Institute for Visualization and Interactive Systems

Universität Stuttgart

INSTITUTE FOR VISUALIZATION AND INTERACTIVE SYSTEMS

SIMULATION TECHNOLOGY DEGREE COURSE

Master thesis

Mesh-based boundary handling using pressure forces

A new method for rigid body SPH boundary objects

First Supervisor

Prof. Dr. Daniel WEISKOPF

Institute for Visualization and Interactive Systems

Second Supervisor

Stefan REINHARDT M.A.

Institute for Visualization and Interactive Systems

Submitted by

Author	Etienne OTT
Matriculation number	3207635
SimTech-Nr.	48
Submission date	November 2018

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit mit dem Titel

Mesh-based boundary handling using pressure forces

selbständig verfasst habe, dass ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist, dass ich die Arbeit weder vollständig noch in Teilen bereits veröffentlicht habe und dass das elektronische Exemplar mit den anderen Exemplaren übereinstimmt.

Unterschrift: _____

Datum: _____

UNIVERSITÄT STUTTGART

Auszug

Faculty for Computer Science, Electrical Engineering and Information Technology
Institute for Visualization and Interactive Systems

Master of Science

Mesh-based boundary handling using pressure forces

von Etienne OTT

Wir präsentieren einen neuen Ansatz um Randobjekte in Fluid-Simulationen zu integrieren, welche mittels der Methode der Smoothed Particle Hydrodynamics durchgeführt werden. In unserer Methode werden Kollisions- bzw. Randobjekte als polygonales Mesh repräsentiert. Das Fluid interagiert direkt mit diesen. Hierzu verwenden wir jedoch keine Schnitttests und Reflektionen wie üblich. Wir korrigieren die Partikeldichte stattdessen durch ein kontinuierliches Feld. Weiter leiten wir daraus Druck- und Reibungskräfte her, welche auf die Partikel wirken, die mit dem Festkörper interagieren. Numerische Experimente, welche die Vorteile sowie auch die Schwächen unseres Modells hervorheben, werden durchgeführt. Weiter wird ein Vergleich mit einer weit verbreiteten Methode präsentiert. Im Fazit diskutieren wir unsere Methode und können schlussfolgern, dass sie Verbesserungen bezüglich mancher Anwendungsfälle benötigt, aber das Potential aufweist die Performance von gekoppelten Starrkörper-Fluid Simulationen zu erhöhen.

UNIVERSITÄT STUTTGART

Abstract

Faculty for Computer Science, Electrical Engineering and Information Technology
Institute for Visualization and Interactive Systems

Master of Science

Mesh-based boundary handling using pressure forces

by Etienne OTT

We present a new approach for incorporating boundary objects into fluid simulations performed with the method of Smoothed Particle Hydrodynamics. In our method collision and boundary objects are represented as polygonal meshes. The fluid interacts directly with these objects. However, unlike common approaches we do not employ intersection tests and reflections. Instead we correct the particle density through a continuous field. We further derive pressure and friction forces from this, which act on the particles that interact with the boundary object. Numerical experiments, which highlight the strengths and weaknesses of our method, are performed. Additionally, a comparison with a widely used method is presented. In the concluding remarks we discuss our method and come to the conclusion, that it requires improvements for some use cases, but has the potential to increase performance of coupled rigid-fluid simulations.

Table of Contents

List of Figures

Chapter 1

Introduction

Performing fluid simulations is a way to create realistic looking fluid animations without specifying its behavior manually and is a common approach in the field of Computer Graphics (CG). However, a simulation is only as suitable as the chosen model. To simulate fluids on a mesoscopic scale, the Navier-Stokes equations (compare Constantin and Foias [CF88] for an extensive overview) are commonly used to model the fluid's behavior. Based on these equations, different fluid models have been developed and while some approaches are not used anymore, many coexist due to their individual strengths and weaknesses.

These models can be classified as either using an Eulerian or a Lagrangian perspective. On the Eulerian side, typical approaches to solve differential equations numerically are for example Finite Differences or Finite Elements. Here, space is discretized into cells using a grid. The fluid properties, such as velocity or pressure, are calculated at fixed positions within each cell. They have been proven to work well within closed boundary conditions. However, they seem to have issues with open boundary conditions such as free surface flow. These issues might occur, for example, at the interface to air or other fluids.

In contrast, when using Lagrangian models the fluid is discretized by particle-like objects, that represent a certain amount of fluid. This has the advantage that interactions, particularly with boundary objects, are easier to handle as these interactions can be modeled to be force-like and can be included in the solver for the equations of motion. One popular Lagrangian method used in CG is called Smoothed Particle Hydrodynamics (SPH).

There exist several approaches to handle boundary objects when simulating a fluid using SPH. These approaches can be classified by how the boundary objects are represented. A boundary object, e.g. a ball, can be modeled by scattering particles on the surface or by a mesh or signed distance fields, etc. However, most methods require the geometric shape of the object to be given in a discrete form, for example approximated as a polygonal mesh. Meshes are commonly used in scientific computing and CG, in part because they integrate

well into other techniques and possess well defined properties[1]. Thus, they are suitable to be used for the geometric representation of boundary objects.

In this work, we investigate different boundary models and present one approach based on polygonal meshes. When employing triangular meshes as a boundary model, the interaction between the mesh and the fluid needs to be handled. Direct collision techniques can be used as well as forces acting on a particle basis. A force-based approach is usually used when treating the boundary objects as particles. We present a model that uses a triangular mesh but treats the interaction with pressure and friction forces. The pressure force prevents the fluid from penetrating the boundary object and the friction force includes surface roughness.

After the introduction we present recent work on boundary handling in SPH and the coupling of rigid-body and fluid simulations in chapter 3. Subsequently, in chapter 2 we provide background information about SPH itself and several techniques concerning meshes. In chapter 4 we discuss different boundary models and present our own. We compare our model with the model of Akinci et al. [AIA*12], which is further discussed in section 4.1, and present the results in chapter 6. Lastly, we discuss the strengths and weaknesses of our method as well as give a brief outlook on possible future improvements.

[1] In comparison, an object given in parametric form $\{x \in \mathbb{R}^3 \mid c(x) \leq 1\}$ also has clearly defined properties, but calculating them, e.g the volume, can be very complex, depending on its characteristic function c.

Chapter 2

Related Work

SPH was first introduced in 1977 by Lucy [Luc77] as well as Gingold and Monaghan [GM77] to simulate stellar objects. It proved to be useful at a time when computing resources were still slow compared to modern standards. The method was further developed to suit more general purposes in numerical fluid dynamics and was regularly revisited, e.g. by Monaghan [Mon92, Mon05, Mon12].

SPH gained attention in a large variety of use cases of fluid dynamics in part due to its extensibility and robustness (compare Monaghan [Mon92]). The method of SPH is also used in the field of CG for efficient simulation of fluids since the mid-1990s (by Stam and Fiume [SF95]) and has been subject of study in many more works. In the State of the Art report of Ihmsen et al. [IOS*14] an overview of the latest developments in the field of SPH is given and the specific improvements to techniques are discussed.

The task of integrating boundary objects into SPH simulations is of special interest to enable the simulation of more complex scenarios. Distance-penalty-based approaches can serve as simple methods to keep the fluid within the domain space, but don't adapt well for more complicated objects (see Ihmsen et al. [IOS*14] p. 9). In this work, we use a reflection method (described in chapter 5) to model the domain boundaries similar to the slip-boundaries introduced by Hu and Adams [HA06] in their work on multi-phase fluid simulations with SPH.

Boundaries inside the domain are usually handled differently. Akinci et al. [AIA*12] presented a method, that samples the boundary object with particles. They introduced a density correction of fluid particles and formulate a pressure-based force acting from these boundary particles on the fluid. The method of Schechter and Bridson [SB12] sampled the boundary objects in a similar manner. They also enclosed the free surface with particles to simulate the surrounding air phase. In addition to preventing boundary penetration, this also models the surface tension of the fluid.

It is also possible to model elastic bodies with these methods, for example in the simulation of cloth. Akinci et al. [ACAT13] presented an adapted method of their previous work, that takes deformable bodies into account. The method of Akinci et al. [AIA*12] is of particular interest due to its versatility and robustness and is summarized in section 4.1 in order to provide a comparison with a widely used method.

Boundary models avoiding the sampling of boundary object as particles have also been developed. Huber et al. [HEW15] presented a method for modeling thin elastic bodies represented as triangular meshes. This method employs both a repulsion force and continuous intersection handling to prevent penetration. The method of Harada et al. [HKK07] uses a similar repulsion force for rigid bodies given as triangular meshes. Because the repulsion force expects the density to be corrected by the boundary, they place temporary boundary particles onto the surface of the object to calculate the contribution of the boundary object to the fluid particle density.

Koschier and Bender [KB17] presented a method that avoids these temporary boundary particles altogether and is not limited to polygonal meshes. They calculate the intersection volume between a fluid particle's sphere of influence and the boundary object and use this volume, in combination with weight functions, to formulate the density correction and fluid-boundary forces. Because it shares important concepts with our method, it is summarized in section 4.2.

In section 4.3 we introduce modifications to the method of Koschier and Bender [KB17] for boundary objects modeled as triangular meshes. These modifications simplify the calculation of the weight functions for the locally planar surface geometry. Additionally, while the original method requires certain attributes to be calculated on the grid points of a sampling grid and therefore introduces a resolution parameter, out method does not require choosing a resolution other than the one implied by approximating curved surfaces by polygonal meshes.

