# Free Surface Flows with Moving and Deforming Objects for LBM

Nils Thürey, Klaus Iglberger, Ulrich Rüde

Institute for System Simulation (LSS),
University of Erlangen-Nuremberg, Cauerstr. 6, 91058 Erlangen, Germany
Email: {nils.thuerey, klaus.iglberger, ulrich.ruede}@cs.fau.de

## Abstract

We will present an algorithm to handle moving and deforming objects for free surface fluid simulations with the *lattice Boltzmann method* (LBM). To achieve this we extend methods available for flows without a free surface to enables simulations of moving objects with varying surface roughness, two-way coupled interaction and improved mass conservation. We furthermore show how to efficiently initialize boundary conditions for the moving objects from an arbitrary triangle mesh. The triangulation of the free surface can be improved by removing the interface between fluid and an immersed object. We will demonstrate the capabilities of our approach with two simpler test cases, and a more complicated simulation using an animated character as obstacle.

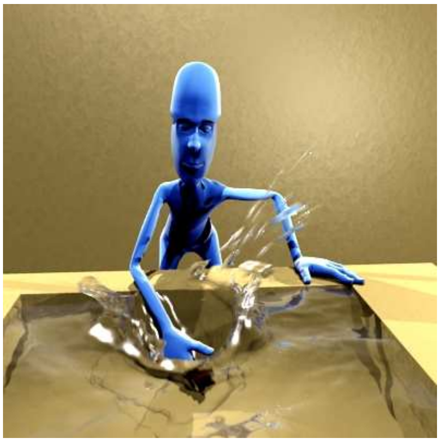

Figure 1: Example of interaction between an animated character and a volume of fluid.

## 1 Introduction & Related Work

The physically based animation of free surface fluids has become an important aspect for the creation of computer generated imagery. Since the works presented in [1] and [2] various approaches for simulating liquids have become popular and were extended to handle a variety of problems, such as the correct handling of moving and deforming objects. An example for such a case can be seen in Fig. 1, where an animated character interacts with a fluid.

While algorithms for handling coupled rigid body simulations with fluids have been developed for level-set based *Navier-Stokes* (NS) solvers [3] [4], we will in the following present an algorithm to handle these problems with the *lattice Boltzmann method* (LBM). It originates from discrete compressible gas simulations [5], and approximates the NS equations without the need for an iterative solver by relaxing the incompressibility constraint [6]. The algorithm has been used in various areas, e.g. for single phase fluid computations [7], or in combination with a VOF model to simulate free surface flows for metal foaming processes [8]. We will apply the same free surface model, that will be briefly explained in Section 2, and which was also used in e.g. [9]. Another popular approach to approximate the NS equations are *smoothed particle hydrodynamics* (SPH) solvers [10]. For this class of methods, deforming objects can be handled as e.g. shown in [11]. In the following, we will briefly explain the simulation of free surface fluids with LBM. Afterwards, boundary conditions for moving obstacles will be described in more detail. After the lattice initialization and surface generation we will discuss three different test simulations.

## 2 Free Surface Flows with LBM

The Lattice-Boltzmann Method (LBM) is a grid-based technique that was derived from discrete gas molecule simulations. Each grid cell stores a set of *distribution functions* (DFs), which represent an amount of fluid moving with a fixed velocity. We use the common three-dimensional incompressible LBM model *D3Q19i* with nineteen grid velocities $\mathbf{e}_i, i = 1...19$. The macroscopic fluid properties, such as density $\rho$ and velocity $\mathbf{v}$ can be calculated by summation of the DFs, and are needed to calculate the equilibrium DFs for a cell:

$$f_i^{eq}(\rho, \mathbf{v}) = w_i \left[ \rho + 3\mathbf{e_i} \cdot \mathbf{v} - \frac{3}{2}\mathbf{v}^2 + \frac{9}{2}(\mathbf{e_i} \cdot \mathbf{v})^2 \right] . \quad (1)$$

For the *D3Q19i* model, the weights $w$ are given as $w_1 = 1/3$, $w_{2..7} = 1/18$, and $w_{8..19} = 1/36$. The LBM algorithm proceeds by first handling the movement of the DFs, which is equivalent to copying them to their adjacent neighbors, and then computes the molecular collisions that would have occurred during this movement. The collisions are handled by a relaxation towards the equilibrium with a relaxation factor $\omega$ that is given by the physical fluid viscosity. First the streaming of the DFs is performed, which can be written as:

$$f_i(\mathbf{x}, t)' = f_i(\mathbf{x} - \mathbf{e}_i, t) \; . \quad (2)$$

Afterwards, the new post-collision DFs for the next timestep are computed as

$$f_i(\mathbf{x}, t + \Delta t) = \; (1 - \omega) f_i(\mathbf{x}, t)' + \; \omega f_i^{eq}. \quad (3)$$

An overview of the basic algorithm is shown in Fig. 2. More details of the algorithm and a derivation of the model can be found in [12, 13]. To simulate a fluid with a free surface we use a method similar to the *Volume-of-Fluid* approach for NS solvers. A detailed description of the free surface treatment can be found in various publications, e.g. [8, 14]. It computes a fill fraction $\epsilon$ for all interface cells, and updates these values according to the interface movement. Furthermore, the adaptive time stepping algorithm from [15] is applied.

