Liquid Boundaries for Implicit Incompressible SPH

Jens Cornelis, Markus Ihmsen, Andreas Peer, Matthias Teschner

Department of Computer Science, University of Freiburg, Germany

Abstract

We propose a novel unified particle representation for fluids and solid boundaries in Implicit Incompressible SPH (IISPH). In contrast to existing particle representations, the proposed concept does not require a separate processing of fluid and boundary particles. On one hand, this results in a simplified solver implementation with improved efficiency. On the other hand, the unified fluid and boundary representation adds flexibility to IISPH which enables versatile effects. In particular, particles can now dynamically interchange their role between fluid and boundary which we therefore refer to as liquid boundary. The paper mainly focuses on the description of the unified representation and on the application of the concept to visual effects such as solidification and liquefaction. To support the realization of these effects, the concept of unified fluid and liquid boundary particles is extended to a third particle type, so-called candidate particles that are used in a transition phase between fluid and liquid boundaries.

Keywords: computer animation, fluid animation, physically-based simulation

1. Introduction

Boundary handling in fluid simulations based on Smoothed Particle Hydrodynamics (SPH) is a challenging topic and numerous approaches have been developed, e.g., [1, 2, 3, 4, 5, 6, 7, 8]. Although there exist various ways to represent solid boundaries, particle representations seem to be preferred in SPH fluids. E.g., in the context of the state-of-the-art pressure solver IISPH, boundary particles are incorporated into the proposed Jacobi solver to handle the interaction between fluid and rigid objects [2, 8, 9]. The introduced boundary particles, however, require a specific processing and in general, fluid and boundary particles are not easily interchangeable.

We address this issue and propose to use only one unified particle type to represent fluid and solid boundaries. This simplifies the solver implementation which is also accompanied by some performance gain especially in parallelized settings. The main benefit, however, is the introduction of additional degrees of freedom in terms of animation effects. Fluid and liquid boundary particles are now dynamically interchangeable which enables visual effects such as solidification and liquefaction. The unified particles can stably change their role between fluid and liquid boundary instantly at any time during a simulation. Stabilization schemes for the IISPH pressure solver are not required. In order to allow for additional animation effects, a third particle role, so-called candidates, are introduced that can be used in the transition phase between fluid and liquid boundary. Candidates are not required for stability purposes, but might be used to guide solidification processes, e.g., when a fluid volume is supposed to take a prescribed shape that should act as liquid boundary (see Figure 1).

In order to be able to switch between particle roles at any time, the density errors at fluid, candidate and liquid boundary particles have to be manageable by the IISPH pressure solver in a stable way. For fluid particles, this is naturally the case. For candidates and liquid boundaries, however, the respective properties have to be implemented on condition that the density error does not negatively influence the stability of IISPH if the particle role changes. I.e., liquid boundary particles should act as boundary for fluid particles, but it should also be guaranteed that a role change from liquid boundary to fluid does not affect the stability of IISPH. Therefore, we propose to use a grid with cell sizes that correspond to the particle size and to align liquid boundary particles to the grid as indicated in Figure 4. This guarantees that liquid boundary particles are appropriately sampled and that a transition to fluid particles would not be hazardous to the stability of the simulation. For candidate particles, the situation is similar. On one hand, they should move to prescribed sample positions where they are typically transformed to liquid boundary particles. On the other hand, it should also be possible to transform them to fluid particles at any time. This is realized by incorporating candidates into the IISPH pressure solver where the standard computation of velocity differences from non-pressure forces is replaced by an animated velocity to a target position. This guarantees that the computed pressure forces do not only preserve an appropriate density at fluid particles, but also at candidates.

The paper mainly focuses on the description how to combine fluid, candidate and liquid boundary particles in a unified way within IISPH. It particularly shows how to realize the functionality of candidate and liquid boundary particles in a way that enables arbitrary role changes at any time in a stable way. Experiments illustrate some of the animation effects that can be realized with the proposed concept of unified particles.

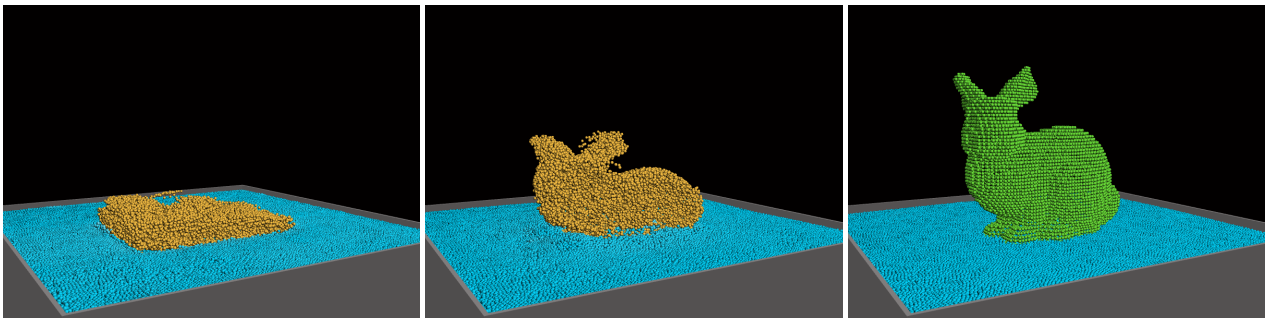

Figure 1: Fluid volume taking the shape of a rigid and serving as liquid boundary. **Yellow particles**: candidate particles animated towards their positions on the liquid boundary. **Green particles**: liquid boundary particles.