Chapter 3

Background

Since we have discussed in chapter 2, that SPH is well established within the field CG, we would like to give a brief overview over several aspects in this section, which are necessary for the preparation of the method introduced in section 4.3.

3.1 SPH

SPH represents a fluid as a number of interacting particles, such that a continuous fluid quantity A can be calculated from a finite number of samples. The value of A at point $\mathbf{x}$ can be expressed as

$$A(\mathbf{x}) = \int_{\mathbb{R}^3} A(\mathbf{x}^*) \cdot W(\|\mathbf{x}-\mathbf{x}^*\|_2, h)\, d\mathbf{x}^*, \tag{3.1}$$

where W is a kernel function and h is the so-called smoothing length, both of which are further explained in section 3.1.1. Through the kernel function the contributions of other points $\mathbf{x}^*$ on the point $\mathbf{x}$ are limited to a h-neighborhood around $\mathbf{x}$.

In the following, the particles are denoted by i with $1 \leq i \leq N$, where N is the total number of particles. Each particle represents a certain volume of the fluid, which can be calculated by $\frac{m_i}{\rho_i}$ from their mass m_i and their density ρ_i. For equation 3.1 this means that a particle covers a continuous region of $\mathbb{R}^3$, which reduces the integration to a sum over the particle quantities. In particular, A at a particle position $\mathbf{x}_i$ can be expressed as

$$A(\mathbf{x}_i) = \sum_j \frac{m_j}{\rho_j} \cdot A(\mathbf{x}_j) \cdot W(\|\mathbf{x}_i - \mathbf{x}_j\|_2, h). \tag{3.2}$$

Because the kernel has a value of zero outside $[0, h]$, only particles in a neighborhood of $\mathbf{x}_i$ with radius h need to be considered. For a fixed fluid volume, the particle radius is usually

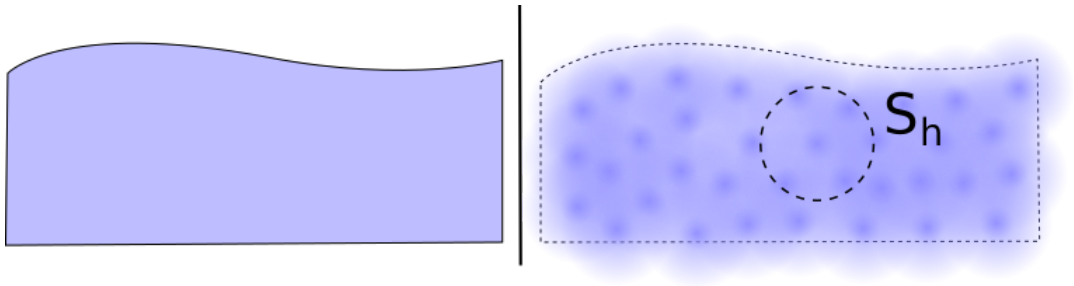

Figure 3.1: Left: A body of fluid in its original shape. Right: The same body of fluid discretized by 35 particles. A h-neighborhood of one particle is highlighted.

chosen smaller the bigger N is. Because h is also usually chosen as multiple of the particle radius, the number of other particles within the sphere of influence S_i^h of one particle is roughly constant over N.

Figure 3.1 shows a visualization of the discretization performed by SPH. A body of fluid is approximated by 35 particles that influence other particles. The sphere of influence S_i^h of one particle is highlighted and four other particles are within it.

As SPH has an inherently spatial basis, requiring many calculations on the position of particles and objects, it is beneficial to use data structures that support querying data from a particular region in the domain space (compare Samet [Sam90] p.17). Such spatial data structures (SDS) exist in a variety of forms, from simple cell grids and space-filling curves to balanced trees. Performance is greatly increased by placing the fluid particles and the boundary data into SDS.

3.1.1 Kernel Function

The kernel function $W(r,h)$ is fundamental to SPH as it defines how strongly a pair of particles interact. It depends on their distance r to each other and the smoothing length h. It is common to use different kernel functions for different purposes within a SPH simulation so that each shows a behavior suited to that purpose (compare Müller et al. [MCG03]). A suitable function needs to have three important characteristics:

1. The kernel function has a compact support c with $\|c\| = s$.
2. $\lim_{h\to 0} W(r,h)$ is equal to the Dirac delta function δ.
3. $\int_{\mathbb{R}} W(r,h)\, dr = 1$.

The smoothing length h is usually either s or $\frac{s}{2}$, depending on whether the function is non-zero on $[0,s]$ or $[-\frac{s}{2},\frac{s}{2}]$. The function is only called with positive values, thus a function with support $[-\frac{s}{2},\frac{s}{2}]$ is considered to have a smoothing length of $\frac{s}{2}$, despite the function being also

non-zero on $[-\frac{s}{2}, 0)$. Since the kernel is used for discretization of continuous quantities, there are additional characteristics that are useful for a function.

1. The function W is continuous on $[0, h]$
2. The function W is twice piece-wise continuously differentiable on $[0, h]$

These are not necessary but can be helpful to avoid discontinuities and artifacts in the simulation. For example if W is not continuous, then particles experience jolts due to the sudden jump in values, which can negatively impact the stability of the simulation. However a function might still be useful without possessing these characteristics.

A Gaussian function is cited by Monaghan [Mon92] p.545 as the archetypal kernel function. However a Gaussian does not have a compact support, thus it needs to be shifted and scaled to fulfill the first requirement. In that case h is $\frac{s}{2}$.

3.1.2 Density and Pressure

From equation 3.2 a formula for the density can be derived by substituting A for ρ. As a result the density ρ of the fluid at a particle position $\mathbf{x}_i$ can be calculated by

$$\rho(\mathbf{x}_i) = \sum_j m_j \cdot W(\|\mathbf{x}_i - \mathbf{x}_j\|_2, h). \tag{3.3}$$

For pressure, different models exist, that are employed in SPH. One common method is called Weakly Compressible SPH (WCSPH) and is described by Becker and Teschner [BT07]. They compute the pressure by using a so-called Equation of State (EOS):

$$p(\mathbf{x}_i) = k \cdot (\rho(\mathbf{x}_i) - \rho_0^f), \tag{3.4}$$

where ρ_0^f is the reference density of the fluid (e.g. 1000 $\frac{kg}{m^3}$ for water) and k is an elasticity constant. This introduces a discrepancy from the Navier-Stokes equations for strictly incompressible fluids, but one that has little influence on the simulation in most common use cases of SPH. Other methods avoid this at a cost of increased computational cost. Ihmsen et al. [IOS*14] elaborate on the differences in their report.

In the following, unless otherwise stated, we use the abbreviations $W_{ij} = W(\|\mathbf{x}_i - \mathbf{x}_j\|_2, h)$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$.

3.1.3 Forces

The two driving forces behind a basic SPH simulation are caused by pressure differences and the fluid's shear-viscosity. Both are acting pairwise between particles. Some methods also include a force modeling surface tension near the boundary between the fluid and solid objects or the surrounding air, which is not modeled, but implied. Other forces can include friction at the domain space boundaries or more complex boundary objects (see chapter 4).

For pressure and viscosity forces, Ihmsen et al. [IOS*14] use the equations

$$\mathbf{F}_i^p = -\frac{m_i}{\rho_i}\nabla p_i \tag{3.5}$$

and

$$\mathbf{F}_i^v = m_i \mu \nabla^2 \mathbf{v}_i, \tag{3.6}$$

which describe the sum of forces from all other particles on particle i. Using the SPH approach from section 3.1 they further provide the following approximations for the spatial derivatives of a quantity A:

$$\nabla A_i = \rho_i \sum_j m_j \cdot \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}\right) \cdot \nabla W_{ij}, \tag{3.7}$$

and

$$\nabla^2 A_i = 2\sum_j \frac{m_j}{\rho_j} \cdot (A_i - A_j) \cdot \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + \epsilon h^2}. \tag{3.8}$$

Using equations 3.5, 3.6, 3.7 and 3.8, we receive the formulations for the pressure force $\mathbf{F}_{i\leftarrow j}^p$ and the viscosity force $\mathbf{F}_{i\leftarrow j}^v$ for a single particle j acting on particle i:

$$\mathbf{F}_{i\leftarrow j}^p = -m_i m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \cdot \nabla W_{ij} \tag{3.9}$$

and

$$\mathbf{F}_{i\leftarrow j}^v = \frac{2 m_i m_j \mu}{\rho_j (\mathbf{x}_{ij} \cdot \mathbf{x}_{ij}) + \epsilon h^2} \cdot \mathbf{v}_{ij} \cdot \mathbf{x}_{ij} \cdot \nabla W_{ij}. \tag{3.10}$$

Here, ϵ is a constant to avoid division by zero if $\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} = 0$ and μ is a scaling constant.

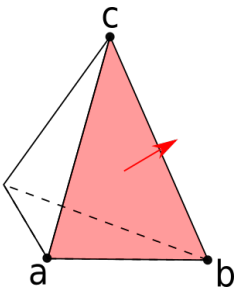

Figure 3.2: Example of a triangular mesh with four nodes and four faces. Highlighted in red is a face with its three defining nodes **a**, **b** and **c** and its normal vector pointing out of the mesh.