The next sections will describe obstacle boundary conditions for the LBM. Similar to the free surface handling, they require additional flags to determine the type of a cell. The necessary equations and algorithms for initialization will be described in the following.

## 3 Obstacle Boundary Conditions

The standard way to implement obstacles for LBM fluid simulations is the *bounce-back* scheme for no-slip boundary conditions. They result in a zero normal and tangential velocity at the obstacle boundary. In terms of DFs this means that during the streaming step the DFs are reflected at the obstacle surface, thus Eq. (2) is changed to

$$f_i(\mathbf{x}, t + \Delta t)' = f_{\tilde{i}}(\mathbf{x}, t) \quad (4)$$

for all cells where the neighbor at $\mathbf{x} + \mathbf{e}_i$ is an obstacle cell. Note that $f_{\tilde{i}}$ denotes the DF along the inverse velocity vector of $f_i$, thus $\mathbf{e}_{\tilde{i}} = -\mathbf{e}_i$. These boundary conditions can be used to model ”sticky” walls, that slow down the fluid – e.g. rough stone surfaces.

Another type of boundary conditions are free-slip boundary conditions, which only result in a normal velocity of zero, but leave the tangential velocity at the obstacle interface unchanged. Hence, during the streaming step of the LBM, the DFs are reflected only along the obstacle normal, in contrast to Eq. (4), where they are reflected along normal and tangential direction. This free-slip scheme is shown to the right of Fig. 3. Note that while no-slip boundary conditions can be handled locally for a cell, free-slip conditions require DFs from a neighboring cell. Furthermore, the free-slip handling is equivalent to the no-slip boundaries if the neighboring cell is not a fluid cell. Free-slip boundary conditions can be used to model smooth surfaces that do not slow down the fluid, such as glass walls.

To model materials that have properties in between the two extremes described above, the free- and no-slip boundary conditions can be linearly interpolated. Given the reflected DFs $f_r$ from the free-slip treatment, this gives:

$$f_i(\mathbf{x}, t + \Delta t)' = w_p f_{\tilde{i}}(\mathbf{x}, t) + (1 - w_p) f_r(\mathbf{x}, t) \; , \quad (5)$$

where $w_p$ is the parameter to control the surface smoothness. For $w_p = 1$ Eq. (5) reduces to no-slip boundary conditions, while $w_p = 0$ yields a free-slip boundary.

The boundary conditions here have first order accuracy (for arbitrary shapes). Higher order boundary conditions, that take into account the position of the obstacle surface along each lattice connection, the most common second order one being [16], can also be applied for LBM. These, however, would significantly increase the computational overhead for moving obstacles, and Eq. (7) will demonstrate that the first order boundary conditions already give good results.

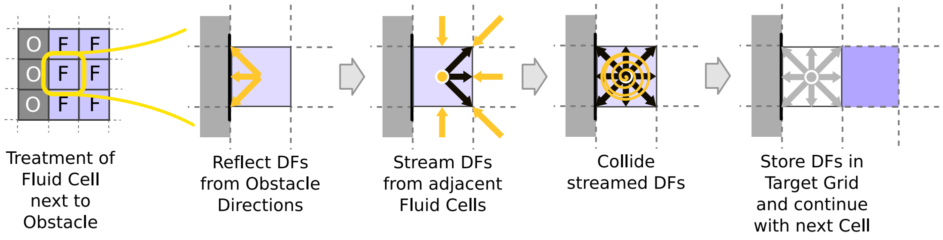
O
F
F
O
F
F
O
F
F
Treatment of Fluid Cell next to Obstacle
Reflect DFs from Obstacle Directions
Stream DFs from adjacent Fluid Cells
Collide streamed DFs
Store DFs in Target Grid and continue with next Cell

Figure 2: Overview of the LBM steps for a single cell near a no-slip obstacle.

## 4 Moving Obstacles

If the obstacle is moving, the momentum of the movement has to be transferred to the fluid. Here we use the model described in [17]. While the basic no-slip handling from Eq. (4) remains the same, an additional forcing term is added during streaming:

$$f_i(\mathbf{x}, t+\Delta t)' = f_{\tilde{i}}(\mathbf{x}, t) + 2\, w_i\, \rho_f\, 3\, \mathbf{e}_i \cdot \mathbf{u}_o, \quad (6)$$

where $\rho_f$ is the fluid density and $\mathbf{u}_o$ the obstacle velocity at the obstacle boundary. The fluid density can be approximated by the initial density, hence $\rho_f = 1$.