2. Related Work

SPH has developed into a popular approach for Lagrangian simulations, e.g. [1, 10, 3]. Nowadays, boundary handling is one of the main research foci to realize fluid-solid interactions in a stable and efficient way. While some approaches compute distance-based penalty forces, e.g. [4, 5], the majority of recent techniques relies on density-based pressure forces where the density computation at fluid particles considers contributions from particles that sample the boundary. Boundary particles are either generated by constraining fluid particles to static positions [6] or by mirroring particles across the boundary [7], which generally addresses time step restrictions of distance-based penalty forces. When particles are constrained to static positions as frozen particles, either more than one layer of boundary particle is used [11] or the positions of particles that intersect the rigid are corrected [9]. The concept has also been used for fluid-air interfaces in [12], where a layer of ghost particles is not only generated for solids, but also to represent air at the free surface. In [13], ghost particles at the free surface are avoided. Instead, approximate forces are efficiently computed to account for ghost particles [13].

Our unified particle approach is based on IISPH [2] which uses a boundary handling that has been introduced in [8]. Solid boundaries are sampled with particles and density-based hydrodynamic forces are computed to handle the interaction of fluid particles with adjacent boundary particles. Although IISPH uses particles for the fluid and for solid boundaries, they are processed in different ways and it is not possible to, e.g., process or consider a boundary particle as a fluid particle. We address this issue by proposing unified particles for the fluid and for solid boundaries. We extend the concept to animated candidate particles and illustrate the utility of the concept with visual effects that could not be realized with standard IISPH.

Using unified particles in different roles as fluid, candidate or liquid boundary enables us to dynamically change the role of a particle which can be used in versatile animation effects. Therefore, our approach is closely related to recent research that addresses fluid control. In this context, [14] introduced three levels of embedded controllers in order to apply scalar pressure fields on a grid-based fluid. An alternative technique to animate the fluid flow based on control particles has been presented in [15]. Additionally, a low-pass filter is applied on the velocity field in order to separate fine-scale detail and preserve small-scale fluid motion. Recently, position-based fluids [16] have been used for fluid control by [17]. Increasing the artistic freedom in fluid control has been addressed in [18], where the animator can specify the desired fluid pose. Here, the simulation result is modified by solving a small-scale linear optimization problem. Different from that, [19] applies a divergence-free force field which represents the gradient field of a potential function defined by the shape of a target. While our paper focuses on unified particles for IISPH, the utility of the concept is shown with techniques for fluid control that shares similarities with [19]. Instead of applying a force field, however, we adapt the velocity of candidate particles in order to direct them towards a desired position. The subsequent IISPH pressure solve ensures that the final velocity field is divergence-free.

Freezing SPH fluid particles to form solid objects or melting them back to a liquid state has been discussed in various publications, e.g. [20, 21]. In these works, particles are arranged in a locally defined lattice using elastic restoration forces. Later, [22] proposed to simulate solids with SPH. Liquefaction and solidification effects have been presented. In contrast, our paper does not focus on a physically based simulation of solids. Instead, we focus on a stable and efficient fluid-solid interaction with unified particles. Nevertheless, our concept also allows to realize liquefaction and solidification effects.

In the context of position-based dynamics [23], a unified particle physics framework has been presented in [24]. The concept can be used to simulate gases, liquids, deformable solids and rigid bodies with particles by employing various position-based constraint formulations. In contrast, our concept is based on the IISPH pressure solver. Further, the role of a particle can be changed instantly during a simulation without introducing stability issues to the pressure solver. A divergence-free velocity field is computed for all particles independent from their current role.

3. Concept

Our method considers different roles for particles. *Fluid particles* f represent the fluid body. *Liquid boundary particles* l represent boundaries which can be the surface or the volume of a solid shape. *Candidate particles* c are animated particles that obey the density constraint of the fluid body. They are typically

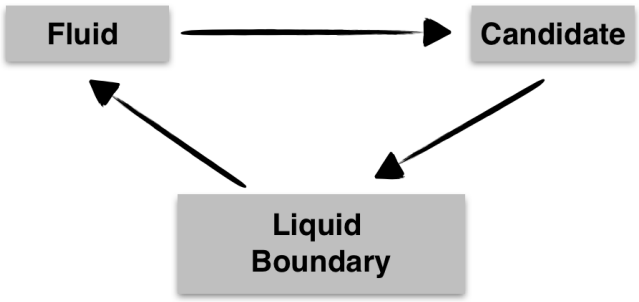

Figure 2: Roles and role transitions that are considered in our implementation.

used in the transition phase from fluid to liquid boundary when former fluid particles should move to valid sample positions of liquid boundary particles. When generating a liquid boundary, candidates are typically chosen as the nearest neighbor from the fluid body for each sample position at the rigid object. Although arbitrary role transitions are possible, we focus on the transitions depicted in Figure 2 as they have been used in the presented scenarios.

In the following, we explain the assignment of roles to particles with a focus on candidates and liquid boundaries that we introduce to IISPH. We further describe the transitions between different roles and finally discuss implementation details in IISPH.

3.1. Role assignment

Liquid boundaries. Rigid objects are discretized into uniform grid cells. These grid cells are potential sample positions x_{rigid} for liquid boundary particles.

