: Comparison of performance (fps) varying the percentage of screen filled and the number of particles.

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

The four scenarios used in the comparison aforementioned were tested with three different numbers 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% of the screen, but, as the number of particles increases, performance drops in the other three scenarios, as can be seen in Table 6.2.

This 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 and theartifacts of smoothing disappears. These results can be seen in Fig. 6.3, which compares the smoothing results for $100k$, $500k$, and $1M$ fluid particles.

Figure 6.3: Smoothing results for $100k$ (left), $500k$ (middle) and $1M$ (right) fluid particles.

As the number of particles increases, the smoothing results improve, which can be observed by comparing the left and middle results of Fig. 6.3. But, as the number keeps increasing, the improvement on the surface smoothing is almost imperceptible, which can be seen by comparing the middle and right results of Fig. 6.3.

The full visualization of the multiphase simulation results can be seen at `https://www.youtube.com/watch?v=XbTWpGaYfis`.

6.3 Viscoelastic Simulation

Firstly, a visual analysis of the methods will be made using different shapes and correction coefficient values. Then, performance analysis will be done comparing computational times of the GPU and CPU implementations.

6.3.1 Visual Analysis

As previously described, different shapes were released towards the ground: a sphere, a cube, and a pyramid. By the fact that viscoelastic fluids are being simulated, it is expected the fluid will tend to return to its initial shape, and the fluid should not spread on the floor like a regular fluid, but instead, it should show an elastic deformation.

The expected result is achieved with the three objects as shown in Fig. 6.4, Fig. 6.5, and Fig. 6.6. The fluid tends to keep its original shape and has a gelatinous appearance. Also, the fluid bounces back to the air because the solid-fluid coupling is not being considered.

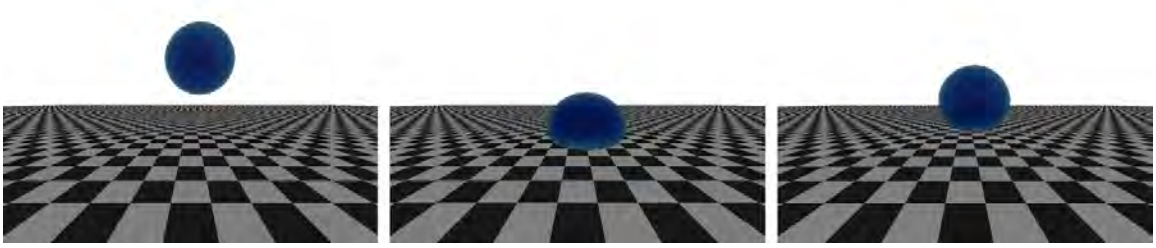

Figure 6.4: Sphere test case results.

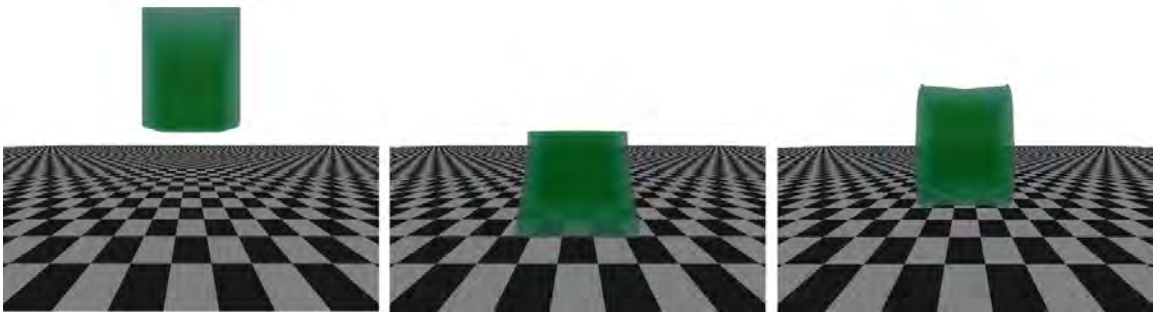

Figure 6.5: Square test case results.

When the one-way solid-fluid coupling is computed, the fluid does not bounce back and sticks on the boundary. This behavior is shown in Fig. 6.7 and in Fig. 6.8 in which the fluid particles stick on the boundary but is still affected by the gravitational force. This behavior can be seen at the tip of the pyramid which tends to bend in the gravity direction.

The influence of the correction coefficient (c) was analyzed. As the coefficient value decreases, less elastic the fluid becomes and spreads on the floor losing its original shape. This

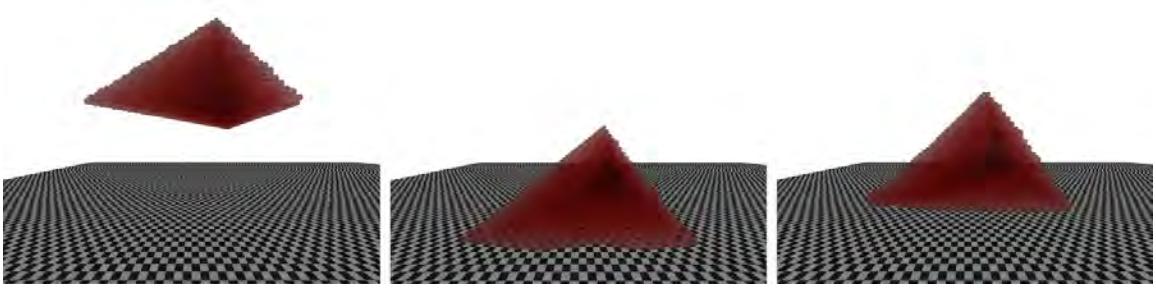

Figure 6.6: Pyramid test case results.

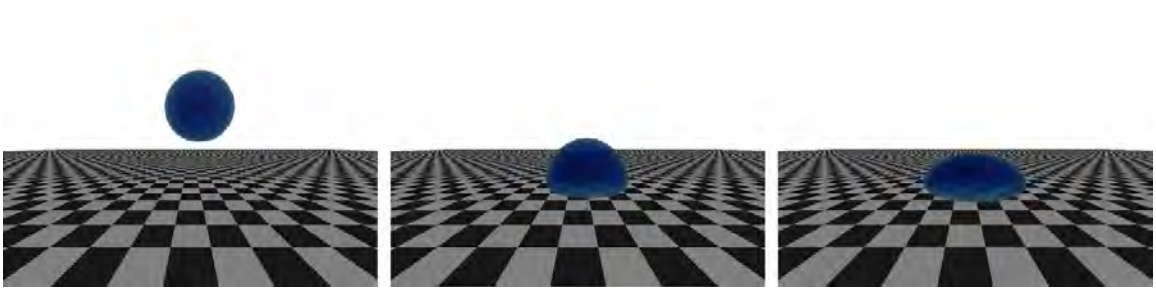

Figure 6.7: Sphere test case with one-way solid-fluid coupling.

result can be seen in Fig. 6.9, that shows the three different fluids, with the same original shape but different correction coefficients, after hitting the floor.