For free-slip boundary conditions, the acceleration should only occur in the normal direction of the obstacle surface. The velocity to be used in Eq. (6) is thus projected onto the surface normal. The forcing term is now only added to those DFs where the reflection is reduced to Eq. (4). This is due to the fact that tangential to the free-slip surface there is no acceleration of the fluid due to the surface smoothness. Thus the fluid adjacent to a free-slip obstacle only needs to be accelerated along the normal direction. To still keep the flux balance for free-slip boundaries, the acceleration term of Eq. (6) needs to be multiplied by 2. For free- and part-slip boundary conditions Eq. (6) is thus changed to

$$\begin{aligned} f_i(\mathbf{x}, t+\Delta t)' = w_p \Big( f_{\tilde{i}}(\mathbf{x}, t) + \\ 2\, w_i\, \rho_f\, 3\, \mathbf{e}_i \cdot \mathbf{u}_o \Big) + (1-w_p) \Big( f_r(\mathbf{x}, t) + \\ w_r(\mathbf{x}, t)\, 2\, w_i\, \rho_f\, 3\, \mathbf{e}_i \cdot \big[ (\mathbf{n}_o \cdot \mathbf{u}_o) \mathbf{n}_o \big] \Big), \quad (7) \end{aligned}$$

where $\mathbf{n}_o$ is the obstacle normal at $\mathbf{x}$. $w_r$ is an indicator function that has a value of one if the streaming of $f_r$ reduces to a no-slip reflection, and zero otherwise. The forcing term is thus only applied during the stream step for DFs such as the center arrow of the source cell in Fig. 3 near free-slip boundaries.

While this deals with the obstacle to fluid momentum transfer, the force acting upon the obstacle due to the fluid pressure and movement can be calculated as

$$\begin{aligned} F_o = \frac{\Delta x}{\Delta t} \sum_{\forall \mathbf{x}_b} \sum_{i=1}^{19} \mathbf{e}_i w_o(\mathbf{x}_b + \mathbf{e}_i \Delta t) \\ \Big( f_{\tilde{i}}(\mathbf{x}_b + \mathbf{e}_i \Delta t, t) + \\ f_i(\mathbf{x}_b + \mathbf{e}_i \Delta t, t+\Delta t) \Big), \quad (8) \end{aligned}$$

where $w_o(\mathbf{x})$ is an indicator function that is equal to one when the cell at $\mathbf{x}$ is a fluid cell, and zero otherwise. Here the first sum traverses all boundary cells with position $\mathbf{x}_b$ of the obstacle. Eq. (8) thus approximates the surface integral of the shear stresses and pressure forces along the obstacle surface. Hence, with Eq. (6) and Eq. (7) the fluid to obstacle coupling is computed, while the combination with Eq. (8) enables full two-way coupled fluid simulations.

## 5 Lattice Initialization

Triangular meshes were used to describe the obstacle regions. For the simulation, the movement and position information has to be transferred from the triangle mesh to the LBM grid. This is done by creating point samples from the triangle mesh, that ensure a closed layer for the obstacle surface and can furthermore be used to generate velocity information for the boundary conditions.

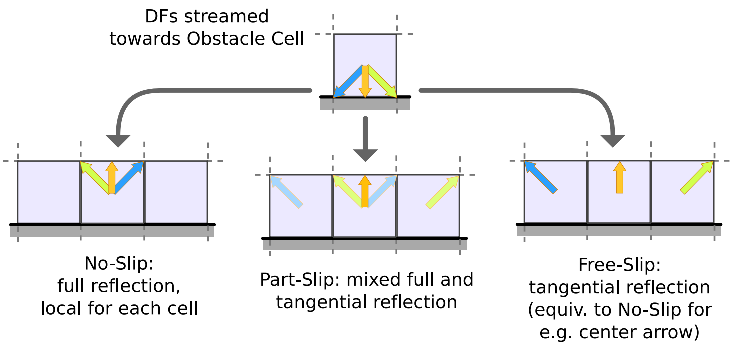
DFs streamed towards Obstacle Cell
No-Slip: full reflection, local for each cell
Part-Slip: mixed full and tangential reflection
Free-Slip: tangential reflection (equiv. to No-Slip for e.g. center arrow)

Figure 3: The different types of boundary conditions for LBM obstacles.