Candidates. This particle role supports the transition of a particle from fluid to liquid boundary. If a rigid object should be sampled with liquid boundary particles, we search the nearest fluid particle for each sample position x_{rigid} and update its role to a candidate particle. The search is accelerated with a k-d tree [25] and restricted to a user-defined axis-aligned bounding box. After changing the role of a particle from fluid to candidate and assigning the particle to the sample position in the rigid body, it is not considered in the search for the next sample point. Thus, a candidate is assigned to exactly one liquid boundary position. If there is a sufficient number of fluid particles, each liquid boundary sample has one candidate. Candidate particles are characterized by a velocity of user-defined magnitude towards the assigned sample position x_{rigid}. If the distance of a candidate to its assigned position is below a threshold, the candidate turns to a liquid boundary particle.

3.2. Role transition

The main issue in role transitions is the density at a particle. This is due to the fact that IISPH computes pressure forces that counteract density deviations. Thus, density deviations negatively affect the stability of the simulation and the simulation timestep, i.e. the efficiency. As pressure forces are computed at fluid particles, and candidates or liquid boundaries can transition to fluid particles at any time, it is essential to minimize density deviations at all particles to prevent unstable accelerations in case of a role transition.

Candidates. The density deviation at candidates is handled by the IISPH pressure solver. However, in contrast to fluid particles, the effect of non-pressure forces on the velocity is not considered. Instead, the animated velocity towards the sample position in the liquid boundary is taken to initialize the pressure solve.

Liquid boundaries. In general, the boundary sampling has to fulfill two requirements. First, the sampling has to be sufficiently dense to avoid fluid leakage. Second, oversampled boundaries should be avoided for performance reasons. As discussed in [26], a sampling is appropriate, if the boundary particle number density δ_b is within the range from $3.6\Gamma(h)$ to $10\Gamma(h)$, where $\Gamma(h) = \frac{1}{h^3}$. In our context, oversampling is not acceptable as the pressure solver would not be able to handle candidates with large density deviations when they are close to their sampling position in the boundary. Further, liquid boundaries could not be transformed to fluid particles without serious stability issues. Therefore, we propose a uniform grid with a cell size that corresponds to the particle diameter. The grid is implemented as a list using spatial hashing as described in [27]. The grid avoids over- and undersampling of the boundary and allows the stable transition of liquid boundaries to fluid particles. An example scenario that compares an appropriate and an oversampled liquid boundary is shown in Figure 3.

The employed uniform grid is computed once for each rigid object in a scenario. Rigid-body transformations, i.e. translation and rotation, are just applied to the grid. Deformable solids are not considered. This significantly reduces the computational cost of the liquid boundary computation. If the liquid boundary is fully sampled with particles, the computational costs of the liquid boundary handling correspond to the IISPH implementation with standard boundary particles.

In order to populate the grid, the rigid surface is sampled with [26]. However, instead of generating boundary particles at the respective sample positions, we mark the cells as liquid boundary cells as visualized in Figure 4. Alternative techniques such as ray casting could also be applied. For volumetric shapes, the fluid volume that represents the object should match the respective volume of the object. Thus, it is not sufficient to only represent the surface of the object with liquid boundary cells, but also the interior of the object. In order to determine the cells that are inside the object, we propose to use the voting algorithm in [28].

4. Implementation

In the following, we give an overview of IISPH. It is shown how to compute the intermediate velocities for all particle roles. It is further explained, how to compute the density for all particle roles. Finally, the modified pressure force computation is explained.

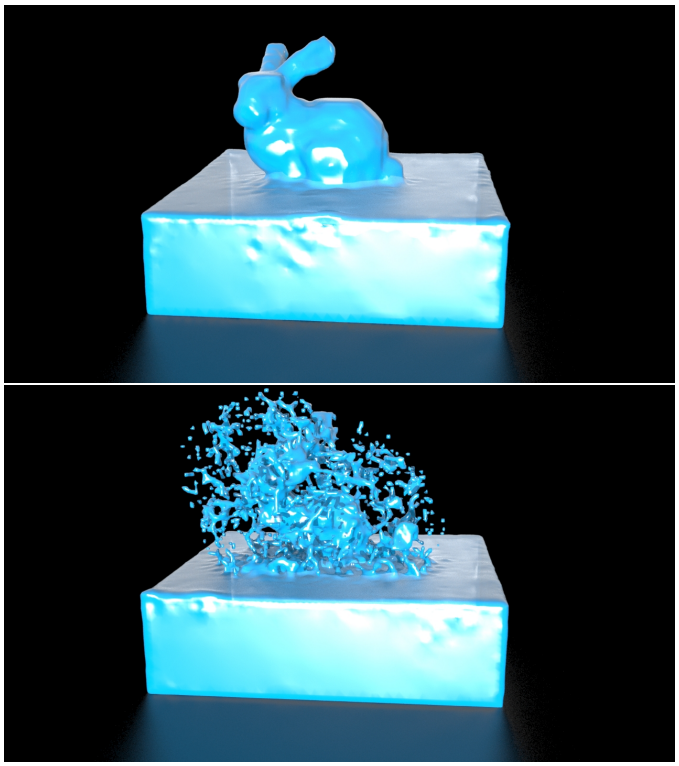

Figure 3: Liquid boundary sampling. The first image shows the transition of a liquid boundary to fluid particles with an appropriate sampling using the proposed uniform grid. The second image shows the instabilites that occur in the transition of oversampled liquid boundaries to fluid particles.