The achieved results are visually coherent and resemble a gelatin, being visually similar to the one found in the work of Takahashi et al. (TAKAHASHI et al., 2016). Also, it is possible to handle large deformations, different from the work of Mao and Yang (MAO; YANG, 2006).

The full visualization of the viscoelastic simulation results can be seen at `https://www.youtube.com/watch?v=Ap0eaPaBzIo`.

6.3.2 Performance Analysis

Table 6.3 shows the simulation times of the parallel CPU implementation using OpenMP and the parallel GPU implementation using CUDA for different numbers of particles. The time in the table represents the computation time of calculating particles interactions, new particle positions, and ordering the arrays according to their cell.

From the aforementioned results, it is possible to notice that the GPU version provides

Table 6.3: CPU and GPU computation times and their respective speedups for each test case.

# Particles	CPU (ms)	GPU (ms)	Speedup
$100k$	461	64	7.2
$300k$	1544	196	7.8
$500k$	2579	325	7.9
$750k$	3956	495	7.9
$1M$	5368	670	8.0

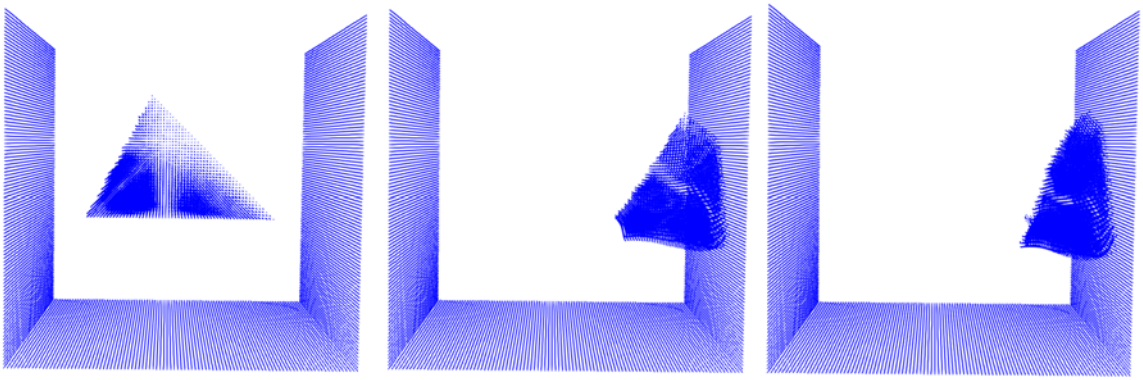

Figure 6.8: Pyramid test case with one way solid-fluid coupling and $v_0 = (10,0,0)$.

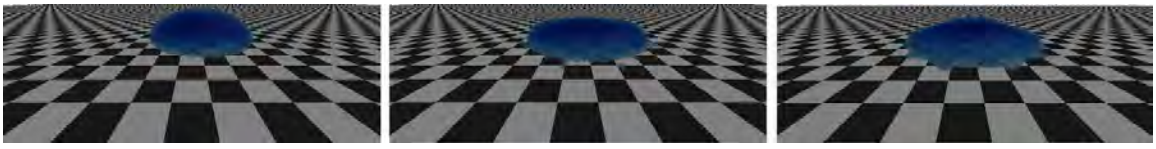

Figure 6.9: Sphere test case results. Left: $c = 0.01$; Middle: $c = 0.001$; Right: $c = 0.0005$.

an average of 7.76 speedup in comparison to the parallel CPU implementation.

Two main calculations occur in a time step: solving the new particle position and reordering the arrays using the particle cell position. In both GPU and CPU implementations, the reordering calculation takes less than 1% of the computation time. But, in the GPU version, the computation time increases from 1ms to 3ms, while in the CPU version increases from 3ms to 46ms when compared using $100k$ and $1M$ particles, respectively.

The work of Takahashi et al. (TAKAHASHI et al., 2016) can solve the viscoelastic SPH in $10s$/step in a simulation of $110.8k$ particles. The proposed solution in this work presents a faster simulation being able to solve a simulation with $100k$ particles in 64ms (15 fps) and also to simulate 1M particles in 670ms which is a better performance than the work of Takahashi et al. But, unlike the work of Takahashi et al., our work is not able to deal with connection control and a full solid-fluid coupling.

6.4 Ray Tracing Rendering

The rendering technique will be analyzed regarding visual quality and performance (fps). The results will be compared with the one found in the work of Xiao et al. (XIAO; ZHANG; YANG, 2017), which can achieve a performance of 8 fps for approximately 2M particles and a resolution of 1920 x 1920.

6.4.1 Visual Quality

Visually, the technique can present high-quality results. Similar to the work of Xiao et al. (XIAO; ZHANG; YANG, 2017), the smoothing method can preserve sharp features like the corners of the cube, as can be seen in Fig. 6.10 and, as stated in the work of van der Laan et al. (LAAN; GREEN; SAINZ, 2009), this smoothing method produces better results than the Bilateral Gaussian Smoothing and Gaussian Blur.

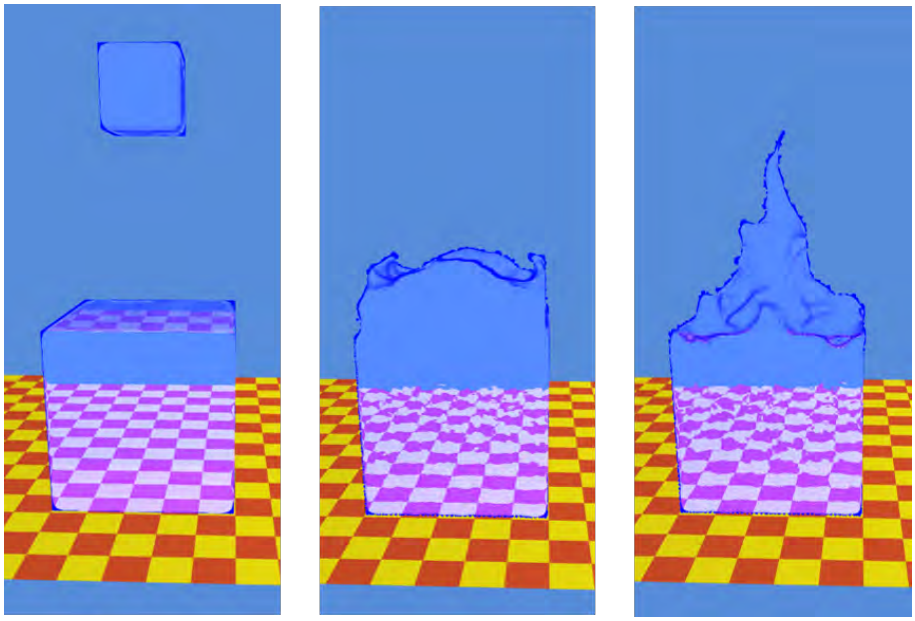

Figure 6.10: Rendering of the water drop scene.