## 5.1 Point Samples

The point samples on the mesh surface are not required to be regularly spaced, but have to ensure that a closed layer of obstacle cells is created for a given LBM grid resolution. No fluid element is allowed to move from one side of a thin obstacle to the other. Thus, no two fluid cells on opposing sides of an obstacle should be connected by a lattice velocity vector. Depending on the size of a grid cell $\Delta x$ a feature size $s_f = \Delta x/2$ is used in the following. Given a triangle with points $\mathbf{p}_1$,$\mathbf{p}_2$,$\mathbf{p}_3$ and normal $\mathbf{n}$, the number of divisions along the sides are computed as

$$s_u = \text{floor}(\frac{|\mathbf{p}_2 - \mathbf{p}_1|}{s_f}) \ , \ s_v = \text{floor}(\frac{|\mathbf{p}_3 - \mathbf{p}_1|}{s_f}) \ . \qquad (9)$$

Then the points $\mathbf{o}$ on the surface can be computed using barycentric coordinates:

$$\begin{aligned} \mathbf{o}_{u,v,1}, \ \mathbf{o}_{u,v,2} &= \\ \left(1 - \frac{u+1/4}{s_u} - \frac{v+1/4}{s_v}\right)&\mathbf{p}_1 + \\ \frac{u+1/4}{s_u}\mathbf{p}_2 + \frac{v+1/4}{s_v}&\mathbf{p}_3 \pm \frac{1}{4}s_f\mathbf{n} \ , \\ \text{with } 0 \le u \le s_u, \ 0 \le v \le s_v, \ v+u \le 1 \ . & \qquad (10) \end{aligned}$$

Note that Eq. (10) creates two points, each of which are offset by the triangle normal and feature size. Eq. (9) guarantees, that the points have a sufficiently small spacing in a plane of the grid, while the normal offset of Eq. (10) ensures that the obstacle layer has a thickness of one to two cells.

To ensure that a minimum of points is generated, the points $\mathbf{p}_1$,$\mathbf{p}_2$ and $\mathbf{p}_3$ of the triangle are permuted to ensure that the values $s_u$ and $s_v$ of Eq. (9) are minimal. Hence, the two shortest sides of the triangle are used for point generation.

## 5.2 Grid Initialization

The whole collection of points $\mathcal{P}$ for an obstacle object thus contains the points generated by Eq. (10) for all triangles, and the original vertices of the mesh. The vertices are also offset by $\pm 1/4 s_f \mathbf{n}$, and are necessary as $s_u$ and $s_v$ can be zero. This would mean that no points are generated in the plane of the triangle, but as the triangle is smaller than a single grid cell, its vertices suffice to initialize the LBM grid. This, however, also means that if the mesh is finely triangulated, not all of the vertices would be necessary for initialization. In this case an external program or suitable software library could be used to simplify the mesh, possibly according to the feature size. In general, the method described above assumes that the feature size if usually significantly smaller than the average triangle size, which is normally the case for detailed fluid animations.

For animated meshes it is required that the animation is described by movements of the vertices, thus the triangle structure itself doesn't change. Before each time step of the LBM, the point set $\mathcal{P}(t)$ is generated for the current time step. Furthermore either the point set of the last time step $\mathcal{P}(t - \Delta t)$ can be used, or generated anew. Now a loop over all points $\mathbf{p}(t)_i$ in $\mathcal{P}(t)$ yields the grid positions of obstacle cells to be initialized for the current time step. The obstacle velocity $\mathbf{u}$ for Eq. (4) can be computed

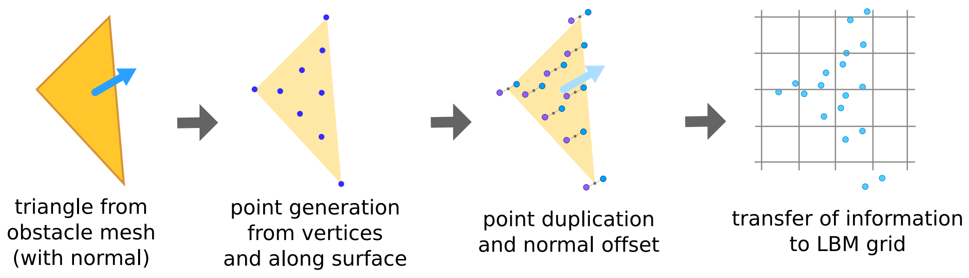
triangle from obstacle mesh (with normal)
point generation from vertices and along surface
point duplication and normal offset
transfer of information to LBM grid

Figure 4: Overview of the obstacle initialization for a single triangle of the mesh.

with

$$\mathbf{u} = \Big(\mathbf{p}(t)_i - \mathbf{p}(t-\Delta t)_i\Big)/\Delta x \,. \tag{11}$$

For optimization purposes, the acceleration term of Eq. (4) can be precomputed and stored in the LBM grid, as obstacle cells do not require DFs to be stored. As often more than a single point of $\mathcal{P}$ maps to one grid cell, these multiple values could be averaged with appropriate weights. However, initializing a cell with the first point that maps to it also yields good results – this scheme will also be used in the following.

During this pass over the obstacle points the maximum velocity on the obstacle surface is easily computed, and can be used to adapt the time step size of the simulation correspondingly. In a second pass of the point set $\mathcal{P}(t-\Delta t)$ old obstacle cells have to be removed from the grid. These can be initialized by empty cells, or, if a fluid cell is in the neighborhood with an interface cell. The latter case is necessary as fluid cells are not allowed to be in the neighborhood of an empty cell, so alternatively the fluid cell could be converted to an interface cell.