4.1. IISPH

The original IISPH formulation can be derived from a discretization of the continuity equation $\frac{D\rho}{Dt} = -\rho\nabla\mathbf{v}$ at time t. The time derivative is approximated with $\frac{\rho_i(t+\Delta t)-\rho_i(t)}{\Delta t}$, where $\rho_i(t)$ is the density of particle i at time t. The spatial derivative is approximated with SPH as $\nabla\mathbf{v}_i = -\frac{1}{\rho_i}\sum_j m_j(\mathbf{v}_i(t+\Delta t) - \mathbf{v}_j(t+\Delta t))\nabla W_{ij}$, where m_j denotes the mass of the neighboring particle j. $W_{ij}(t) = W(\mathbf{x}_i(t) - \mathbf{x}_j(t))$ is a kernel function with finite support. This results in the following discretized form of the continuity equation:

$$\frac{\rho_i(t+\Delta t) - \rho_i(t)}{\Delta t} = \sum_j m_j(\mathbf{v}_i(t+\Delta t) - \mathbf{v}_j(t+\Delta t))\nabla W_{ij}(t). \tag{1}$$

The unknown velocities in (1) can be replaced by unknown pressure forces $\mathbf{F}_i^p(t)$ by using $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^{adv} + \frac{\mathbf{F}_i^p(t)}{m_i}$, where the intermediate velocities $\mathbf{v}_i^{adv}$ can be computed as

$$\mathbf{v}_i^{adv} = \mathbf{v}_i(t) + \Delta t\frac{\mathbf{F}_i^{adv}(t)}{m_i}. \tag{2}$$

Applying the intermediate velocities results in intermediate densities:

$$\rho_i^{adv} = \rho_i(t) + \Delta t\sum_j m_j(\mathbf{v}_i^{adv} - \mathbf{v}_j^{adv})\nabla W_{ij}(t). \tag{3}$$

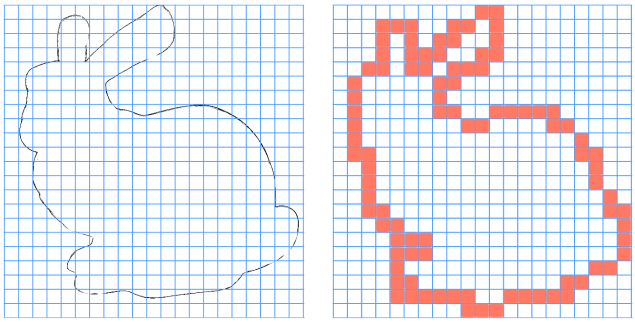

Figure 4: The surface of the object is represented on a grid. Cells that represent boundary are marked as boundary cells. Afterwards, internal cells can be marked as boundary cells using scanline voting.

Assuming that the desired density of the fluid is the rest density, i.e. $\rho_i(t+\Delta t) = \rho_0$, leads to

$$\Delta t^2\sum_j m_j\left(\frac{\mathbf{F}_i^p(t)}{m_i} - \frac{\mathbf{F}_j^p(t)}{m_j}\right)\nabla W_{ij}(t) = \rho_o - \rho_i^{adv}$$

Given that $\mathbf{F}_i^p(t) = -m_i\sum_j m_j(\frac{p_i(t)}{\rho_i^2(t)} + \frac{p_j(t)}{\rho_j^2(t)})\nabla W_{ij}(t)$, this is a linear system with unknown pressure value $p_i(t)$ per particle. In the following, we extend this linear system to different particle roles.

4.2. Intermediate velocities

In order to incorporate different particle roles f, c, and l into IISPH, we apply the splitting concept to all particle roles by first computing velocities $\mathbf{v}^{adv}$ without pressure forces. We further compute the respective intermediate densities ρ^{adv}. This step corresponds to Eq. (2) in standard IISPH.

Candidate particles c are given an animation velocity towards their assigned sample position $\mathbf{x}_{rigid}$ as

$$\mathbf{v}_c^{ani} = \Delta t\left(\frac{\mathbf{x}_{rigid}(t+\Delta t) - \mathbf{x}_c(t)}{\|\mathbf{x}_{rigid}(t+\Delta t) - \mathbf{x}_c(t)\|}\right)\beta, \tag{4}$$

where β is a user-defined parameter. This can be seen as applying a spring force to a candidate into the direction of the assigned liquid boundary position. Instead of just using β as a constant value, it is also possible to apply a distance-based easing function. In this case, particles will not move towards the liquid boundary with linear speed which will make the animation look more organic. Following the IISPH concept, the intermediate velocity of a candidate is computed as

$$\mathbf{v}_c^{adv} = (1-\alpha)\mathbf{v}_c^* + \alpha\mathbf{v}_c^{ani} \tag{5}$$

with $\mathbf{v}_c^* = \mathbf{v}_c(t) + \Delta t\frac{\mathbf{F}_c^{adv}(t)}{m_c}$. It is crucial, that candidates are incorporated into the subsequent pressure solve to guarantee low density deviations at these particles. Otherwise, unstable pressure forces might occur in case of a role transition. Large density deviations would also be perceived as volume loss and reduce the plausibility of the animation.

The velocity of liquid boundaries l is computed from the linear and angular velocity of the rigid body. For fluid particles f, the intermediate velocity is computed as $\mathbf{v}_f^{adv} = \mathbf{v}_f(t) + \Delta t\frac{\mathbf{F}_f^{adv}(t)}{m_f}$.