The technique can render a Ray Tracing based reflection and refraction of the ambient as can be seen in Fig. 6.10 and Fig. 6.11. But the method is only capable of rendering a single layer of the fluid, which is not physically accurate and also is not capable of visualizing the fluid internal volume.

Different from the work of Xiao et al. (XIAO; ZHANG; YANG, 2017), which depends on the particle mass to reconstruct the iso-surface, the proposed method only relies on the sphere created using the particle's position, so it can be integrated into many particle-based methods, such as SPH, Particles-In-Cell (PIC), and the massless method MPS, without the need to modify any step of the rendering.

6.4.2 Performance

To find the best visual result with high performance, several numbers of iterations on the smoothing step were tested and, both, visual quality and fps were compared using the drop water test case, and the results can be found in Fig. 6.12 and Table 6.4.

The smoothing process was done with 50 iterations, and the final solution can render the fluid composed by 500k particles at 22 fps for a 1000 x 1000 resolution, which is the same resolution from the work of Xiao et al. (XIAO; ZHANG; YANG, 2017).

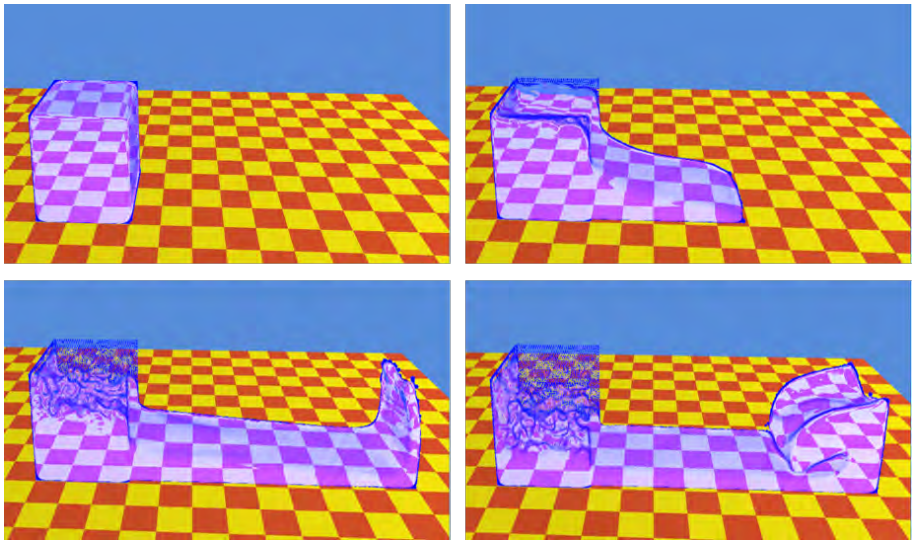

Figure 6.11: Rendering of the dam break scene.

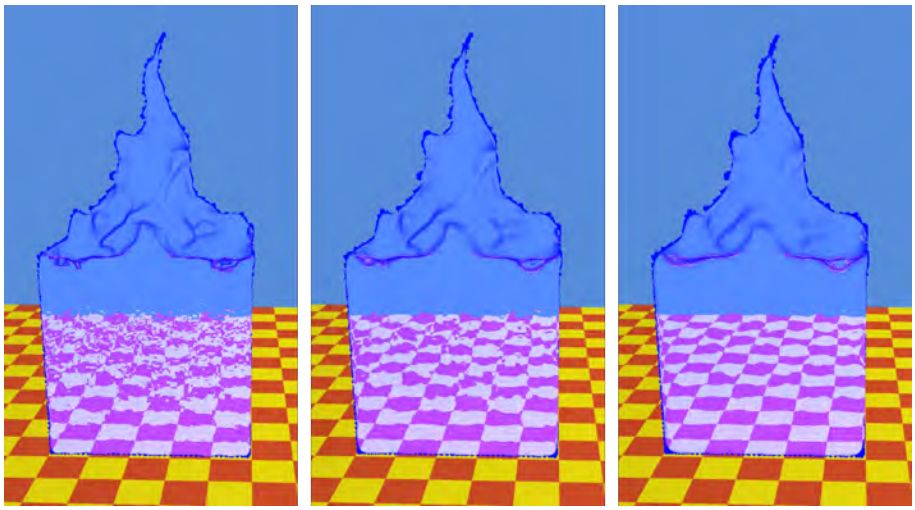

Figure 6.12: Rendering with different smoothing iterations: 25 (left), 50 (middle), and 100 (right).

To analyze the performance of the rendering, several tests were performed varying the image resolution, number of particles and, for static and dynamic scenes, as can be seen in Table 6.5. As expected, as the resolution value or the number of particles increases, the performance of the system gets worse. But for the dynamic scene, in which the acceleration structure is constantly reconstructed, the initial results indicate that the performance is kept almost constant with a mean value of 3.5 fps; but this scenario needs more evaluation to understand its behavior.

In comparison with the work of Xiao et al. (XIAO; ZHANG; YANG, 2017), our work has a worse performance, but the first results show that the performance is apparently constant for any number of particles or resolution, while in the work of Xiao et al. the performance drops for more complex scenarios, but more evaluation must be done to be certain of this result. Also, the work of Xiao et al. (XIAO; ZHANG; YANG, 2017) uses the particle mass to reconstruct the surface while our method only uses the particle position, being able to render massless methods

Table 6.4: Variations on the fps given the number of smoothing iterations.

# iterations	fps
25	34
50	22
100	13

Table 6.5: Variations on the fps given image resolution and number of particles for static and dynamic scenes.

	750k static	750k dynamic	1M static	1M dynamic	2M static	2M dynamic
1024x1024	23	5	17	4	18	3
1440x1440	17	5	17	4	12	2
1920x1920	11	4	8	3	8	2

such as the MPS without any modification.

7

CONCLUSION

This work has investigated the SPH method for large multiphase and viscoelastic flows but also how to render the fluid, balancing computational performance and the visual quality.