### 5.3 Mass Conservation

Note that the boundary conditions so far do not conserve mass – due to the approximation of the obstacle boundary with cells, the moving object can cover a different number of cells during its movement. For flows without a free surface, hence with completely submerged objects, this is usually unproblematic. However, with free surface flows, this can lead to significant errors in the mass conservation. To alleviate this problem, the change of mass in the system due a moving no-slip obstacle can be directly computed as

$$\Delta M_o = M_{\text{add}} - M_{\text{sub}} + \sum_{\forall \mathbf{x}_b} 2\, w_i\, \rho_f\, 3\, \mathbf{e}_i \cdot \mathbf{u}_o \,. \tag{12}$$

Here $M_{\text{add}}$ denotes the gained mass from interface cells that are newly created in the current LBM step due to the object movement (as explained in the previous paragraph). On the other hand, $M_{\text{sub}}$ is the mass that was lost from all fluid cells that were reinitialized as new boundary cells during the last step. As a third component, the sum of all acceleration terms from Eq. (4). Note that Eq. (12) assumes no-slip boundary conditions, and thus has to be changed as explained above for free- or part-slip obstacles. By accumulating $M_o(t+\Delta t) = M_o(t) + \Delta M_o$ for each moving obstacle in the simulation, the change of mass due to this object can be calculated. A parameter that can be freely chosen is the fill fraction $\epsilon_n$ for the newly created interface cells from the pass over the last object position with $\mathcal{P}(t-\Delta t)$. We now set $\epsilon_n = 0$ if $M_o(t) \geq 0$, and likewise $\epsilon_n = 1.5$ when $M_o(t) < 0$. As the front side of the object removes mass from the system, the back side can thus be used to control the mass loss. Note that if further accuracy of the mass correction is necessary, the initialization of all $\epsilon_n$ could be normalized by the number of new interface cells. This, however, would require a second pass over these cells.

## 6 Surface Generation

While the fluid can usually be reconstructed directly from the fluid fraction values of the free surface tracking, the surfaces adjacent to obstacles require additional treatments. The fluid surface itself is

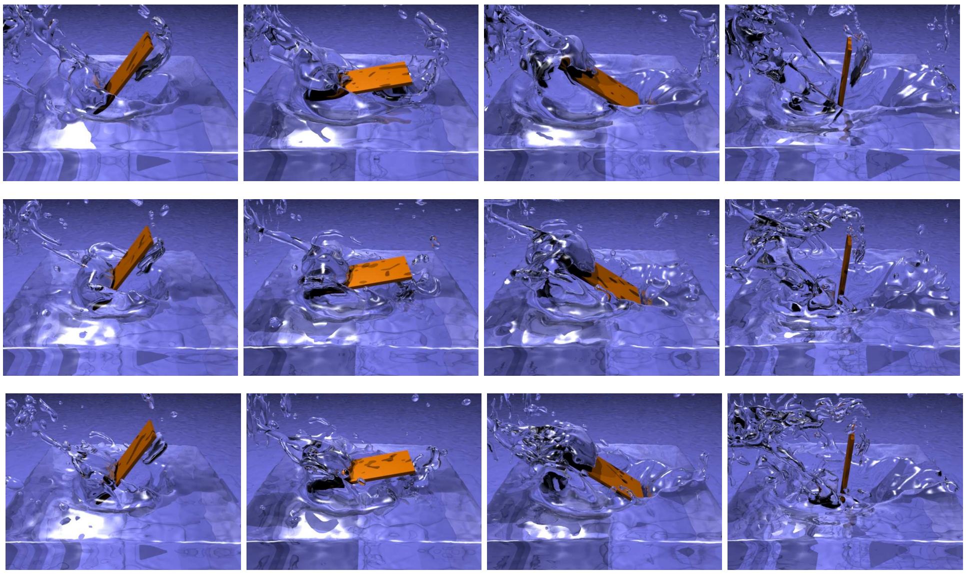
Figure 6: Test case for moving rigid bodies: a rotating box is lowered into a basin of fluid. The upper row of pictures uses no-slip boundary conditions for the box, the middle one part-slip and the lowest one free-slip boundary conditions. An increased amount of splashes is visible in the lower rows of pictures.

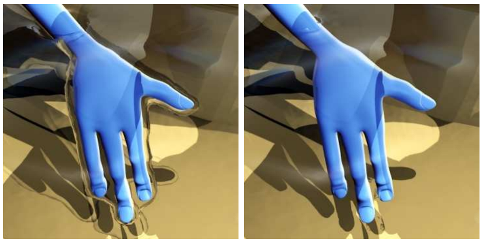
Figure 5: Comparison of the two surface generation approaches. To the left, surfaces are generated for domain sides and around the submerged obstacle. To the right, these surfaces are removed with the algorithm described in Section 6.

given by the fill fraction isolevel $\lambda$ of $\lambda = 1/2$, and can be triangulated using e.g. the marching cubes algorithm [18]. The obstacle initialization as described above will result in a fluid surface around the obstacle surface without touching it. Thus a separation of fluid and obstacle surfaces is visible, in contrast to a real fluid, that would connect with the immersed object at the interface and not have a visible surface in the immersed regions.