4.3. Density computation

The advection of particles with $\mathbf{v}^{adv}$ contributes to the density deviation that is minimized by the subsequent pressure solve. We extend Eq. (3) and predict the intermediate density for both fluid particles f and candidate particles c with respect to all neighboring fluid particles j, liquid boundary particles l and candidate particles c as

$$\begin{aligned}\rho_f^{adv} = &\sum_j m_j W_{fj} + \sum_c m_c W_{fc} + \sum_l m_l W_{fl} \\ &+ \Delta t \sum_j m_j \mathbf{v}_{fj}^{adv} \nabla W_{fj} \\ &+ \Delta t \sum_c m_c \mathbf{v}_{fc}^{adv} \nabla W_{fc} \\ &+ \Delta t \sum_l m_l \left(\mathbf{v}_f^{adv} - \mathbf{v}_l(t+\Delta t)\right) \nabla W_{fl}. \end{aligned} \tag{6}$$

This predicted density error is resolved by IISPH.

4.4. Pressure forces

Finally, pressure forces are derived from the pressure that has been computed by IISPH. The displacement of a fluid particle f and a candidate particle c considering neighboring fluid particles j, candidate particles c and liquid boundary particles l is computed as

$$\begin{aligned}\Delta t^2 \frac{\mathbf{F}_f^p}{m_f} = &\sum_j -\Delta t^2 \frac{m_j}{\rho_j^2} \nabla W_{fj} p_j + \sum_c -\Delta t^2 \frac{m_c}{\rho_c^2} \nabla W_{fc} p_c + \\ &(-\Delta t^2 \sum_j \frac{m_j}{\rho_f^2} \nabla W_{fj} - \Delta t^2 \sum_l \frac{m_l}{\rho_f^2} \nabla W_{fl} - \\ &\Delta t^2 \sum_c \frac{m_c}{\rho_f^2} \nabla W_{fc}) p_f. \end{aligned}$$

The pressure force of a candidate c is computed accordingly. The resulting pressure forces are used for updating velocities and positions of fluid particles f and candidate c. In terms of liquid boundaries l, the velocities have been previously computed according to the velocity of the rigid object which leads to updated positions in the integration step.

The algorithm is outlined in Algorithm 1.

Algorithm 1 IISPH with liquid boundary using relaxed Jacobi. l indicates the iteration.

procedure ROLE ASSIGNMENT
  for all *particle i* **do**
    test candidate / liquid boundary (see Section 3.1)
procedure COMPUTE DENSITY
  for all *particle i* **do**
    if *particle i* $\in$ fluid particles $\vee$ candidates **then**
      compute $\rho_i(t) = \sum_j m_j W_{ij}(t)$
procedure COMPUTE INTERMEDIATE VELOCITY
  for all *particle i* **do**
    if *particle i* $\in$ fluid particles **then**
      predict $\mathbf{v}_i^{adv} = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{adv}(t)}{m_i}$ (2)
    else if *particle i* $\in$ candidates **then**
      compute $\mathbf{v}_i^{ani}$ (4)
      predict $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{adv}(t)}{m_i}$ (2)
      $\mathbf{v}_i^{adv} = (1-\alpha)\mathbf{v}_i^* + \alpha \mathbf{v}_i^{ani}$ (5)
    else if *particle i* $\in$ liquid boundary **then**
      $\mathbf{v}_i(t+\Delta t) = v_{rigid}(t+\Delta t, x_i)$
procedure COMPUTE INTERMEDIATE DENSITY
  for all *particle i* **do**
    if *particle* $\in$ fluid particles $\vee$ candidates **then**
      compute ρ_i^{adv} (6)
procedure PRESSURE SOLVE WITH IISPH
procedure INTEGRATION
  for all *particle i* $\in$ fluid particles $\vee$ candidates **do**
    $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^{adv} + \Delta t \mathbf{F}_i^p(t)/m_i$
    $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t)$
  for all *particle i* $\in$ liquid boundary **do**
    $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t)$

5. Results

5.1. Implementation

The proposed concept is an extension of IISPH [2] and of the boundary handling presented in [8]. The neighborhood search is realized with the data structures proposed in [29]. [30] is used for surface tension. The implementation is fully parallelized [29]. For the surface reconstruction, we apply [31]. The rendering of the final images and videos is performed with Houdini and Mantra [32]. All scenes have been computed on a 16-core 3.46 GHz Intel i7. For our simulations, we choose $\alpha = 0.5$ and we use a non-linear easing curve for β. We use a threshold of 0.3% for the average density error in our simulations.

5.2. Parameter discussion

We introduced two parameters: β is the animation velocity of candidate particles that move towards their assigned position on the liquid boundary, whereas α is a blending parameter for the linear combination of the previous particle velocity with the animation velocity.

In our test scenarios, we chose β in such way, that it does not negatively affect the CFL number of the simulation. This means, that β is chosen such that the velocity of candidate particles does not exceed the maximum velocity of fluid particles. Smaller time steps will be necessary for larger values of β. However, since β can be seen as a spring force, the value of β also determines, how long it takes for particles to move towards the rigid object shape.

We used the bunny scene with 500k particles and tested different values of α. Since α blends between a previous particle velocity and the desired animation velocity, this parameter is crucial for the simulation. We chose different values for testing the behavior of α. For $\alpha = 1.0$, the particle velocity is completely overwritten by the animation velocity. When we apply $\alpha = 0.75$ and $\alpha = 0.5$, the animation velocity is introduced less aggressive. Please note, that the animation takes slightly longer for smaller values of α, since the animation speed of candidates

is damped by this parameter. However, the visual difference of the simulation is imperceptible.