3.2 Mesh

Triangular meshes are a common way to describe rigid bodies in the field of computer graphics. A mesh M is defined by a set of nodes $K \subset \mathbb{R}^3$ and a set of faces $F \subset \mathbb{R}^3$. The nodes are on the surface of the body and each face f connects three nodes ($\mathbf{f}_a \in K, \mathbf{f}_b \in K, \mathbf{f}_c \in K$) in a triangle. The face is fully defined by the three nodes, but also includes the area of the triangle.

The normal vector orientation of a face is defined by the order of nodes in the tuple, meaning that the order determines in which direction the normal vector is pointing. The nodes are numbered clockwise around the normal vector when looking along its direction. This information is necessary as the normal vector also describes which side of the face is considered facing outside of the mesh. Given a face f, the normal vector $\mathbf{n}_f$ of the face can be calculated by

$$\mathbf{n}_f = \frac{(\mathbf{f}_b - \mathbf{f}_a) \times (\mathbf{f}_c - \mathbf{f}_a)}{\|(\mathbf{f}_b - \mathbf{f}_a) \times (\mathbf{f}_c - \mathbf{f}_a)\|_2}. \tag{3.11}$$

Many algorithms working on a mesh require it to be orientable and closed, meaning that there is no path from any point on the outside surface of the mesh to any point on the inside without crossing at least one face.

In the following, we distinguish between a rigid body B and a mesh M only by name. The mesh is the geometric representation of the body, while B includes additional quantities of the rigid body, such as its mass m_b or reference density ρ_0^b. For geometric algorithms the terms can be used interchangeably.

3.2.1 Mesh Algorithms

To denote if a point **x** is inside a rigid body B we introduce the sign function s by

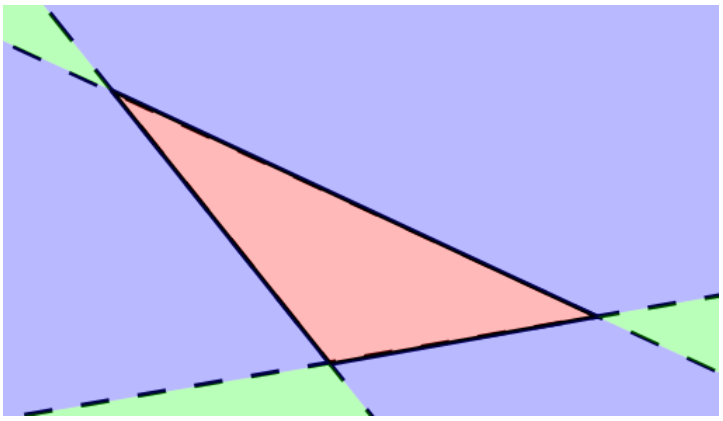

Figure 3.3: The plane, in which a triangle in 3D is embedded, divided into seven regions of three different types, which are color-coded. Red is the inner regime, blue the edge regime and green the corner regime.

$$s(\mathbf{x}) = \begin{cases} 1 & if\ \mathbf{x} \notin B, \\ -1 & if\ \mathbf{x} \in B. \end{cases} \tag{3.12}$$

For meshes this can be calculated through a ray-cast algorithm that casts a ray into a random direction and counts the number of intersections between the ray and the faces of the mesh. If the starting point of the ray is outside the mesh, this number is zero or even. Otherwise it is an odd number. The intersections can be calculated by the algorithm of Möller and Trumbore [MT05].

As a single face is not a complete mesh, the function $s(\mathbf{x})$ is not applicable. Instead we consider the plane, in which the triangle is embedded, and use its normal vector to calculate on which side $\mathbf{x}$ is. We define the planar sign function as follows:

$$s_{\text{planar}}(\mathbf{x}, f) = \begin{cases} 1 & if\ (\mathbf{x} - \mathbf{f}_a) \cdot \mathbf{n}_f \geq 0, \\ -1 & else. \end{cases} \tag{3.13}$$

3.2.2 Point-Triangle Distance and Distance Regime

The distance between a face f and a point $\mathbf{x}$ is defined as the point of f that is the closest to $\mathbf{x}$. Considering the plane in which the face is embedded, illustrated in figure 3.3, we can divide the plane into seven regions of three different regimes $r(\mathbf{x}, f)$:

1. Inner regime $r(\mathbf{x}, f) = 0$, colored in red. It is characterized by the projection point of x onto the plane falling within the triangle.

2. Edge regime $r(\mathbf{x}, f) = 1$, colored in blue. It is characterized by the points being closest to the corresponding edge of the triangle.

3. Corner regime $r(\mathbf{x}, f) = 2$, colored in green. It is characterized by the points being closest to the corresponding corner of the triangle.

Although we consider only a plane here, the regimes are defined in $\mathbb{R}^3$. The boundaries between regimes extend perpendicular to the plane towards infinity. David Eberly [Ebe99] presented an algorithm to calculate the regime and the distance of a point $\mathbf{x}$ to f. Once the region is identified, the algorithm calculates the distance by various point-point, point-line and point-plane distance algorithms. Because the closest point of f to $\mathbf{x}$ is identified in the process, we get the distance regime $r(\mathbf{x}, f)$, the distance vector $\mathbf{d}(\mathbf{x}, f)$ and its length $d(\mathbf{x}, f) = \|\mathbf{d}(x, f)\|_2$ in one pass.

Similar to s_{planar} in equation 3.13, we can define a planar distance function d_{planar}, which returns the distance from $\mathbf{x}$ to its projection onto the plane of the triangle. It is defined as

$$d_{planar}(\mathbf{x}, f) = (\mathbf{x} - \mathbf{f}_a) \cdot \mathbf{n}_f. \tag{3.14}$$

3.2.3 Plane-Sphere Intersection

A plane can intersect a sphere in no point, one point or on a circle. In the last case this circle divides the sphere into a spherical cap and a remainder. The height of the spherical cap is the distance from the surface of the sphere to the circle along an infinite line going through the center of the sphere and the center of the circle. The volume of a spherical cap with height l and spherical radius h is

$$V_c(l) = \frac{1}{3}\pi l^2 \cdot (3h - l). \tag{3.15}$$

The volume of the remainder is $V_S - V_c(l)$ where $V_S = \frac{4}{3}\pi h^3$ is the volume of the full sphere.

Chapter 4

Boundary Objects

As briefly discussed in chapter 2, several approaches exist to model boundary objects within SPH. These can be sorted into two classes by their general approach. The first is based on sampling the boundary object as particles and the second models the boundary object using a given geometric description and calculating forces from it. In section 4.3 we will present a method of the second class for boundary objects in the form of triangular meshes. For comparison, in section 4.1 we summarize the method of Akinci et al. [AIA*12] for the first class and the method of Koschier and Bender [KB17] in section 4.2 for the second class.

In all three approaches the density of a fluid particle near the boundary object B is corrected by a quantity ρ^b such that $\rho_i = \rho_i^f + \rho_i^b$, where ρ_i^f is the fluid density as calculated by equation 3.3. This correction contributes to an altered behavior of the fluid, but is insufficient to ensure the fluid does not penetrate the boundary object. Thus, a pressure-based force is introduced, that pushes the fluid away from the object. Furthermore, the physical phenomenon of viscosity-based friction near solid surfaces is modeled by a corresponding friction force.

4.1 Particle Sampling Method

Akinci et al. [AIA*12] present a method to model boundary objects and incorporate them as dynamic rigid bodies in SPH simulations as well as coupling these models to physics simulations for dynamic rigid bodies. This method belongs to the first of the aforementioned classes of methods as it represents the boundary object as a set of particles, that interact with the fluid particles and transfer forces to the rigid body for the coupled physics simulation.

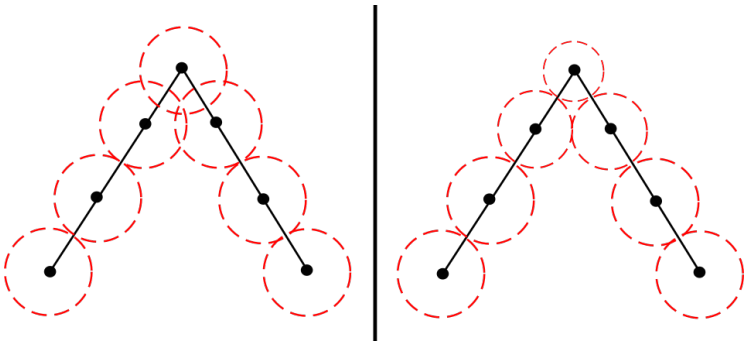

Figure 4.1: The surface of a boundary object being sampled by particles. The influence a particle has on the density correction ρ^b is visualized as the size of a red circle around each particle. Left: A uniform mass leads to a higher density where particles are close together. Right: Including the local number density corrects the calculation of ρ^b.

4.1.1 Sampling

The method of Akinci et al. [AIA*12] samples a number of points M on the surface of the boundary object, which can be given as a triangular mesh or in another form of geometric description. Ideally the surface is sampled evenly such that every point has an average distance to its close neighbors of roughly $\sqrt{\frac{A_b}{M}}$, where A_b is the surface area of the boundary object. This helps to ensure that there are no regions of the surface that would allow penetration of fluid particles due to under-sampling of that region.

Depending on the chosen distribution method, by which the coordinates are generated, the particles may be sampled more densely near edges of the boundary object. This needs to be considered for the calculation of ρ^b.