This appearance can be gained by extrapolating the fluid fraction values into the obstacle layer. We thus set fluid fraction values for obstacle cells $\epsilon_b$ in the following way: $\epsilon_b(\mathbf{x}) = 1$ is used for all obstacle cells at $\mathbf{x}$ with fluid, but without interface cell neighbors. If an obstacle cell has both fluid and interface neighbors, its fluid fraction value is set to an average fluid fraction with

$$\epsilon_b(\mathbf{x}) = \frac{\sum_{i=1}^{19} w_b(\mathbf{x}+\mathbf{e}_i)\epsilon_b(\mathbf{x}+\mathbf{e}_i)}{\sum_{i=1}^{19} w_b(\mathbf{x}+\mathbf{e}_i)} , \qquad (13)$$

where $w_b(\mathbf{x})$ is an indicator function that is one if the cell at $\mathbf{x}$ is either a fluid, interface or empty cell, and zero otherwise. In a second pass, all obstacle cells that have not been modified, and thus are not in the neighborhood of the fluid, are set to $\epsilon_b = 0.99\lambda$. This is necessary as the initialization from Section 5 can result in two layers of obstacle cells, in which case the surface should be moved further inwards to ensure that is inside of the original mesh.

An example of this modified surface generation can be seen in Fig. 5. To the left of Fig. 5 an image of the uncorrected fluid surface is shown, while the right side shows the actual moved fluid surface. Note that for transparent objects it will moreover be

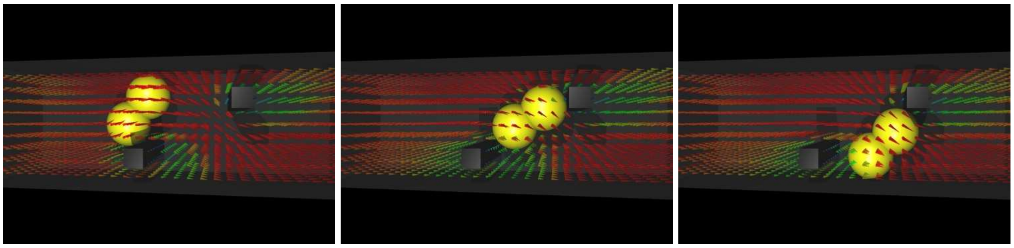

Figure 7: Two-way coupling of the boundary conditions. An object consisting of two connected sphere is bouncing through a channel filled with fluid. The arrows indicate the fluid velocity.

necessary to clip the fluid surface with the obstacle mesh to hide the inner parts while retaining an exact connection of the fluid surface with the obstacle.

## 7 Results

In the following three different simulations for the algorithm described above will be presented. The first, and relatively simple, test case with a rigid moving object can be seen in Fig. 6. Here a simplified "mixer" is modelled – a box is lowered into the fluid while rotating along the y-axis. This simulation was performed with a resolution of $128^3$ and a viscosity of water. The following simulation times are given for a standard Pentium4 with 3.0GHz. For the mixer example the simulation took $27$ seconds per frame on average. Each row of pictures uses different boundary conditions for the obstacle. From top to bottom the boundary conditions are: no-slip, part-slip with $w_p = 0.002$ and free-slip. As the effect of $w_p$ is highly non-linear, the weight has to be small in order to show a noticeable effect of the free-slip boundary conditions. It can be seen that the fluid is correctly accelerated and pushed away by the rectangular shape. The free-slip simulation exhibits a larger amount of splashes, as there is less slowdown in tangential direction. As expected, the part-slip boundary conditions lie in between the two other cases.

A different setup to test the two-way coupling of the moving boundary conditions is shown in Fig. 7. Here an moving object consisting of two connected sphere is pushed through a channel with fluid and two fixed rectangular obstacles. The object is accelerated up to the fluid velocity, then bounces through the two fixed obstacles. In this case a simulation resolution of $60 \times 30 \times 30$ was used, with no-slip boundary conditions for all obstacles. Due to the small size, the total simulation only took ca. $15$ seconds.

The boundary conditions for deforming meshes are demonstrated in Fig. 8, where an animated character is interacting with the fluid. The simulation with a resolution of $166 \times 166 \times 200$ took on average $244$ seconds per frame, and the obstacle initialization with more than $125k$ points for the $5038$ triangle mesh ca. $7\%$ of the time for each LBM step.

## 8 Conclusions

We have presented an algorithm to efficiently handle arbitrary triangle meshes as moving obstacles of varying surface smoothness for LBM free surface simulations. Our approach guarantees a closed layer of obstacle cells and improves mass conservation. We have moreover shown how to remove the interface between the fluid and an obstacle for the triangulation. As demonstrated with test cases of varying complexity, the algorithm is physically accurate and does not significantly increase computation times.