The WCSPH proposed by Vieira-e-Silva, Brito et al. (SILVA et al., 2015) was extended to simulate multiphase flow by modifying the density calculation as proposed by Solenthaler and Pajarola (SOLENTHALER; PAJAROLA, 2008). The simulation shows a realistic behavior as the denser liquid pushes the lighter fluid and XSPH was able to create realistic boundary conditions. But the simulation took a long time to converge.

The method was also extended to simulate viscoelastic flows by calculating a velocity correction proposed by Takahashi et al. (TAKAHASHI et al., 2016). Using this correction, the fluid tends to keep its original shape and has a gelatinous appearance. Differently from the work of Takahashi et al. (TAKAHASHI et al., 2016), our solution is computed on GPU and can solve a simulation with 100k particles in 64ms (15 fps) and $1M$ particles in 670ms which is a better performance than the work of Takahashi et al. But, our solution is not able to deal with connection control and a full solid-fluid coupling.

To render the multiphase fluid, a solution was proposed based on the work of van der Laan et al. (LAAN; GREEN; SAINZ, 2009), which can render the fluid in real time and 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 OpenGL Shading Language (GLSL) and has the particle position as primitive, so, it can be used with other particle-based simulation methods, such as MPS (SILVA et al., 2017) and PIC (JIANG et al., 2015). The approach has limitations, such as dealing only with two fluids, only supporting non-miscible fluids and the refraction is not realistic.

To achieve a more realistic render solution, a Ray Tracing based solution was proposed that has the particle position as input and can be used with any particle-based fluid simulation method. The solution reconstructs the fluid surface using the smoothing method proposed by van der Laan et al. (LAAN; GREEN; SAINZ, 2009) and the solution can render the reflection and refraction of the environment. The first results indicate that the proposed solution can render a dynamic scene at 3 fps independently of the number of particles but it needs more evaluation and the method can render massless methods such as the MPS without any modification.

7.1 Publications

During the Master degree, three works were published:

- Full paper published in the XVI Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2017 - Computing Track) about viscoelastic fluid simulation (BRITO et al., 2017).
- Full paper published in the Symposium on Virtual and Augmented Reality (SVR 2017) about simulation and rendering multiphase flow (BRITO et al., 2017).
- Co-author in the full paper published in the Symposium on Virtual and Augmented Reality (SVR 2017) about simulation for incompressible fluids on GPU (SILVA et al., 2017).
- Poster published in the Special Interest Group on Computer GRAPHICS and Interactive Techniques (SIGGRAPH) conference about augmentation of surfaces (BRITO et al., 2016).

7.2 Future Work

This work can be improved in many ways. For the multiphase SPH model, as the fluid took a lot of time to converge, an investigation can be done to find an approach that can solve the boundary condition more efficiently. For instance, Chen et al. (CHEN et al., 2015) proposed density re-initialization and a force based boundary condition being able to achieve a more stable pressure profile and a higher numerical accuracy.

A new force can be added to be able to control interface tension and to perform a more realistic simulation (SOLENTHALER; PAJAROLA, 2008). Then, the model can be used for coastal and other hydraulic applications to validate it in a real-world problem (VIOLEAU; ROGERS, 2016) and be compared numerically with theoretical or experimental results.

The viscoelastic SPH model can be improved to deal with connection control being able to simulate the interaction between two different fluids and splitting behavior (TAKAHASHI et al., 2016). Another possibility is to compare the numerical solution with other works, such as (TAKAHASHI et al., 2016) and (MAO; YANG, 2006).

Different particle-based methods can be implemented using the viscoelastic approach and the GPU code, such as ISPH (XU; STANSBY; LAURENCE, 2009) and MPS (KOSHIZUKA; OKA, 1996) (DUAN et al., 2017). As these methods simulate incompressible fluids, that preserve the fluid volume, the simulation can be more stable and numerically accurate, but will have a high computational cost because it solves a linear system.

The rendering solution can also be improved. First, the multiphase solution should be able to render more than two fluids. A more accurate blur function can be used such as the

one proposed in the work of Reichl et al. (REICHL et al., 2014). For the Ray Tracing based solution, more evaluation must be done to be certain that performance is practically constant for any number of particles and an investigation must be done to improve the memory consumption and computational performance. The solution can be improved to render fluid internal volume similar to the work of Zirr and Dachsbacher (ZIRR; DACHSBACHER, 2015) and, finally, a full Ray Tracing algorithm can be integrated in the solution being able to render the scene with global illumination.

References

AKINCI, G. et al. An Efficient Surface Reconstruction Pipeline for Particle-Based Fluids. In: WORKSHOP ON VIRTUAL REALITY INTERACTION AND PHYSICAL SIMULATION. The Eurographics Association, 2012.

AKINCI, G. et al. Parallel Surface Reconstruction for Particle-Based Fluids. In: COMPUTER GRAPHICS FORUM. 2012. volume 31, number 6, pages 1797–1809.

AKINCI, N. et al. Screen space foam rendering. **Journal of WSCG**, volume 21, 2013.

ANDREA, C. **A meshless lagrangian method for free-surface flows and interface flows with fragmentation**. 2005. Tese (Doutorado em Ciência da Computação) — PhD thesis, Universita di Roma La Sapienza.

APPEL, A. Some techniques for shading machine renderings of solids. In: PROCEEDINGS OF THE APRIL 30–MAY 2, 1968, SPRING JOINT COMPUTER CONFERENCE. 1968. pages 37–45.

ASAI, M. et al. A stabilized incompressible SPH method by relaxing the density invariance condition. **Journal of Applied Mathematics**, volume 2012, 6 2012.

ATAIE-ASHTIANI, B.; FARHADI, L. A stable moving-particle semi-implicit method for free surface flows. **Fluid Dynamics Research**, volume 38, number 4, pages 241 – 256, 2006.

BECKER, M.; IHMSEN, M.; TESCHNER, M. Corotated SPH for Deformable Solids. In: PROCEEDINGS OF THE FIFTH EUROGRAPHICS CONFERENCE ON NATURAL PHENOMENA, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association, 2009. pages 27–34. (NPH'09).

BELYTSCHKO, T. et al. Meshless methods: an overview and recent developments. **Computer methods in applied mechanics and engineering**, volume 139, number 1, pages 3–47, 1996.

BLINN, J. F. A generalization of algebraic surface drawing. **ACM Transactions on Graphics (TOG)**, volume 1, number 3, pages 235–256, 1982.

BØCKMANN, A.; SHIPILOVA, O.; SKEIE, G. Incompressible SPH for free surface flows. **Computers & Fluids**, volume 67, pages 138–151, 2012.