If the boundary object is given as a polygonal mesh, each face could be considered individually and regular grid be placed over the surface of the face. The grid cell size is chosen as $\sqrt{\frac{M \cdot A_f}{A_b}}$, where A_f is the area of face f. For large M this method approximately samples the whole of B evenly with M particles, however it also produces over- or under-sampling if the shapes of the faces do not line up well with the grid.

4.1.2 Density Correction

Akinci et al. [AIA*12] propose a method to calculate ρ^b, that also considers non-uniform sampling. To achieve this they introduce the local number density

$$\Psi_j^b = \frac{\rho_0^f}{\sum_k W_{jk}}, \tag{4.1}$$

where k denotes all particles of the boundary object in the neighborhood of a particle j. Hence, Ψ_j^b is a measure for the mass of j and is inversely proportional to the sampling density

in the region around $\mathbf{x}_j$. The density of a fluid particle i is then corrected using the local number density Ψ^b via

$$\rho_i^b = \sum_k \Psi_k^b \cdot W_{ik}. \tag{4.2}$$

Figure 4.1 illustrates the effect Ψ^b has on ρ^b. Assuming a uniform mass for each particle leads to an overestimation of ρ^b where the boundary object is sampled more densely. Taking Ψ^b into consideration corrects the calculation of ρ^b.

Akinci et al. [AIA*12] further note that it is sufficient to sample the boundary object as a single layer of boundary particles, because Ψ^b corrects for the missing layers by increasing the influence of the first one. As only rigid bodies are considered, the local number density Ψ^b does not change during the simulation. The values of Ψ^b can be precomputed in the first simulation step and saved in the same SDS as the boundary particles' positions.

4.1.3 Pressure and Friction Forces

With Ψ_j^b as a measure for the mass of a boundary particle, the pressure force $\mathbf{F}_i^{p,b}$ from B on a fluid particle i can be derived from equation 3.9. With the assumption that the density on the surface of B is the same as ρ_i, this results in

$$\mathbf{F}_i^{p,b} = -m_i \sum_k \Psi_k^b(\rho_0^f) \cdot \left(\frac{p_i}{\rho_i^2}\right) \cdot \nabla W_{ik}, \tag{4.3}$$

where k denotes the boundary particles.

Akinci et al. [AIA*12] derive a formulation for a friction force between the boundary object and the fluid from a model for laminar artificial viscosity, which was introduced by Monaghan [Mon05]. The friction force acting on a fluid particle i is modeled as

$$\mathbf{F}_i^f = -m_i \sum_k \Psi_k^b(\rho_0^f) \cdot \Pi_{ik} \cdot \nabla W_{ik}, \tag{4.4}$$

where Π_{ik} is a quantity describing the laminar viscosity, which is maximal if $\mathbf{v}_{ik}$ is tangential to the surface. Π_{ik} is given as

$$\Pi_{ik} = -\frac{\sigma h c_s}{2\rho_i} \cdot \left(\frac{\min(\mathbf{v}_{ik} \cdot \mathbf{x}_{ik}, 0)}{\|\mathbf{x}_{ik}\|_2^2 + \epsilon h^2}\right), \tag{4.5}$$

where σ is boundary viscosity constant and c_s a quantity describing the speed of numerical propagation, which is similar to the speed of sound in the fluid, but also influenced by h and the time step Δt.

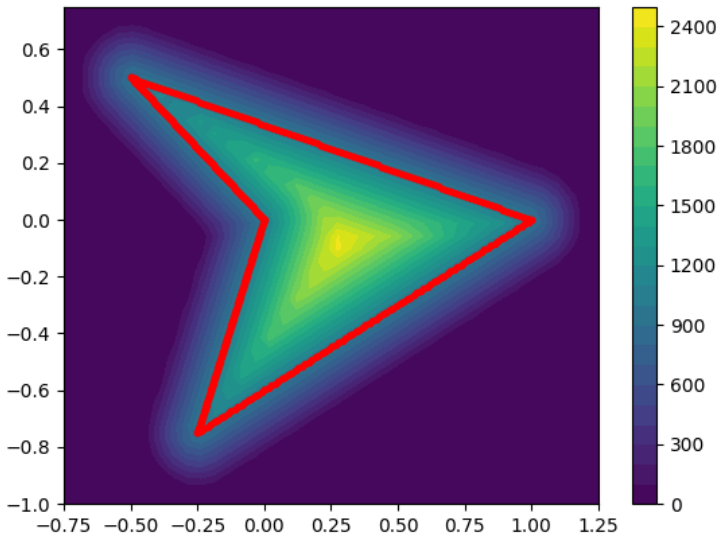

Figure 4.2: The function $\gamma_h(\Phi(\mathbf{x}))$ with $h = 0.2\ m$ and $\rho_0^f = 1000\ \frac{kg \cdot m}{s^2}$ for a 2D mesh highlighted by red lines. The color scale shows the $\gamma_h(\Phi(\mathbf{x}))$ values as filled contour areas with 25 buckets.

4.2 Density Field Method

Koschier and Bender [KB17] propose a method to calculate the density correction and fluid-boundary forces through a geometric description of the boundary. In comparison to the method of Akinci et al. [AIA*12] this method does not sample the boundary as particles. The density correction and forces are instead directly derived from geometric properties of the boundary object. As a result the boundary's smoothness is preserved, whereas the method of Akinci et al. [AIA*12] is only as smooth as the sampling density allows. As these properties don't change for rigid bodies, most of the terms contributing to the density and force equations can be precomputed in the first time step and be used during the simulation. This facilitates a much faster fluid-boundary handling than in the particle sampling method.

4.2.1 Global Calculation of Density Field

The signed distance $\Phi(\mathbf{x})$ of a point $\mathbf{x}$ to a mesh is the sign of $\mathbf{x}$ multiplied with the minimum distance to all faces of the mesh. Over all points $\mathbf{x}$ in the simulation domain, this results in a field that contains the information on distance and sign relative to the mesh. $\Phi(\mathbf{x})$ is defined as

$$\Phi(\mathbf{x}) = s(\mathbf{x}) \cdot min_f d(\mathbf{x}, f). \tag{4.6}$$

If a particle is moving towards B, the density correction ρ^b should be zero if the distance is greater than h, but increase the smaller the distance is. Once the particle is inside B, the density correction should further increase with distance to the surface. To facilitate this, the following weighting function $\gamma_h(r)$ is used:

$$\gamma_h(r) = \begin{cases} \rho_0^f \cdot (1 - \frac{r}{h}) & if\ r \leq h, \\ 0 & else. \end{cases} \tag{4.7}$$

Again, considered over all points $\mathbf{x}$ in the simulation domain, $\gamma_h(\Phi(\mathbf{x}))$ has a higher value the higher the influence of B on the fluid at $\mathbf{x}$ is. Figure 4.2 shows a 2D example of a calculated γ_h-field for a mesh drawn by the four red lines. This has the expected behavior of increasing values when tracing a line from outside the mesh to the inner area. At a distance of $h = 0.2$ from the outside of the faces the values are zero, while the values directly on the surface are equal to the reference density.

From this field Koschier and Bender [KB17] derive the density correction for a fluid particle i by computing the overlapping volume of the boundary object within the particle's sphere of influence S_i^h. For example, let us consider a particle close enough to the boundary object B that a face intersects S_i^h. The intersecting volume displaces any fluid that would otherwise fill this region and contribute to the particle's density calculation. By multiplying a given function with a kernel and integrating over this intersecting volume, we get a value that corresponds directly to the influence of the boundary object. With the weighted density function γ_h we obtain the density correction function ρ^b by

$$\rho^b(\mathbf{x}_i) = \int_{S_i^h \cap B} \gamma_h(\Phi(\mathbf{x}^*)) \cdot W(\|\mathbf{x}_i - \mathbf{x}^*\|_2, h)\ d\mathbf{x}^*. \tag{4.8}$$

Considering that the boundary object is modeled as a rigid body, the values of this function do not change over time within the reference frame of the object. Therefore, it is desirable to compute values of ρ^b for the boundary object only once as a precompute step to the simulation. The method of Koschier and Bender [KB17] achieves this by placing a grid of a given resolution over the boundary object within its reference frame.

The particle position $\mathbf{x}_i$ corresponds to a point $\hat{\mathbf{x}}_i = \mathbf{R}_b \cdot (\mathbf{x}_i - \mathbf{x}_b)$ relative to the center of mass $\mathbf{x}_b$ of B and its rotation $\mathbf{R}_b$. With this grid, $\rho^b(\hat{\mathbf{x}}_i)$ can be calculated as an interpolation of the surrounding grid points of $\hat{\mathbf{x}}_i$. Consequently, the computational cost is reduced to calculating $\hat{\mathbf{x}}_i$ as well as interpolating from a fixed number of grid points. This discretization of ρ^b on a grid has the further advantage that the gradient of ρ^b can be calculated in a similar way. The gradient is required to get the pressure force acting between the fluid particles and the boundary object.