In the future, the method could be extended by e.g. coupling it to simulations for rigid bodies, deforming elastic objects or even fracturing. Although this might not be crucial for most physically based animations, it would also be interesting to develop higher order methods for free-slip boundary conditions and efficient boundary conditions that guarantee mass conservation.

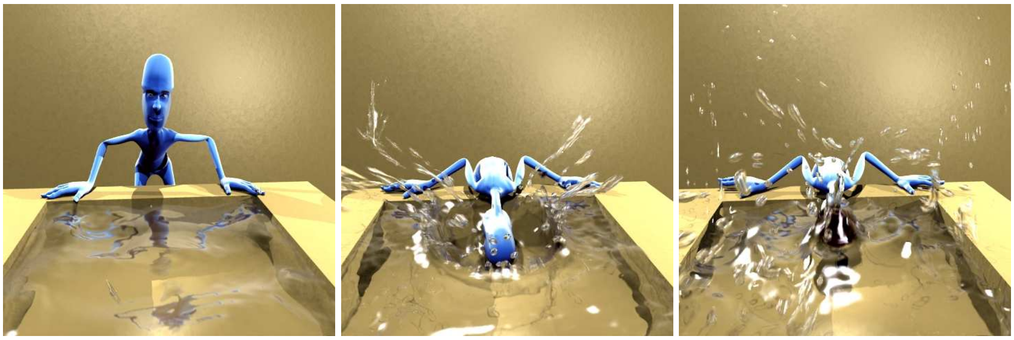

Figure 8: Example of a deforming mesh from an animated character interacting with the fluid.

## References

[1] Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows. *J. of Comp. Phys.*, 152:457–492, 1999.

[2] Jos Stam. Stable Fluids. *Proc. of ACM SIGGRAPH*, pages 121–128, 1999.

[3] Mark Carlson, Peter John Mucha, and Greg Turk. Rigid Fluid: Animating the Interplay Between Rigid Bodies and Fluid. *ACM Trans. Graph.*, 23(3):377–384, 2004.

[4] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling Water and Smoke to Thin Deformable and Rigid Shells. *ACM Trans. Graph.*, 24(3):973–981, 2005.

[5] Uriel Frisch, Dominique d'Humières, Brosl Hasslacher, Pierre Lallemand, Yves Pomeau, and Jean-Pierre Rivert. Lattice Gas Hydrodynamics in Two and Three Dimensions. *Complex Systems*, 1:649–707, 1987.

[6] X. He and L.-S. Luo. A Priori Derivation of Lattice Boltzmann Equation. *Phys. Rev. E*, 55:R6333–R6336, 1997.

[7] Xiaoming Wei, Ye Zhao, Zhe Fan, Wei Li, Suzanne Yoakum-Stover, and Arie Kaufman. Natural phenomena: Blowing in the wind. *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 75–85, July 2003.

[8] C. Körner, T. Pohl, U. Rüde, N. Thürey, and T. Zeiser. Parallel Lattice Boltzmann Methods for CFD Applications. In A.M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers*, volume 51 of *LNCSE*, pages 439–465. Springer, 2005.

[9] N. Thürey and U. Rüde. Free Surface Lattice-Boltzmann fluid simulations with and without level sets. *Proc. of Vision, Modelling, and Visualization VMV*, pages 199–208, 2004.

[10] J. J. Monaghan. Smoothed particle hydrodynamics. *Rep. Prog. Phys.*, 68:1703–1758, 2005.

[11] Matthias Müller, Simon Schirm, Matthias Teschner, Bruno Heidelberger, and Markus Gross. Interaction of Fluids with Deformable Solids. *Journal of Computer Animation and Virtual Worlds (CAVW)*, 15(3-4):159–171, July 2004.

[12] X. He and L.-S. Luo. Lattice Boltzmann model for the incompressible Navier-Stokes equations. *J. of Stat. Phys.*, 88:927–944, 1997.

[13] Dazhi Yu, Renwei Mei, Li-Shi Luo, and Wei Shyy. Viscous flow computations with the method of lattice Boltzmann equation. *Progress in Aerospace Sciences 39*, 5, 2003.

[14] N. Thürey and C. Körner and U. Rüde. Interactive Free Surface Fluids with the Lattice Boltzmann Method, Technical Report 05-4. Technical report, Department of Computer Science 10 System Simulation, 2005.

[15] N. Thürey and T. Pohl and U. Rüde and M. Oechsner and C. Körner. Optimization and Stabilization of LBM Free Surface Flow Simulations using Adaptive Parameterization. *Computers and Fluids*, 35 [8-9]:934–939, September-November 2006.