BONET, J.; KULASEGARAM, S. A simplified approach to enhance the performance of smooth particle hydrodynamics methods. **Applied Mathematics and Computation**, volume 126, number 2, pages 133–155, 2002.

BONET, J.; LOK, T.-S. Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. **Computer Methods in applied mechanics and engineering**, volume 180, number 1, pages 97–115, 1999.

BRITO, C. et al. Multimodal augmentation of surfaces using conductive 3D printing. In: ACM SIGGRAPH 2016 POSTERS. 2016. pages 15.

BRITO, C. et al. Screen Space Rendering Solution for Multiphase SPH Simulation. In: SYMPOSIUM ON VIRTUAL AND AUGMENTED REALITY (SVR), 2017. 2017. pages 309–318.

BRITO, C. et al. Large Viscoelastic Fluid Simulation on GPU. In: GAMES AND DIGITAL ENTERTAINMENT (SBGAMES), 2017 XVI BRAZILIAN SYMPOSIUM ON. 2017.

BROWN, D. L.; CORTEZ, R.; MINION, M. L. Accurate projection methods for the incompressible Navier–Stokes equations. **Journal of computational physics**, volume 168, number 2, pages 464–499, 2001.

CHANTASIRIWAN, S. Performance of multiquadric collocation method in solving lid-driven cavity flow problem with low Reynolds number. **COMPUTER MODELING IN ENGINEERING AND SCIENCES**, volume 15, number 3, pages 137, 2006.

CHEN, J.; YANG, K.; YUAN, Y. SPH-based visual simulation of fluid. In: COMPUTER SCIENCE & EDUCATION, 2009. ICCSE'09. 4TH INTERNATIONAL CONFERENCE ON. 2009. pages 690–693.

CHEN, Y.; LEE, J.; ESKANDARIAN, A. **Meshless methods in solid mechanics**. Springer Science & Business Media, 2006.

CHEN, Z. et al. An SPH model for multiphase flows with complex interfaces and large density differences. **Journal of Computational Physics**, volume 283, pages 169–188, 2015.

CIRIO, G. et al. Six degrees-of-freedom haptic interaction with fluids. **IEEE Transactions on Visualization and Computer Graphics**, volume 17, number 11, pages 1714–1727, 2011.

CIRIO, G. et al. "Tap, squeeze and stir" the virtual world: touching the different states of matter through 6dof haptic interaction. In: VIRTUAL REALITY CONFERENCE (VR), 2011 IEEE. 2011. pages 123–126.

CIRIO, G. et al. Six-oof haptic interaction with fluids, solids, and their transitions. In: WORLD HAPTICS CONFERENCE (WHC), 2013. 2013. pages 157–162.

CLAVET, S.; BEAUDOIN, P.; POULIN, P. Particle-based viscoelastic fluid simulation. In: PROCEEDINGS OF THE 2005 ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. 2005. pages 219–228.

CRESPO, A. et al. GPUs, a new tool of acceleration in CFD: efficiency and reliability on smoothed particle hydrodynamics methods. **PLoS One**, volume 6, number 6, pages e20685, 2011.

CRESPO, A. et al. DualSPHysics: open-source parallel {CFD} solver based on smoothed particle hydrodynamics (sph). **Computer Physics Communications**, volume 187, pages 204 – 216, 2015.

CRESPO, A. J. C. **Application of the smoothed particle hydrodynamics model SPHysics to free surface hydrodynamics**. 2008. Tese (Doutorado em Ciência da Computação) — Ph. D. thesis, University of De Vigo.

CUMMINS, S. J.; RUDMAN, M. An SPH projection method. **Journal of computational physics**, volume 152, number 2, pages 584–607, 1999.

DEHNEN, W.; ALY, H. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. **Monthly Notices of the Royal Astronomical Society**, volume 425, number 2, pages 1068–1082, 2012.

DOMÍNGUEZ, J. M. et al. Neighbour lists in smoothed particle hydrodynamics. **International Journal for Numerical Methods in Fluids**, volume 67, number 12, pages 2026–2042, 2011.

DUAN, G. et al. A multiphase MPS solver for modeling multi-fluid interaction with free surface and its application in oil spill. **Computer Methods in Applied Mechanics and Engineering**, volume 320, pages 133 – 161, 2017.

ENRIGHT, D. et al. A hybrid particle level set method for improved interface capturing. **Journal of Computational physics**, volume 183, number 1, pages 83–116, 2002.

EPIC GAMES. **Unreal Engine - Water in Unity**. Online; accessed 12-February-2018, `https://docs.unrealengine.com/latest/INT/Resources/Showcases/Effects/WaterExamples/`.

EPIC GAMES. **Unreal Engine - Cascade Particle Systems**. Online; accessed 12-February-2018, `https://docs.unrealengine.com/latest/INT/Engine/Rendering/ParticleSystems/index.html`.

FANG, J. et al. Improved SPH methods for simulating free surface flows of viscous fluids. **Applied Numerical Mathematics**, volume 59, number 2, pages 251–271, 2009.

FRAEDRICH, R.; AUER, S.; WESTERMANN, R. Efficient high-quality volume rendering of SPH data. **Visualization and Computer Graphics, IEEE Transactions on**, volume 16, number 6, pages 1533–1540, 2010.

FRANCI, A.; CREMONESI, M. On the effect of standard PFEM remeshing on volume conservation in free-surface fluid flow problems. **Computational Particle Mechanics**, volume 4, number 3, pages 331–343, 2017.

GHASEMI V, A.; FIROOZABADI, B.; MAHDINIA, M. 2D numerical simulation of density currents using the SPH projection method. **European journal of mechanics. B, Fluids**, volume 38, pages 38–46, 2013.

GINGOLD, R. A.; MONAGHAN, J. J. Smoothed particle hydrodynamics: theory and application to non-spherical stars. **Monthly notices of the royal astronomical society**, volume 181, number 3, pages 375–389, 1977.

HARADA, T.; KOSHIZUKA, S.; KAWAGUCHI, Y. Improvement in the boundary conditions of smoothed particle hydrodynamics. **Computer Graphics & Geometry**, volume 9, number 3, pages 2–15, 2007.

HARADA, T.; KOSHIZUKA, S.; KAWAGUCHI, Y. Smoothed particle hydrodynamics on GPUs. In: COMPUTER GRAPHICS INTERNATIONAL. 2007. pages 63–70.

HECK, T. et al. Modeling extracellular matrix viscoelasticity using smoothed particle hydrodynamics with improved boundary treatment. **Computer Methods in Applied Mechanics and Engineering**, volume 322, pages 515 – 540, 2017.