Since candidates are incorporated into the regular IISPH update, their velocity is corrected such that the density error stays below the given threshold of 0.3%. The density for $\alpha = 1.0$ is 1002.108 on average, while it is 1002.074 for $\alpha = 0.5$. Since the applied animation velocity field is not divergence free, more iterations are necessary for resolving the density error introduced by large α values. IISPH solves for the estimated density error that is introduced by velocities due to advection. This error is additionally increased when a divergent velocity field is applied. While on average only 18 iterations were necessary for $\alpha = 0.5$, the IISPH solver took 62 iterations for resolving the density error in case of $\alpha = 1.0$. We choose $\alpha = 0.5$ for our simulations, since it provides a stable trade-off between performance and simulation stability. Even smaller values of α do not positively affect the simulation, but significantly slow down the liquid boundary assembly.

5.3. Application examples

The cup scene as depicted in Figure 5 shows a fluid that morphs to the shape of a cup. The fluid-solid interface built upon liquid boundary particles prevents fluid particles from penetrating the cup shape. Thus, the cup can be filled with fluid. Transformations of the rigid object are reflected by the liquid boundary interface. Finally, the disassembly of the liquid boundary with a vortex force is shown. The scene is simulated with 1.3 million particles. The time step is 0.005 seconds for a particle spacing of 0.025 meters. The computation time is 20 seconds per frame. We compute 50 frames per movie second.

The ship scene demonstrates the performance of the method with a large-scale scene that consists of 12 million particles in average. The interior of the ship is sampled with liquid boundary cells and represented with liquid boundary particles in order to have a volumetric representation. The time step for the simulation of this scene is 0.01 seconds with a particle spacing of 0.05 meters. One frame is computed in 190 seconds. An illustration of the scene can be seen in Figure 6

6. Conclusion and Limitations

We have shown how to incorporate unified particles with various roles into IISPH [2]. We use candidates that are animated by applying a fictitious animation velocity in order to take the shape of a rigid. The fluid-solid interface is resolved with liquid boundaries that prevent fluid particles from penetrating the rigid shape.

When assigning a new role to a particle, certain requirements have to be fulfilled. In SPH, this is typically a low density deviation as large density errors negatively affect the stability and efficiency of a simulation. While the density of a candidate is resolved by the pressure solve, liquid boundaries have to be processed in a different way. We have proposed to position liquid boundaries on a uniform grid that discretizes the solid objects boundary. This avoids density error due to an over- or undersampling of the boundary.

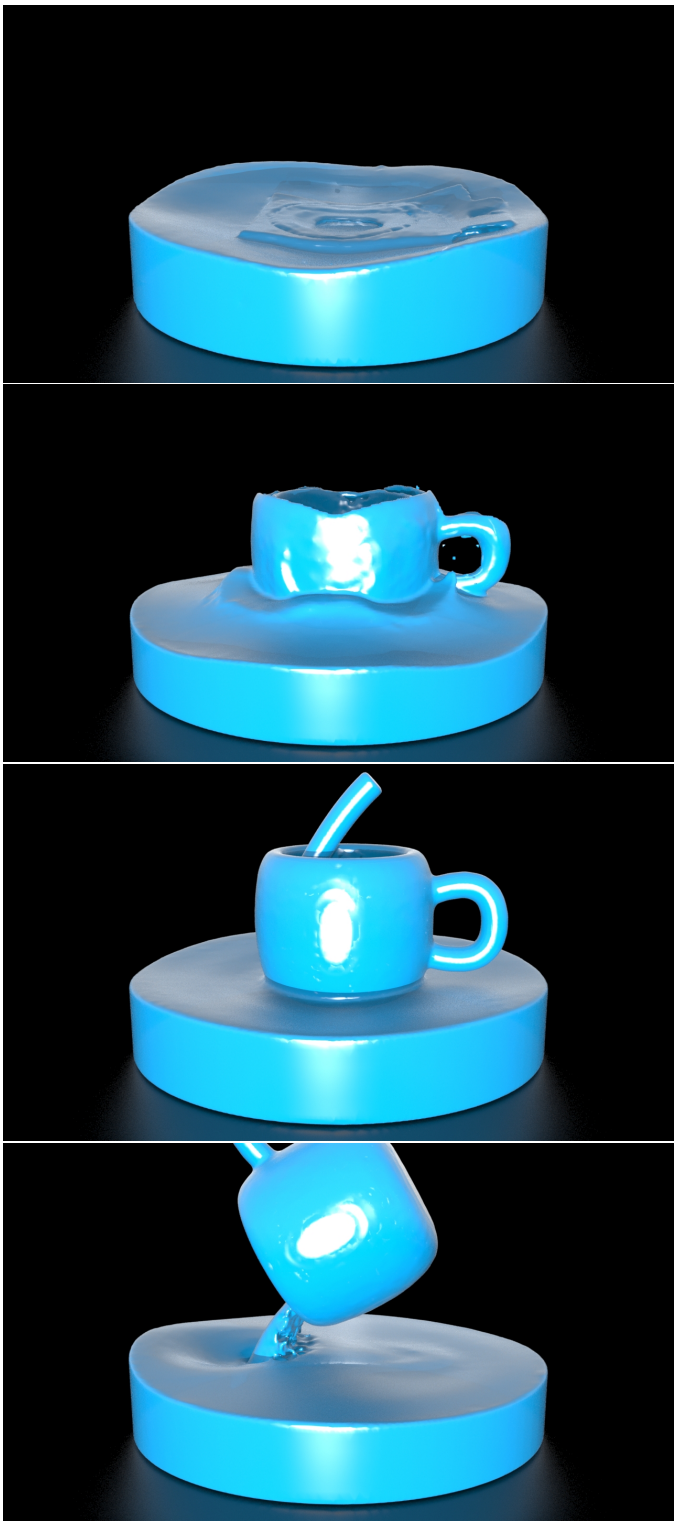

Figure 5: **Cup scene:** Fluid is animated to take the shape of a cup. The liquid boundary can be used for fluid-solid interaction and the shape can be animated. The liquid boundary shape is filled with fluid without any leakage. The simulation consists of 1.3 million particles.