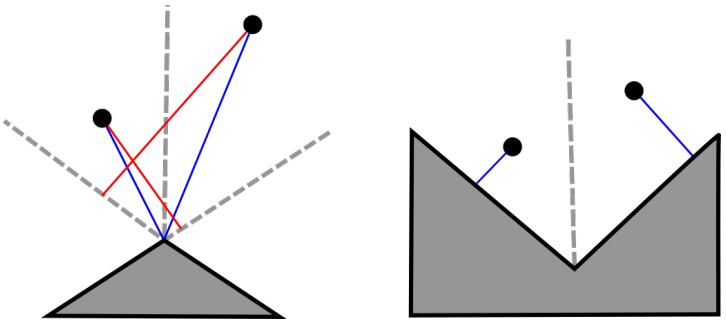

Figure 4.3: Left: A convex corner of a 2D mesh with two points and their distance vectors to both the corner (in blue) and the face with the furthest planar distance (in red). Right: A concave corner of a 2D mesh with two points and their distance vectors to the closest face (in blue).

4.3 Mesh-based Method

The density correction ρ^b using the method of Koschier and Bender [KB17] can be calculated to an arbitrary degree of smoothness by increasing the grid's resolution. For boundary objects modeled as a triangular mesh, the density correction will be only as smooth as the mesh itself. By using the mesh data to decompose ρ^b into a set of local functions each tailored to one face of the mesh, we can avoid the need to discretizing the continuous global field ρ^b. A face f, that has a closest distance to a particle position $\mathbf{x}_i$ of greater than h, does not need to be considered. In the following we reduce querying faces to a set of local faces

$$F_h(\mathbf{x}_i) = \{f \in F \mid d(\mathbf{x}_i, f) \leq h\}. \tag{4.9}$$

4.3.1 Density Correction

Within a region $D_f = \{\mathbf{x} \in \mathbb{R}^3 \mid d(\mathbf{x}, f) \leq h\}$ around a face f, the weighted distance function $\gamma_h(\Phi(\mathbf{x}))$ has a shape that is linear in $\mathbf{d}(\mathbf{x}, f)$. This is demonstrated in the example used in figure 4.2, where the contour lines are parallel to the faces in the middle regime and circular around corners in the corner regime. However these regions overlap near corners in the 2D case and near corner and edges in the 3D case. This is an issue for convex corners on the outside of B or concave corners on the inside. While the distance to all nearby faces within the overlap region is the same, the planar sign function is not and choosing arbitrarily between associating with either face would yield the wrong density correction.

The left example in figure 4.3 illustrates this issue in the 2D case with two points in the corner regime of a convex corner. The distance vector to the corner is drawn in blue, while the distance vector towards the face with the furthest planar distance is drawn in red. With this distinction the overlap region is split by the angle bisector into two regions each associated with the face on the same side as the rest of D_f. In figure 4.3 this results in the left point

being associated with the left face and the right point with the right. In the following, we will denote the face closest to a particle position $\mathbf{x}_i$ with $\hat{f}$, taking the distinction described above into consideration.

For concave corners this issue does not occur because the distance regime is the edge regime for both faces. This is demonstrated in the right example in figure 4.3. Here, $\hat{f}$ is always correctly identified and switches when the particle crosses the angle bisector.

As described in section 4.2.1, the density correction is derived from the intersection volume $V_i^b = S_i^h \cap B$. Let us consider a case where $\mathbf{x}_i$ is in the inner regime of $\hat{f}$ and has a positive value of $s(\mathbf{x}_i, \hat{f})$. In this case the intersection volume V_i^b is that of a spherical cap with a height equal to $h - d(\mathbf{x}_i, \hat{f})$. If $s(\mathbf{x}_i, \hat{f})$ is negative the volume of the spherical cap is subtracted from the volume of S_i^h as the distance always has a positive sign but more than half of S_i^h is already inside the boundary object.

If $\mathbf{x}_i$ is in the edge or corner regime of $\hat{f}$ the intersection volume is more complicated to calculate as more faces and the angles between them need to be considered. To avoid this, we use an approximation that appears sufficiently accurate in the numerical experiments (compare chapter 6). The intersection volume is always calculated as if $\mathbf{x}_i$ was in the inner regime, but the true distance $d(\mathbf{x}_i, \hat{f})$ is used instead of $d_{\text{planar}}(\mathbf{x}_i, \hat{f})$.

This approximation performs well because it cancels the overestimation of V_i^b, due to calculating the spherical cap on a corner-less plane, with an underestimation due to using $d(\mathbf{x}_i, \hat{f})$ instead of $d_{\text{planar}}(\mathbf{x}, \hat{f})$. It remains an approximation, albeit a useful one, because these don't cancel out exactly.

As the intersection volume V_i^b is a direct measure for how much fluid the boundary object displaces within S_i^h we can calculate the density correction by

$$\rho_i^b = \rho_0^f \cdot \frac{V_i^b}{\frac{4}{3}\pi h^3}. \tag{4.10}$$

Algorithm 1 summarizes the steps to calculate ρ_i^b for a particle i.

4.3.2 Pressure Force

The calculation of ρ_i^b with the method above introduces a useful quantity in the form of V_i^b. If interpreted as a displacement of fluid within S_i^h it yields ρ_i^b but V_i^b can also be used in the calculation of the fluid-boundary pressure force $\mathbf{F}_i^{p,b}$. The point $\mathbf{c}_i = \mathbf{x}_i + \mathbf{d}(\mathbf{x}_i, \hat{f})$ on the surface of B serves as the point on which the pressure force acts on the boundary object and $\frac{\mathbf{d}(\mathbf{x}_i, \hat{f})}{d(\mathbf{x}_i, \hat{f})}$ as the direction of $\mathbf{F}^{p,b}$. Additionally we multiply with $-s_{\text{planar}}(\mathbf{x}_i, \hat{f})$, because $\mathbf{d}(\mathbf{x}_i, \hat{f})$ points towards $\hat{f}$, but $\mathbf{F}_i^{p,b}$ should point outwards from the boundary object regardless of whether $\mathbf{x}_i$ is on the outside or inside of it.

Pseudocode 1 Algorithm to approximate the density correction with only local information.

if $F_h(\mathbf{x}_i) = \emptyset$ **then**
    $\rho_i^b = 0$
if $\{f \in F_h(\mathbf{x}_i) \mid r(\mathbf{x}_i, f) = 0\} \neq \emptyset$ **then**
    $\hat{f} = argmin_f\ d(\mathbf{x}_i, f)$
else
    $\hat{f} = argmax_f\ d_{\text{planar}}(\mathbf{x}_i, f)$
$l = h - d(\mathbf{x}_i, \hat{f})$
if $s_{\text{planar}}(\mathbf{x}_i, \hat{f}) \geq 0$ **then**
    $V_i^b = V_c(l)$
else
    $V_i^b = \frac{4}{3}\pi h^3 - V_c(l)$
$\rho_i^b = \rho_0^f \cdot \frac{V_i^b}{\frac{4}{3}\pi h^3}$

The quantities $\hat{f}$, $\mathbf{d}(\mathbf{x}_i, \hat{f})$ and $s_{\text{planar}}(\mathbf{x}_i, \hat{f})$ are independent of the particle density. However, as discussed in section 3.1.2, the pressure is a function of the density difference to the reference density. While it is possible to place a virtual particle with position $\mathbf{c}_i$ and then calculate the density and pressure as with a fluid particle, this does not work with the intent to avoid sampling the boundary object with virtual particles and we instead approximate the density at position $\mathbf{c}_i$ as ρ_i^f. This approximation is less accurate if $\mathbf{x}_i$ is far away from $\hat{f}$, but because V_i^b also decreases with distance[1], this inaccuracy has little effect.

Similar to the equation of state for the fluid pressure (see equation 3.4), we introduce a parameter k_p that scales the pressure. It is of note that this parameter might be similar, but not necessarily equal to the k used in the fluid pressure calculation. The pressure force is then calculated as

$$\mathbf{F}_i^{p,b} = -s_{\text{planar}}(\mathbf{x}_i, \hat{f}) \cdot V_i^b \cdot k_p \cdot \max(\rho_i, \rho_0^f) \cdot \frac{\mathbf{d}(\mathbf{x}_i, \hat{f})}{d(\mathbf{x}_i, \hat{f})}, \tag{4.11}$$

for the pressure force acting between a particle i and the boundary object B. The density is clamped to ρ_0^f to avoid that particles with low density experience insufficient force to be pushed out of the boundary object.

4.3.3 Friction Force

An equation for a friction force acting between the boundary object and a fluid particle can be derived in the same way as the pressure force. Some differences need to be taken into account though. First, the friction force applies only if the particle is moving towards $\hat{f}$. Second, the force is acting tangential to $\hat{f}$ and opposite of $\mathbf{v}_{ib} = \mathbf{v}_i - \mathbf{v}_b$, the relative velocity between

[1] This is only true outside of the boundary object. An overestimation of $\mathbf{F}_i^{p,b}$ inside of B is not much of a problem as the particle should be pushed out of B anyway.

the particle i and B. For a particle that is moving towards the plane of $\hat{f}$, the value of the scalar product of $\mathbf{n}_{\hat{f}}$ and $\mathbf{v}_{ib}$ is smaller than zero. This also holds true for the corner regime. In figure 4.3, we can imagine the right particle moving towards the plane of the left face while still increasing its true distance $d(\mathbf{x}_i, \hat{f})$. However at this point the particle has crossed the angle bisector, at which point $\hat{f}$ switches to the right face and the particle is considered moving away from $\hat{f}$.