HORI, C. et al. GPU-acceleration for moving particle semi-implicit method. **Computers & Fluids**, volume 51, number 1, pages 174–183, 2011.

HORVATH, C. J.; SOLENTHALER, B. Mass preserving multi-scale SPH. **Pixar Technical Memo 13-04, Pixar Animation Studios**, 2013.

HOSSEINI, S. M.; FENG, J. J. Pressure boundary conditions for computing incompressible flows with SPH. **Journal of Computational physics**, volume 230, number 19, pages 7473–7487, 2011.

HU, X.; ADAMS, N. A constant-density approach for incompressible multi-phase SPH. **Journal of Computational Physics**, volume 228, number 6, pages 2082–2091, 2009.

HU, X. Y.; ADAMS, N. A. A multi-phase SPH method for macroscopic and mesoscopic flows. **Journal of Computational Physics**, volume 213, number 2, pages 844–861, 2006.

IHMSEN, M. et al. SPH Fluids in Computer Graphics. In: EUROGRAPHICS 2014 - STATE OF THE ART REPORTS. The Eurographics Association, 2014.

INC, T. K. G. **OpenGL SDK**. URL: https://www.opengl.org/sdk/.

JIANG, C. et al. The affine particle-in-cell method. **ACM Transactions on Graphics (TOG)**, volume 34, number 4, pages 51, 2015.

JUNIOR, J. R. d. S. et al. Fluid simulation with two-way interaction rigid body using a heterogeneous GPU and CPU environment. In: GAMES AND DIGITAL ENTERTAINMENT (SBGAMES), 2010 BRAZILIAN SYMPOSIUM ON. 2010. pages 156–164.

JUNIOR, J. R. d. S. et al. An architecture for real time fluid simulation using multiple GPUs. **XI SBGames, Brasília**, pages 8, 2012.

KESSLER, J.; CARABAICH, P.; KINOSHITA, N. Fluid dynamics and lighting implementation in PixelJunk Shooter 2. In: ACM SIGGRAPH 2011 TALKS. 2011. pages 70.

KIARA, A.; HENDRICKSON, K.; YUE, D. K. SPH for incompressible free-surface flows. Part I: error analysis of the basic assumptions. **Computers & Fluids**, volume 86, pages 611–624, 2013.

KONDO, M. et al. Incompressible Free Surface Flow Analysis Using Moving Particle Semi-Implicit Method. In: JOINT INTERNATIONAL WORKSHOP: NUCLEAR TECHNOLOGY AND SOCIETY–NEEDS FOR NEXT GENERATION, BERKELEY, CALIFORNIA. 2008. pages 6–8.

KOSHIZUKA, S.; NOBE, A.; OKA, Y. Numerical analysis of breaking waves using the moving particle semi-implicit method. **International Journal for Numerical Methods in Fluids**, volume 26, number 7, pages 751–769, 1998.

KOSHIZUKA, S.; OKA, Y. Moving-particle semi-implicit method for fragmentation of incompressible fluid. **Nuclear science and engineering**, volume 123, number 3, pages 421–434, 1996.

KROG, Ø. E.; ELSTER, A. C. Fast gpu-based fluid simulations using sph. In: **Applied Parallel and Scientific Computing**. Springer, 2012. pages 98–109.

LAAN, W. J. van der; GREEN, S.; SAINZ, M. Screen space fluid rendering with curvature flow. In: PROCEEDINGS OF THE 2009 SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES. 2009. pages 91–98.

LASTIWKA, M.; BASA, M.; QUINLAN, N. J. Permeable and non-reflecting boundary conditions in SPH. , 2009.

LAUTERBACH, C. et al. Fast BVH construction on GPUs. In: COMPUTER GRAPHICS FORUM. 2009. volume 28, number 2, pages 375–384.

LEE, B.-H. et al. Step-by-step improvement of MPS method in simulating violent free-surface motions and impact-loads. **Computer methods in applied mechanics and engineering**, volume 200, number 9, pages 1113–1125, 2011.

LEE, E.-S. et al. Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. **Journal of computational physics**, volume 227, number 18, pages 8417–8436, 2008.

LEE, E.-S. et al. Application of weakly compressible and truly incompressible SPH to 3-D water collapse in waterworks. **Journal of Hydraulic Research**, volume 48, number S1, pages 50–60, 2010.

LEE, W.-J. et al. SGRT: a mobile gpu architecture for real-time ray tracing. In: PROCEEDINGS OF THE 5TH HIGH-PERFORMANCE GRAPHICS CONFERENCE. 2013. pages 109–119.

LIU, M.; LIU, G. Smoothed particle hydrodynamics (SPH): an overview and recent developments. **Archives of computational methods in engineering**, volume 17, number 1, pages 25–76, 2010.

LIU, X. et al. 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. 2015. pages 1595–1598.

LORENSEN, W. E.; CLINE, H. E. Marching cubes: a high resolution 3d surface construction algorithm. In: ACM SIGGRAPH COMPUTER GRAPHICS. 1987. volume 21, number 4, pages 163–169.

LUCY, L. B. A numerical approach to the testing of the fission hypothesis. **The astronomical journal**, volume 82, pages 1013–1024, 1977.

MALLADI, R.; SETHIAN, J. A. Level set methods for curvature flow, image enchancement, and shape recovery in medical images. In: **Visualization and mathematics**. Springer, 1997. pages 329–345.

MAO, H.; YANG, Y.-H. Particle-based non-Newtonian fluid animation with heating effects. **University of Alberta, Tech. Rep**, 2006.

MARRONE, S. et al. δ-SPH model for simulating violent impact flows. **Computer Methods in Applied Mechanics and Engineering**, volume 200, number 13-16, pages 1526–1542, 2011.

MEISTER, M.; BURGER, G.; RAUCH, W. On the Reynolds number sensitivity of smoothed particle hydrodynamics. **Journal of Hydraulic Research**, volume 52, number 6, pages 824–835, 2014.

MILLER, G.; PEARCE, A. Globular dynamics: a connected particle system for animating viscous fluids. **Computers & Graphics**, volume 13, number 3, pages 305–309, 1989.

MOKOS, A. et al. Multi-phase SPH modelling of violent hydrodynamics on GPUs. **Computer Physics Communications**, volume 196, pages 304–316, 2015.

MOKOS, A.; ROGERS, B. D.; STANSBY, P. K. A multi-phase particle shifting algorithm for SPH simulations of violent hydrodynamics with a large number of particles. **Journal of Hydraulic Research**, volume 55, number 2, pages 143–162, 2017.