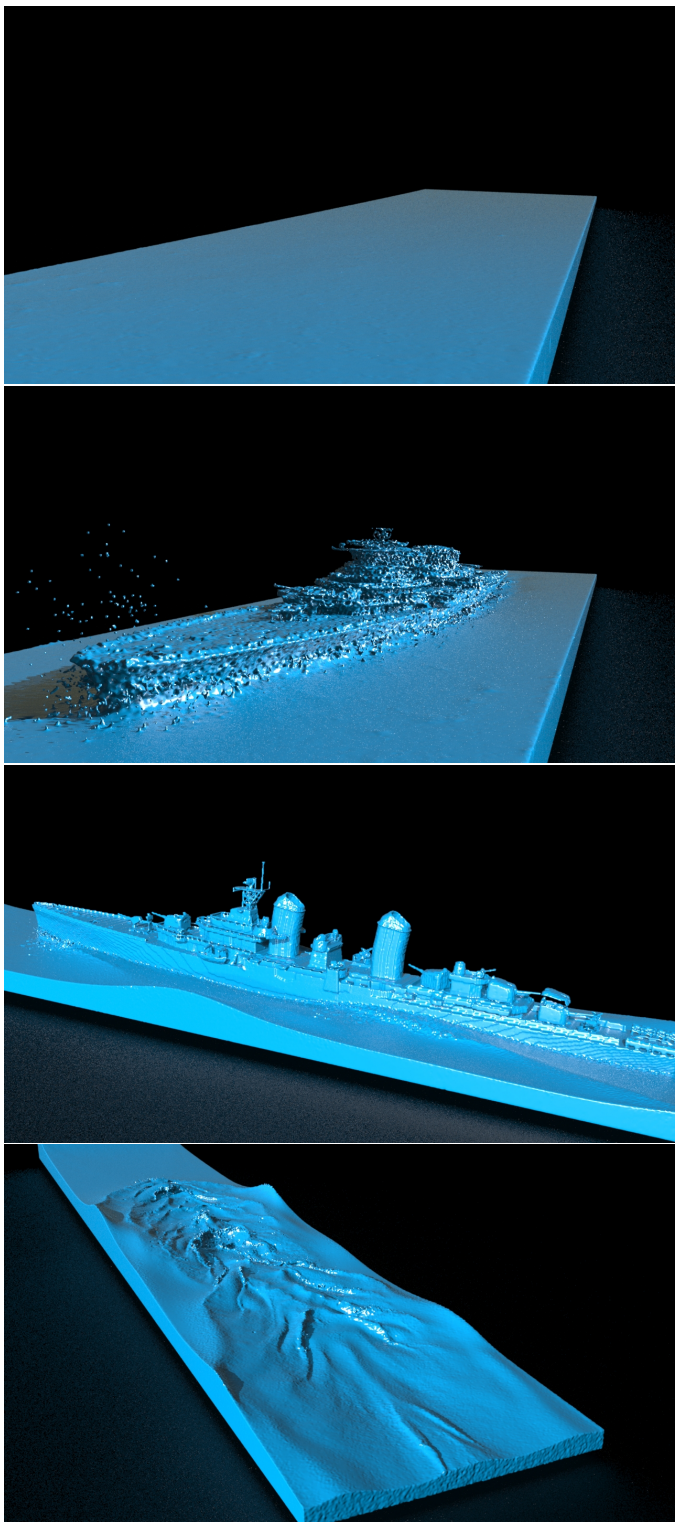

Figure 6: **Ship scene:** Large-scale scene with 12 million particles that shows a ship rising from a fluid stream and tumbling back to fluid state again.

The liquid boundary avoids fluid leakage and thus, it is possible to form solid shapes that contain fluid bulk even when a rigid object is animated. We have shown the strength and stability of our approach with various applications.

Although the uniform grid data structure prevents density errors at the liquid boundary, it introduces undesired aliasing patterns in terms of the surface reconstruction. Thus, different grid patterns could be analyzed. The same holds for the nearest neighbor search using a k-d tree [25]. In our simulations, only one search was necessary since all liquid boundary cells could be provided with a candidate particle in just one step. More advanced techniques could be investigated for other scene setups.

Currently, only one fluid particle is assigned to one sample position in the liquid boundary. Although using an easing curve makes the animation look less artificial, more impressive methods for animating the fluid in order to take the rigid shape could be investigated. In some scenarios, it might be even better to assign multiple candidates per liquid boundary sample in order to show fluid rinsing from the rigid object boundary.

Lastly, only rigid objects are considered. Deformable shapes require additional considerations that were not within the scope of this paper.

References

[1] Monaghan J. Smoothed particle hydrodynamics. Ann Rev Astron Astrophys 1992;30:543–74.