Koschier and Bender [KB17] use the formulation

$$\mathbf{t}_{\hat{f}} = \mathbf{v}_{ib} - \mathbf{n}_{\hat{f}} \cdot (\mathbf{v}_{ib} \cdot \mathbf{n}_{\hat{f}}) \tag{4.12}$$

for the tangential vector in their calculation of the friction force. This equation is also well suited to our adapted method and results in

$$\mathbf{F}_i^f = \begin{cases} \mathbf{t}_{\hat{f}} \cdot V_i^b \cdot \mu^b \cdot \max(\rho_i, \rho_0^f) & \text{if } \mathbf{n}_{\hat{f}} \cdot \mathbf{v}_i \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

for the friction force between B and the fluid particle i.

Chapter 5

Implementation Details

The smoothing length h is a crucial quantity for the simulation scale. Hence it has to be chosen carefully. If we want to simulate a volume V_D with N particles, we expect that each particle represents a volume of $\frac{V_D}{N}$, which is equal to the volume of a cube having side length

$$d_p = \sqrt[3]{\frac{V_D}{N}}. \tag{5.1}$$

We choose the particles as spheres with radius of $\frac{d_p}{2}$. However, it is an open question how much this assumption impacts the simulation results, given that d_p is derived from cubic volumes. Furthermore, we select the smoothing length as $2 \cdot d_p$. Consequently, the mass of a particle can be calculated as

$$m_i = \rho_0^f \cdot d_p^3. \tag{5.2}$$

5.1 Simulation Framework

The simulation framework used in this work was customized for the methods in chapter 4 and inspired by a framework used previously by Reinhardt et al. [RHEW17]. The method, by which the particle positions are initialized in this framework, is based on a sphere packing algorithm (see Ott [Ott18]). A fluid initialized in this way is not immediately in a rest state. This is relevant as far as the fluid's energy state is concerned. The framework uses a Leap-Frog method with a fixed time step Δt for solving the equations of motion.

5.2 Domain Boundaries

Because the SDS imposes restrictions on the position of particles, the fluid needs to be kept within the domain space $D = [D_x^l, D_x^u] \times [D_y^l, D_y^u] \times [D_z^l, D_z^u]$. It is possible to use boundary objects for this as discussed in chapter 4. For performance reasons it is better to use a simpler reflection scheme for the cube-shaped domain space boundaries. The framework performs a check in the form of

$$\mathbf{x}_i^* = \mathbf{x}_i^t + \Delta t \cdot \mathbf{v}_i^{t+\frac{1}{2}} \tag{5.3}$$

and

$$x_{i,k}^{t+1} = \begin{cases} 2D_k^l - x_{i,k}^* & \text{if } x_{i,k}^* \leq D_k^l \\ x_{i,k}^* & \text{if } D_k^l \leq x_{i,k}^* \leq D_k^u \\ 2D_k^u - x_{i,k}^* & \text{if } D_k^u \leq x_{i,k}^* \end{cases} \tag{5.4}$$

for each dimension $k \in \{x, y, z\}$ during the position integration. If a position was reflected in this way, the velocity is also reflected and dampened with a factor $\delta \in [0, 1]$ such that

$$\mathbf{v}_i^{t+\frac{1}{2}} = \delta(\mathbf{v}_i^{t+\frac{1}{2}} - 2\mathbf{e}_k \cdot (\mathbf{e}_k \cdot \mathbf{v}_i^{t+\frac{1}{2}})) \tag{5.5}$$

where $\mathbf{e}_k$ is the unit vector for dimension k.

5.3 Parameters and Kernel Functions

Unless otherwise stated, the framework uses the parameter values listed in table 5.1. Following common nomenclature, we use SI units wherever applicable.

Table 5.2 lists three different kernel functions used in the simulation framework. The Poly6 kernel function, introduced by Müller et al. [MCG03], is used for the calculation of ρ^f and Ψ^b. The Spiky kernel function was introduced by Desbrun and Gascuel [DG96] and is used for the fluid-fluid pressure force $\mathbf{F}^p$ and the forces between fluid and boundary objects. For the fluid-fluid viscosity force $\mathbf{F}^v$ the Wendland kernel is used, which was introduced by Schwab and Wendland [SW92].

Name	Value	Name	Value
ρ_0^f	$1000\ \frac{kg}{m^3}$	ϵ	0.01
k	$1000\ \frac{m^2}{s^2}$	k^b	$1000\ \frac{1}{s^2}$
μ	$0.01\ \frac{1}{m^4 s}$	μ^b	$10\ \frac{1}{s^2}$
σ	$0.1\ \frac{kg}{m^4}$	δ	0.9
c_s	$340\ \frac{m}{s}$	g	$9.81\ \frac{m}{s^2}$
Δt	$0.001\ s$		

Figure 5.1: Parameter values of the simulation framework used in this work.

Name	W_{ij}	∇W_{ij}
Poly6	$\frac{315(h^2-\Vert\boldsymbol{x}_{ij}\Vert_2^2)^3}{64\pi h^9}$	$\boldsymbol{x}_{ij} \cdot \frac{-945(h^2-\Vert\boldsymbol{x}_{ij}\Vert_2^2)^3}{32\pi h^9}$
Spiky	$\frac{15(h-\Vert\boldsymbol{x}_{ij}\Vert_2)^3}{\pi h^6}$	$\boldsymbol{x}_{ij} \cdot \frac{-45(h-\Vert\boldsymbol{x}_{ij}\Vert_2)^3}{\pi h^6 \Vert\boldsymbol{x}_{ij}\Vert_2}$
Wendland	$\frac{168}{16\pi h^3} \cdot \left(1-\frac{\Vert\boldsymbol{x}_{ij}\Vert_2}{h}\right)^4 \cdot \left(\frac{4\Vert\boldsymbol{x}_{ij}\Vert_2}{h}+1\right)$	$\boldsymbol{x}_{ij} \cdot \frac{-210}{\pi h^5 \Vert\boldsymbol{x}_{ij}\Vert_2} \cdot \left(1-\frac{\Vert\boldsymbol{x}_{ij}\Vert_2}{h}\right)^3$

Figure 5.2: Various kernel functions W_{ij} and their derivatives ∇W_{ij}.

Chapter 6

Results

There are different aspects to the methods described in chapter 4 that are of interest for comparing their suitability to different simulation goals. In this section we discuss three experiments that highlight some of these aspects in regard to how they apply to the particle sampling method of Akinci et al. [AIA*12] and our method as presented in section 4.3. Unfortunately it was not possible to include a comparison with the method of Koschier and Bender [KB17] in this work.

6.1 Sheet of Fluid on Inclined Plane

The smoothness of the force field, that is induced by the boundary object on the fluid, is determined by both the inherited smoothness of the object's geometric description as well as the boundary handling method and how well it samples the object. For example, a boundary object in the shape of the cube has a geometric description that is piece-wise smooth on each side. In addition to this, the boundary handling method can further alter the smoothness.

The method of Akinci et al. [AIA*12] discretizes the surface as distinct particles, meaning that a plane becomes a surface composed of spheres. At a higher sampling resolution this surface approximates a plane reasonably well, but causes additional costs. Our method can sample any polygonal shape on a plane perfectly, but also has an error of approximation for curved planes or non-polygonal shapes.

Figure 6.1 shows a comparison of both methods emphasizing their impact on the surface smoothness. 225 particles are initialized as a sheet with a distance of 0.025 m between them. In the beginning they are resting on a 1 $m \times 1$ m plane with a slope of 25%. After 0.8 s have passed, the sheet has slid down the plane. The distance from the initial position depends on each method's friction force parameter (σ and μ^b) and cannot be compared in a simple manner. Comparing the friction forces for each method (see equations 4.4 and 4.13), no direct