MONAGHAN, J. J. Simulating free surface flows with SPH. **Journal of computational physics**, volume 110, number 2, pages 399–406, 1994.

MONAGHAN, J. J. SPH without a tensile instability. **Journal of Computational Physics**, volume 159, number 2, pages 290–311, 2000.

MONAGHAN, J. J. Smoothed particle hydrodynamics. **Reports on progress in physics**, volume 68, number 8, pages 1703, 2005.

MONAGHAN, J. J.; KAJTAR, J. B. SPH particle boundary forces for arbitrary boundaries. **Computer Physics Communications**, volume 180, number 10, pages 1811–1820, 2009.

MONAGHAN, J.; KOCHARYAN, A. SPH simulation of multi-phase flow. **Computer Physics Communications**, volume 87, number 1-2, pages 225–235, 1995.

MONAGHAN, J.; RAFIEE, A. A simple SPH algorithm for multi-fluid flow with high density ratios. **International Journal for Numerical Methods in Fluids**, volume 71, number 5, pages 537–561, 2013.

MORRIS, J. P.; FOX, P. J.; ZHU, Y. Modeling low Reynolds number incompressible flows using SPH. **Journal of computational physics**, volume 136, number 1, pages 214–226, 1997.

MÜLLER, M.; CHARYPAR, D.; GROSS, M. Particle-based fluid simulation for interactive applications. In: PROCEEDINGS OF THE 2003 ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. 2003. pages 154–159.

MÜLLER, M. et al. Point based animation of elastic, plastic and melting objects. In: PROCEEDINGS OF THE 2004 ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. 2004. pages 141–151.

MÜLLER, M. et al. Particle-based fluid-fluid interaction. In: PROCEEDINGS OF THE 2005 ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. 2005. pages 237–244.

MÜLLER, M. et al. Position based dynamics. **Journal of Visual Communication and Image Representation**, volume 18, number 2, pages 109–118, 2007.

MÜLLER, M.; SCHIRM, S.; DUTHALER, S. Screen space meshes. In: PROCEEDINGS OF THE 2007 ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. 2007. pages 9–15.

NAH, J.-H. et al. HART: a hybrid architecture for ray tracing animated scenes. **Visualization and Computer Graphics, IEEE Transactions on**, volume 21, number 3, pages 389–401, 2015.

NEXT LIMIT TECHNOLOGIES. **RealFlow**. Online; accessed 12-February-2018, `http://www.nextlimit.com/realflow/`.

NEXT LIMIT TECHNOLOGIES. **RealFlow 10 Documentation - Particles**: liquid - pbd. Online; accessed 12-February-2018, `http://support.nextlimit.com/display/rf2016docs/DyDomain+-+Particles%3A+Liquid+-+PBD`.

NEXT LIMIT TECHNOLOGIES. **RealFlow 10 Documentation - Particles**: liquid - sph. Online; accessed 12-February-2018, `http://support.nextlimit.com/display/rf2016docs/DyDomain+-+Particles%3A+Liquid+-+SPH`.

NVIDIA. **NVIDIA CUDA Toolkit Documentation**. Online; accessed 12-February-2018, `http://docs.nvidia.com/cuda/`.

NVIDIA. **CUDA Zone | NVIDIA Developer**. 2018.

OPENMP ARCHITECTURE REVIEW BOARD. **OpenMP Application Programming Interface**. Online; accessed 12-February-2018, `http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf`.

ORTHMANN, J. et al. Consistent surface model for SPH-based fluid transport. In: PROCEEDINGS OF THE 12TH ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. 2013. pages 95–103.

PAN, X.-j.; ZHANG, H.-x.; SUN, X.-y. Numerical simulation of sloshing with large deforming free surface by MPS-LES method. **China Ocean Engineering**, volume 26, pages 653–668, 2012.

PANG, W.-M. et al. Fast prototyping of virtual reality based surgical simulators with PhysX-enabled GPU. **Transactions on edutainment IV**, pages 176–188, 2010.

PARKER, S. G. et al. Optix: a general purpose ray tracing engine. **ACM Transactions on Graphics (TOG)**, volume 29, number 4, pages 66, 2010.

PENG, C. et al. Multiphase SPH modeling of free surface flow in porous media with variable porosity. **Computers and Geotechnics**, volume 81, pages 239–248, 2017.

POZORSKI, J.; WAWREŃCZUK, A. SPH computation of incompressible viscous flows. **Journal of Theoretical and Applied Mechanics**, volume 40, number 4, pages 917–937, 2002.

PREMŽOE, S. et al. Particle-Based Simulation of Fluids. In: COMPUTER GRAPHICS FORUM. 2003. volume 22, number 3, pages 401–410.

PRICE, D. J. Resolving high Reynolds numbers in SPH simulations of subsonic turbulence. **arXiv preprint arXiv:1111.1255**, 2011.

RAFIEE, A.; MANZARI, M.; HOSSEINI, M. An incompressible SPH method for simulation of unsteady viscoelastic free-surface flows. **International Journal of Non-Linear Mechanics**, volume 42, number 10, pages 1210–1223, 2007.

REICHL, F. et al. Interactive Rendering of Giga-Particle Fluid Simulations. In: EUROGRAPHICS/ACM SIGGRAPH SYMPOSIUM ON HIGH PERFORMANCE GRAPHICS. 2014. pages 105–116.

ROGERS, B. D. et al. 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.

SCHECHTER, H.; BRIDSON, R. Ghost SPH for animating water. **ACM Transactions on Graphics (TOG)**, volume 31, number 4, pages 61, 2012.

SHADLOO, M. S. et al. A robust weakly compressible SPH method and its comparison with an incompressible SPH. **International Journal for Numerical Methods in Engineering**, volume 89, number 8, pages 939–956, 2012.

SIDE EFFECTS SOFTWARE. **Houdini FX**. Online; accessed 12-February-2018, `https://www.sidefx.com/products/houdini-fx/`.

SIDE EFFECTS SOFTWARE. **Houdini 16.5 - Fluids**. Online; accessed 12-February-2018, `http://www.sidefx.com/docs/houdini/fluid/index.html`.

SIGALOTTI, L. D. G. et al. SPH simulations of time-dependent Poiseuille flow at low Reynolds numbers. **Journal of computational physics**, volume 191, number 2, pages 622–638, 2003.

SILVA, A. L. B. Vieira-e et al. Improved Meshless Method for Simulating Incompressible Fluids on GPU. In: SYMPOSIUM ON VIRTUAL AND AUGMENTED REALITY (SVR), 2017. 2017. pages 297–308.

SILVA, A. V. e et al. A qualitative analysis of fluid simulation using a sph variation. In: PROCEEDINGS OF THE CONGRESS ON NUMERICAL METHODS IN ENGINEERING. 2015.