[2] Ihmsen M, Cornelis J, Solenthaler B, Horvath C, Teschner M. Implicit incompressible SPH. IEEE Transactions on Visualization and Computer Graphics 2014;20(3):426–35.

[3] Ihmsen M, Orthmann J, Solenthaler B, Kolb A, Teschner M. SPH Fluids in Computer Graphics. In: Eurographics 2014 - State of the Art Reports. The Eurographics Association; 2014, p. 21–42.

[4] Müller M, Schirm S, Teschner M, Heidelberger B, Gross M. Interaction of fluids with deformable solids. Computer Animation and Virtual Worlds 2004;15:159–71.

[5] Monaghan J, Kajtar J. SPH particle boundary handling for arbitrary boundaries. Computer Physics Communications 2009;180(10):1811–20.

[6] Libersky L, Petschek A. Smoothed Particle Hydrodynamics with strength of materials. Advances in the Free-Lagrange Method Including Contributions on Adaptive Gridding and the Smoothed Particle Hydrodynamics Method 1991;395:248–57.

[7] Hu X, Adams N. A multi-phase SPH method for macroscopic and mesoscopic flows. Journal of Computational Physics 2006;213:844–61.

[8] Akinci N, Ihmsen M, Akinci G, Solenthaler B, Teschner M. Versatile rigid-fluid coupling for incompressible SPH. ACM Trans Graph 2012;31(4):062:1–062:12.

[9] Ihmsen M, Akinci N, Gissler M, Teschner M. Boundary handling and adaptive time-stepping for PCISPH. In: Proceedings VRIPHYS. 2010, p. 79–88.

[10] Desbrun M, Cani MP. Smoothed particles: A new paradigm for animating highly deformable bodies. In: Eurographics Workshop on Computer Animation and Simulation (EGCAS). Springer-Verlag; 1996, p. 61–76. Published under the name Marie-Paule Gascuel.

[11] Dalrymple R, Knio O. SPH modeling of water waves. In: Coastal Dynamics. 2001, p. 779–87.

[12] Schechter H, Bridson R. Ghost sph for animating water. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 2012;31(4).

[13] Ren B, Yan X, Yang T, Li Cf, Lin M, Hu Sm. Fast SPH simulation for gaseous fluids. The Visual Computer 2015;:1–12.

[14] Foster N, Metaxas D. Controlling fluid animation. In: Computer Graphics International, 1997. Proceedings. 1997, p. 178–88.

[15] Thürey N, Keiser R, Pauly M, Rüde U. Detail-preserving fluid control. Graphical Models 2009;71(6):221–8. 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006).

[16] Macklin M, Müller M. Position based fluids. ACM Transactions on Graphics (SIGGRAPH 2013) 2013;32(4):104:1–104:12.
[17] Zhang S, Yang X, Wu Z, Liu H. Position-based fluid control. In: Proceedings of the 19th Symposium on Interactive 3D Graphics and Games. i3D '15; New York, NY, USA: ACM; 2015, p. 61–8.
[18] Pan Z, Huang J, Tong Y, Zheng C, Bao H. Interactive localized liquid motion editing. ACM Trans Graph 2013;32(6):184:1–184:10.
[19] Shi L, Yu Y. Taming liquids for rapidly changing targets. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '05; 2005, p. 229–36.
[20] Müller M, Keiser R, Nealen A, Pauly M, Gross M, Alexa M. Point based animation of elastic, plastic and melting objects. In: ACM SIGGRAPH / Eurographics Symposium on Computer Animation. 2004, p. 141–151.
[21] Wicke M, Hatt P, Pauly M, Mueller M, Gross M. Versatile virtual materials using implicit connectivity. In: Eurographics Symposium on Point-Based Graphics. 2006, p. 137–44.
[22] Solenthaler B, Schlafli J, Pajarola R. A unifed particle model for fuid-solid interactions. Computer Animation and Virtual Worlds 2007;18:69–82.
[23] Bender J, Müller M, Otaduy M, Teschner M, Macklin M. A survey on position-based simulation methods in computer graphics. Computer Graphics Forum 2014;33(6).
[24] Macklin M, Müller M, Chentanez N, Kim TY. Unified particle physics for real-time applications. ACM Transactions on Graphics (TOG) 2014;33(4):153:1–153:12.
[25] Bentley JL. Multidimensional binary search trees used for associative searching. Commun ACM 1975;18(9):509–17.
[26] Akinci N, Cornelis J, Akinci G, Teschner M. Coupling elastic solids with smoothed particle hydrodynamics fluids. Computer Animation and Virtual Worlds 2013;24:195–203.
[27] Teschner M, Heidelberger B, Müller M, Pomeranets D, Gross M. Optimized Spatial Hashing for Collision Detection of Deformable Objects. VMV 2003 2003;:47–54.
[28] Nooruddin FS, Turk G. Simplification and repair of polygonal models using volumetric techniques. IEEE Transactions on Visualization and Computer Graphics 2003;9:191–205.
[29] Ihmsen M, Akinci N, Becker M, Teschner M. A parallel SPH implementation on multi-core CPUs. Computer Graphics Forum 2011;30(1):99–112.
[30] Becker M, Teschner M. Weakly compressible SPH for free surface flows. Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2007;:209–17.
[31] Ju T, Losasso F, Schaefer S, Warren J. Dual contouring of hermite data. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '02; ACM; 2002, p. 339–46.
[32] Side Effects Software . Houdini [software]. http://wwwsidefxcom/ 2013;URL `http://www.sidefx.com/`.