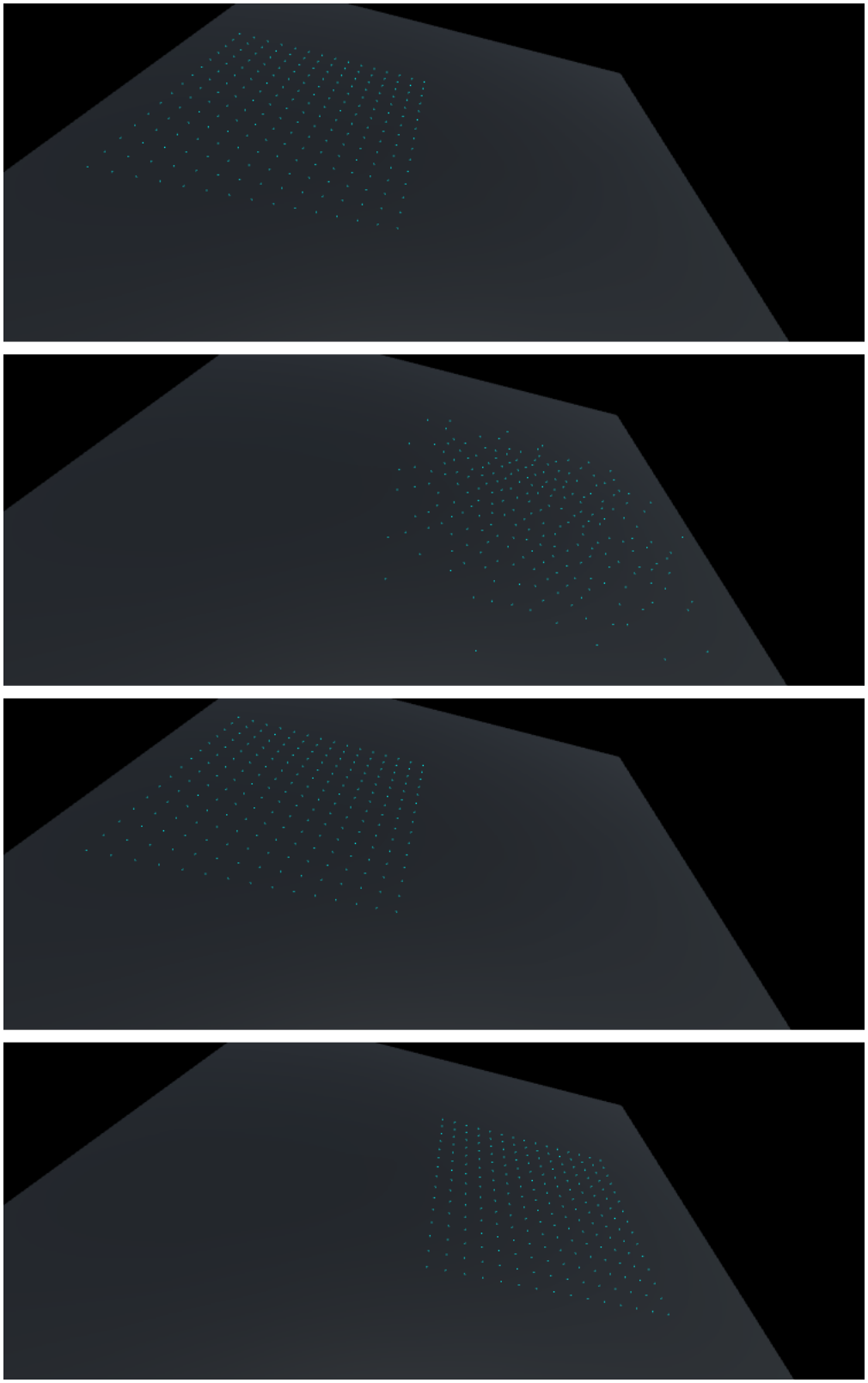

Figure 6.1: A sheet of 225 fluid particles with a distance of 0.025 m between them on a 1 $m \times$ 1 m plane with a slope of 25%. First row: The initial setup of the simulation with the method of Akinci et al. [AIA*12] with 9409 boundary particles for the plane. Second row: The state at $t = 0.8$ s. Third row: The initial setup of the simulation with our method. Fourth row: The state at $t = 0.8$ s.

correspondence between both parameters is apparent. Furthermore, despite the parameter values differing by two orders of magnitude, the fluid shows roughly the same behavior.

A clear difference between the methods is visible in the regularity the particles display at time $t = 0.8\ s$. With the method of Akinci et al. [AIA*12] the particles are disturbed from a straight path down the plane by the rough surface of the boundary particles, despite the comparatively high sampling density[1]. Our method preserves the regularity of the particles throughout the slide down the plane.

6.2 Drop of Fluid Falling on a Pyramid

For the purpose of using boundary objects as barriers to shape the flow, it is important that the particles do not penetrate the barrier. This cannot be guaranteed for every possible initial configuration, but should be the case for a relatively large part of the scenarios used in the field of CG. Therefore it is worthwhile to identify cases for which the method works well to avoid boundary penetration and where it does not.

Figure 6.2 shows an example where a drop of fluid is falling on an obstacle in the shape of a pyramid with a triangular base. Because the drop is centered above the pyramid, the fluid should split into three parts and splash against the ground and distribute over the area not covered by the obstacle. We observed two phenomena that did not fulfill the expectations.

The first occurs while the drop is being split by the pyramid and has not reached the ground yet (see second row in figure 6.2). As expected, the drop is being split by the pyramid into three parts, but some particles have penetrated the boundary object before they are pushed out some time later (compare third row in figure 6.2). The other deviation is noticeable after some time has passed and the fluid is settling into a state of equilibrium (see fourth row in figure 6.2). Some fluid particles remain in the inside of the pyramid and are not pushed out of it. These particles move within the interior of the object, but do not get close enough to the boundary, that they would experience a force pushing them out.

6.3 Breaking Dam Scenario

Stability is always a concern in fluid simulations. For example, problems can arise when the velocity of particles is too high for the used time step. A breaking dam scenario is often used to show susceptibility for or resistance against these kind of problems, due to the high velocity and large pressure forces occurring in this scenario.

[1] Placing the boundary particles $0.8h$ apart leads to a sampling density of 3.762 particles per h^2 surface area. In this example the plane is sampled with a density of 5.929 $\frac{1}{h^2}$.

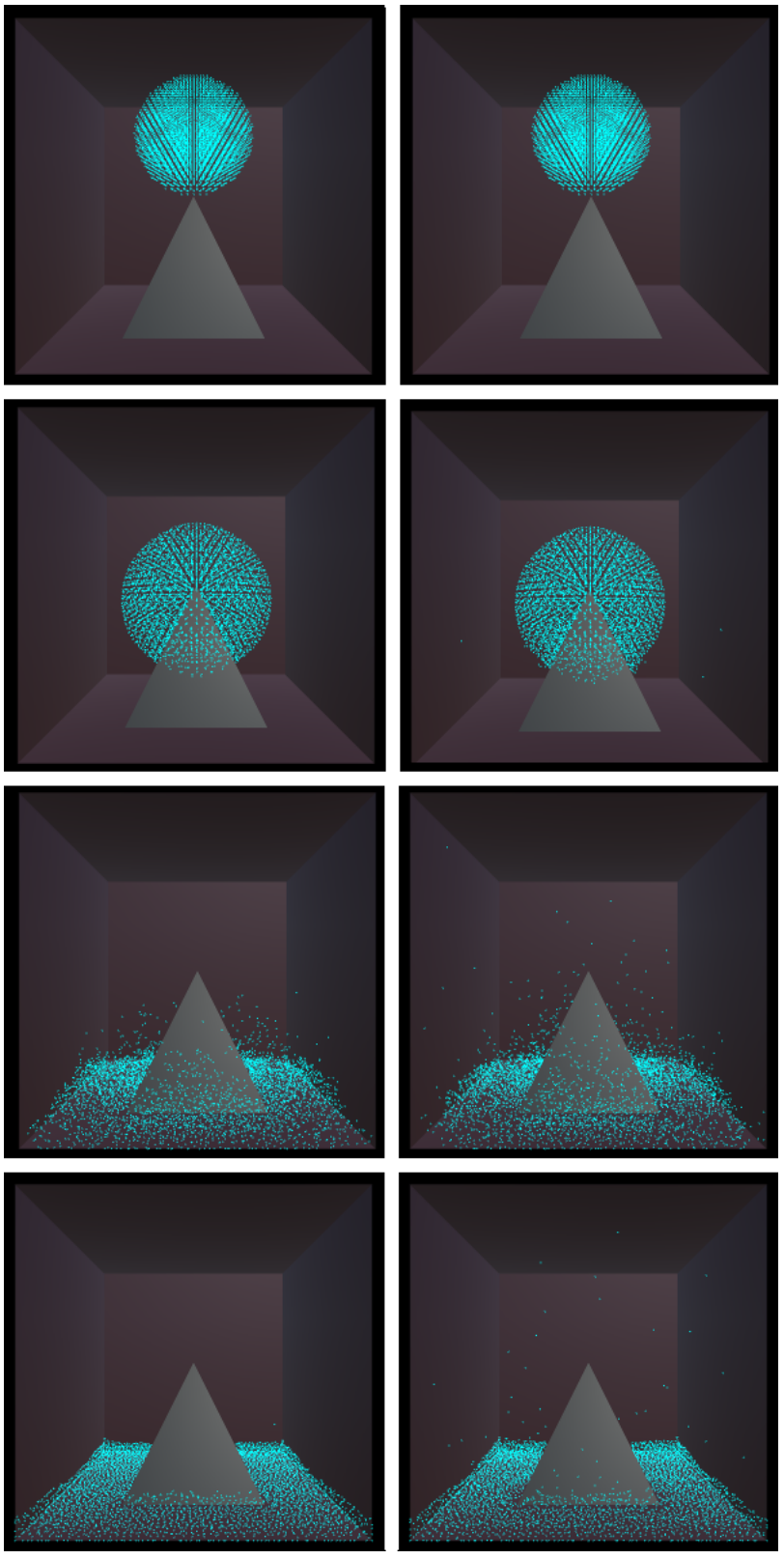

Figure 6.2: A sphere of 5000 particles is dropped on a pyramid with triangular base. Left column: The initial state of the simulation with the method of Akinci et al. [AIA*12]. The second row within the column shows the state at $t = 0.25$ s. The sphere is being split into three parts. The third row shows the state at $t = 0.6$ s, when the fluid has splashed against the ground and spreads out. The fourth row shows the state at $t = 1.2$ s. The fluid has mostly come to a rest state. Right column: The same sequence of states with our method. Not visible are some particles, that have penetrated the boundary object.