SINGH, J.; NARAYANAN, P. Real-Time Ray Tracing of Implicit Surfaces on the GPU. **Visualization and Computer Graphics, IEEE Transactions on**, volume 16, number 2, pages 261–272, March 2010.

SOLENTHALER, B.; PAJAROLA, R. Density contrast SPH interfaces. In: PROCEEDINGS OF THE 2008 ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. 2008. pages 211–218.

SOLENTHALER, B.; SCHLÄFLI, J.; PAJAROLA, R. A unified particle model for fluid–solid interactions. **Computer Animation and Virtual Worlds**, volume 18, number 1, pages 69–82, 2007.

SWEGLE, J.; HICKS, D.; ATTAWAY, S. Smoothed particle hydrodynamics stability analysis. **Journal of computational physics**, volume 116, number 1, pages 123–134, 1995.

SZEWC, K.; POZORSKI, J.; MINIER, J.-P. Analysis of the incompressibility constraint in the smoothed particle hydrodynamics method. **International Journal for Numerical Methods in Engineering**, volume 92, number 4, pages 343–369, 2012.

TAKAHASHI, T. et al. Volume preserving viscoelastic fluids with large deformations using position-based velocity corrections. **The Visual Computer**, volume 32, number 1, pages 57–66, 2016.

TAKAMATSU, K.; KANAI, T. A Fast and Practical Method for Animating Particle-Based Viscoelastic Fluids. **International Journal of Virtual Reality**, volume 10, number 1, pages 29 – 35, 2011.

TANAKA, M.; MASUNAGA, T. Stabilization and smoothing of pressure in MPS method by quasi-compressibility. **Journal of Computational Physics**, volume 229, number 11, pages 4279–4290, 2010.

TARTAKOVSKY, A. M.; MEAKIN, P. A smoothed particle hydrodynamics model for miscible flow in three-dimensional fractures and the two-dimensional Rayleigh–Taylor instability. **Journal of Computational Physics**, volume 207, number 2, pages 610–624, 2005.

TARTAKOVSKY, A. M.; MEAKIN, P. Pore scale modeling of immiscible and miscible fluid flows using smoothed particle hydrodynamics. **Advances in Water Resources**, volume 29, number 10, pages 1464–1478, 2006.

TERZOPOULOS, D.; PLATT, J.; FLEISCHER, K. Heating and melting deformable models. **Computer Animation and Virtual Worlds**, volume 2, number 2, pages 68–73, 1991.

TSURUTA, N.; KHAYYER, A.; GOTOH, H. A short note on Dynamic Stabilization of Moving Particle Semi-implicit method. **Computers & Fluids**, volume 82, pages 158–164, 2013.

UNITY TECHNOLOGIES. **Water in Unity**. Online; accessed 12-February-2018, `https://docs.unity3d.com/2018.1/Documentation/Manual/HOWTO-Water.html`.

VACONDIO, R. et al. Variable resolution for SPH in three dimensions: towards optimal splitting and coalescing for dynamic adaptivity. **Computer Methods in Applied Mechanics and Engineering**, volume 300, pages 442–460, 2016.

VIOLEAU, D.; ROGERS, B. D. Smoothed particle hydrodynamics (SPH) for free-surface flows: past, present and future. **Journal of Hydraulic Research**, volume 54, number 1, pages 1–26, 2016.

WANG, Z.; WANG, Y. Haptic interaction with fluid based on smooth particles and finite elements. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND ITS APPLICATIONS. 2014. pages 808–823.

WATKINS, S. et al. A new prescription for viscosity in smoothed particle hydrodynamics. **Astronomy and Astrophysics Supplement Series**, volume 119, number 1, pages 177–187, 1996.

WHITTED, T. An improved illumination model for shaded display. In: ACM SIGGRAPH 2005 COURSES. 2005. pages 4.

WILCOX, D. C. et al. **Turbulence modeling for CFD**. DCW industries La Canada, CA, 1998. volume 2.

XIAO, X.; ZHANG, S.; YANG, X. 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. 2017. pages 12.

XU, R.; STANSBY, P.; LAURENCE, D. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. **Journal of Computational Physics**, volume 228, number 18, pages 6703–6725, 2009.

XU, X.; DENG, X.-L. An improved weakly compressible SPH method for simulating free surface flows of viscous and viscoelastic fluids. **Computer Physics Communications**, volume 201, pages 43 – 62, 2016.

XU, X.; YU, P. A multiscale SPH method for simulating transient viscoelastic flows using bead-spring chain model. **Journal of Non-Newtonian Fluid Mechanics**, volume 229, pages 27 – 42, 2016.

YAN, X. et al. Multiphase SPH simulation for interactive fluids and solids. **ACM Transactions on Graphics (TOG)**, volume 35, number 4, pages 79, 2016.

YANG, X.; LIU, M.; PENG, S. Smoothed particle hydrodynamics modeling of viscous liquid drop without tensile instability. **Computers & Fluids**, volume 92, pages 199–208, 2014.

YEH, T. Y.; FALOUTSOS, P.; REINMAN, G. Enabling Real-time Physics Simulation in Future Interactive Entertainment. In: PROCEEDINGS OF THE 2006 ACM SIGGRAPH SYMPOSIUM ON VIDEOGAMES, New York, NY, USA. ACM, 2006. pages 71–81. (Sandbox '06).

YU, J. et al. Explicit mesh surfaces for particle based fluids. In: COMPUTER GRAPHICS FORUM. 2012. volume 31, number 2pt4, pages 815–824.

YU, J.; TURK, G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. **ACM Transactions on Graphics (TOG)**, volume 32, number 1, pages 5, 2013.

ZAINALI, A. et al. Numerical investigation of Newtonian and non-Newtonian multiphase flows using ISPH method. **Computer Methods in Applied Mechanics and Engineering**, volume 254, pages 99–113, 2013.

ZHANG, Y.; SOLENTHALER, B.; PAJAROLA, R. Adaptive sampling and rendering of fluids on the GPU. In: PROCEEDINGS OF THE FIFTH EUROGRAPHICS/IEEE VGTC CONFERENCE ON POINT-BASED GRAPHICS. 2008. pages 137–146.

ZHU, X. et al. Implementation of the moving particle semi-implicit method on GPU. **SCIENCE CHINA Physics, Mechanics & Astronomy**, volume 54, number 3, pages 523–532, 2011.

ZIRR, T.; DACHSBACHER, C. Memory-Efficient On-The-Fly Voxelization of Particle Data. In: EGPGV. 2015. pages 11–18.