[16] M'hamed Bouzidi, Dominique d'Humières, Pierre Lallemand, and Li-Shi Luo. Lattice Boltzmann equation on a two-dimensional rectangular grid. *J. Comp. Phys.*, 172(2):704–717, 2001.

[17] A. J. C. Ladd. Numerical simulation of particular suspensions via a discretized Boltzmann equation. Part 2. Numerical results. *J. Fluid Mech.*, 39:271–311, 1994.

[18] William Lorensen and Harvey Cline. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. In *Computer Graphics Vol. 21, No. 4*, pages 163–169, August 1987.

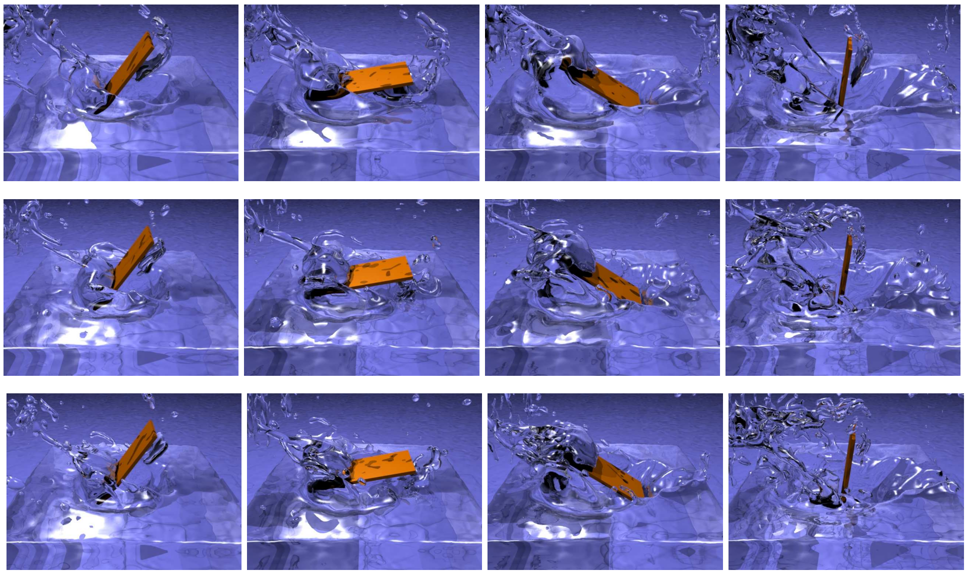

Figure 1: Test case for moving rigid bodies: a rotating box is lowered into a basin of fluid. The upper row of pictures uses no-slip boundary conditions for the box, the middle one part-slip and the lowest one free-slip boundary conditions. An increased amount of splashes is visible in the lower rows of pictures.

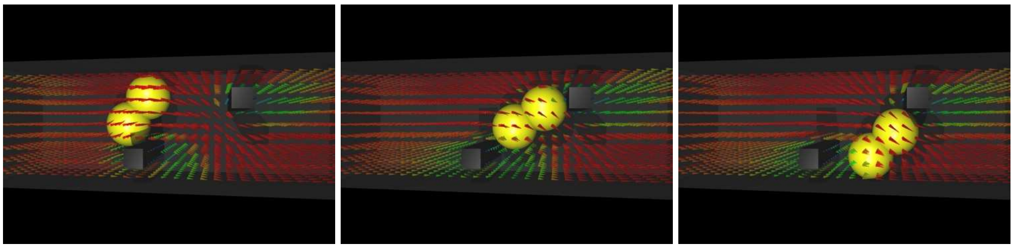

Figure 2: Two-way coupling of the boundary conditions. An object consisting of two connected sphere is bouncing through a channel filled with fluid. The arrows indicate the fluid velocity.

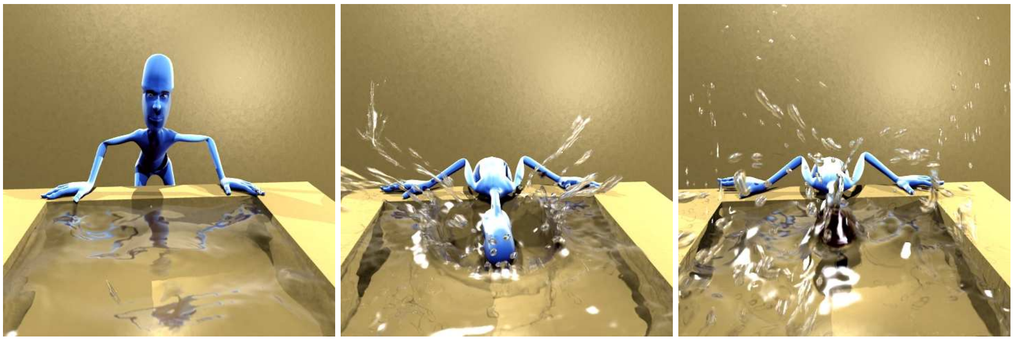

Figure 3: Example of a deforming mesh from an animated character interacting with the fluid.