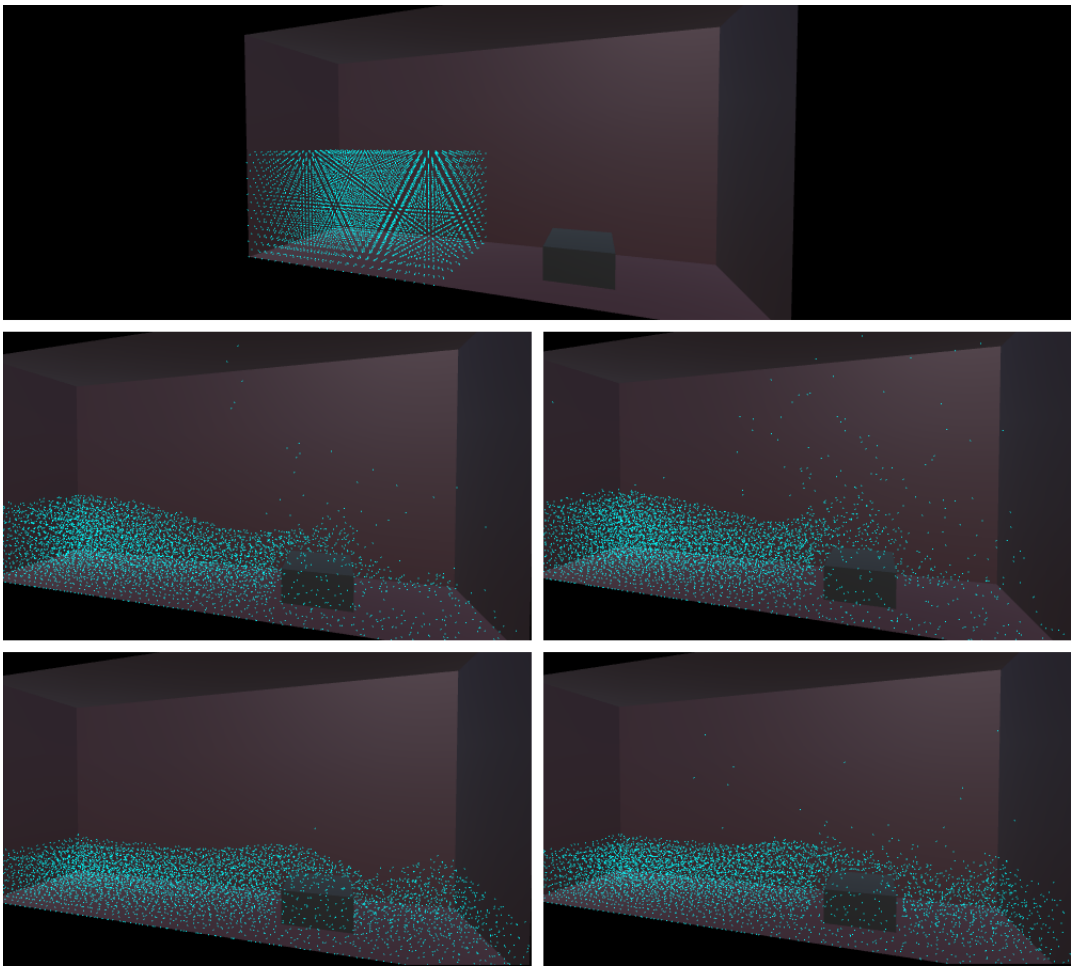

Figure 6.3: A breaking dam scenario with a box-shaped obstacle in the channel. 5040 particles are initialized in a 0.5 $m \times 1\ m \times 0.5\ m$ box and flow into the 2 $m \times 1\ m \times 0.5\ m$ channel. A 0.25 $m \times 0.125\ m \times 0.25\ m$ box is placed as obstacle into the channel. First row: The initial state of the simulation with either method. Left column, first row: The state at $t = 0.6\ s$ with the method of Akinci et al. [AIA*12]. The flow is held back by the the obstacle and the fluid overflows around and over the obstacle. Left column, second row: The state at $t = 1.2\ s$. The fluid has mostly passed the obstacle. Right column: The same sequence of states with our method.

Figure 6.3 shows a breaking dam scenario, where 5040 particles are initialized in a 0.5 $m \times 1\ m \times 0.5\ m$ box and flow into a 2 $m \times 1\ m \times 0.5\ m$ channel. A 0.25 $m \times 0.125\ m \times 0.25\ m$ box is placed as an obstacle into the channel, intended to force the fluid to flow around and over the obstacle. After 0.6 s a flood wave has formed and reached the obstacle. Part of the fluid is held back in front of the obstacle, allowing the following fluid to flow up and over the obstacle. At time $t = 1.2\ s$ the flood wave has mostly passed the obstacle and backs up at the end of the channel.

Both methods show roughly the same behavior, although it is noticeable that with our method the bow wave, produced when the fluid hits the obstacle, happens at a larger distance in front of the obstacle. Similarly, the particles flowing around the side of the obstacle keep a slightly larger distance to the object's surface than with the method of Akinci et al. [AIA*12].

Chapter 7

Discussion & Future Work

The experiments in chapter 6 yield interesting results. The thin sheet slide experiment in section 6.1 showed a fluid behavior well within the expectations for both the particle sampling method as well as our method. This use case presents a strength of our method, as it preserves the mesh's smooth surface without further discretization. However, the experiment in section 6.2, where a drop of fluid is falling onto a pyramid, shows deviations from expected behavior. While the overall result is similar to the control experiment with the method of Akinci at el. [AIA*12], some particles penetrate the boundary object with our method. We have not investigates the reason for this fact. It is possible that the pressure force parameter k^b could yield correct fluid behavior, if the value was chosen higher, but a stronger pressure force may also lead to instabilities. In the breaking dam experiment both methods performed as expected. The experiment is intended to test the sensitivity of the method to high velocities and pressures. In this regard no significant issues were noticeable in the simulation results. A surprising, but not necessarily problematic, phenomenon was observed concerning the distance of the fluid particles to the obstacle. This distance is comparatively high. In cases, where this phenomenon might propose a challenge, it should be possible to employ a suitable modification to equation 4.11 to shift the fluid surface further towards the boundary object.

For polygonal meshes in particular there are a number of existing frameworks to simulate the interaction of dynamic rigid bodies with each other and the domain boundaries. The point on the surface of the boundary object, that acts as the point of interaction for the pressure- and friction-based forces (see equations 4.11 and 4.13), is already being calculated in the process of our method. With this, it is possible to apply Newton's third law to get a formulation for the reaction force acting from the fluid on the object. The single forces can accumulated to form a force acting on the center of mass of the rigid body, thus bridging the fluid simulation to a coupled simulation of dynamic rigid bodies.

Overall, the method presented in this work produces reasonable results. However, improvements are still possible in regard to the boundary penetration, as discussed above. The method has the potential to provide a fast alternative to existing methods for rigid bodies that can be approximated well using triangular meshes.

The performance advantage, that can be achieved by using our method over the methods of Akinci et al. [AIA*12] and Koschier and Bender [KB17] respectively, depends on how the boundary object is constructed. If the boundary representation is given only in the form a polygonal mesh, it might be optimal to employ our method. On the other hand, if the boundary object has a highly curved shape, it is necessary to create a mesh with a large number of faces, whereas the method of Koschier and Bender [KB17] can handle this case with higher accuracy and possibly more efficiently.

Bibliography

[ACAT13] AKINCI N., CORNELIS J., AKINCI G., TESCHNER M.: Coupling elastic solids with smoothed particle hydrodynamics fluids. *Computer Animation and Virtual Worlds 24*, 3-4 (2013), 195–203.

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible sph. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 62.

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 209–217.

[CF88] CONSTANTIN P., FOIAS C.: *Navier-stokes equations*. University of Chicago Press, 1988.

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation'96*. Springer, 1996, pp. 61–76.

[Ebe99] EBERLY D.: Distance between point and triangle in 3d. *Magic Software, http://www. magic-software. com/Documentation/pt3tri3. pdf* (1999).

[GM77] GINGOLD R. A., MONAGHAN J. J.: Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society 181*, 3 (1977), 375–389.

[HA06] HU X. Y., ADAMS N. A.: A multi-phase sph method for macroscopic and mesoscopic flows. *Journal of Computational Physics 213*, 2 (2006), 844–861.

[HEW15] HUBER M., EBERHARDT B., WEISKOPF D.: Boundary handling at cloth–fluid contact. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 14–25.

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics in complex shapes. In *Proceedings of the 23rd Spring Conference on Computer Graphics* (2007), ACM, pp. 191–197.

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: Sph fluids in computer graphics.

[KB17] KOSCHIER D., BENDER J.: Density maps for improved sph boundary handling. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2017), ACM, p. 1.

[Luc77] LUCY L. B.: A numerical approach to the testing of the fission hypothesis. *The astronomical journal 82* (1977), 1013–1024.

[MCG03] 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), Eurographics Association, pp. 154–159.

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 1 (1992), 543–574.

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on progress in physics 68*, 8 (2005), 1703.

[Mon12] MONAGHAN J.: Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics 44* (2012), 323–346.

[MT05] MÖLLER T., TRUMBORE B.: Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses* (2005), ACM, p. 7.

[Ott18] OTT E.: Particle initialization and coupling methods for sph-based fluid animations.

[RHEW17] REINHARDT S., HUBER M., EBERHARDT B., WEISKOPF D.: Fully asynchronous SPH simulation. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2017), pp. 2:1–2:10. URL: `http://doi.acm.org/10.1145/3099564.3099571`, `doi:10.1145/3099564.3099571`.

[Sam90] SAMET H.: Applications of spatial data structures.

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Transactions on Graphics 31*, 4 (2012), 61–68.

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 129–136.

[SW92] SCHWAB C., WENDLAND W. L.: Kernel properties and representations of boundary integral operators. *Mathematische Nachrichten 156*, 1 (1992), 187–218.