Simulating Fluid-Solid Interaction

Olivier Génevaux Arash Habibi Jean-Michel Dischler

LSIIT UMR CNRS-ULP 7005
Boulevard Sébastien Brant, F67400 Illkirch, France
{genevaux, habibi, dischler}@dpt-info.u-strasbg.fr

Abstract

Though realistic eulerian fluid simulation systems now provide believable movements, straightforward renderable surface representation, and affordable computation costs, they are still unable to deal with non-static objects in a realistic manner. Namely, objects can not have an influence on the fluid and be simultaneously affected by the fluid's motion. In this paper, a simulation scheme for fluids allowing automatic generation of physically plausible motions alongside realistic interactions with solids is proposed. The method relies mainly on the definition of a coupling force between the solids and the fluid, thus bridging the gap between commonly used eulerian fluid animation models and lagrangian solid ones. This new method thus improves existing fluid simulations, making them capable of generating new kinds of motions, such as a floating ball displaced by the wave created thanks to its own splash into the water.

Key words: Fluid simulation, interactions, Navier-Stokes equations, spring-masses simulation.

1 Introduction

Since realism is one of the major goals of computer graphics, numerous works attempt to provide so-called "realistic" methods to allow an easy creation of digital equivalents for natural phenomena. Within these, the animation of fluids, and especially liquids, is a particularly difficult task due to the underlying laws that govern fluids' motions. Indeed, these laws, known as the Navier-Stokes, equations are highly unstable partial differential equations that are quite difficult to solve in an efficient way, and even difficult to solve at all.

Multiple types of methods were proposed to simulate fluids. The first kind of method, which consists of what can be called "empirical hand-crafted models", does not involve the Navier-Stokes equations at all, or even physical correctness, but offloads the task of creating a convincing velocity field to an animator, using velocity primitives [4]. Such an approach may be complemented using a small scale turbulent velocity field that is added to the large scale hand-designed flow to increase details and give the fluid a more turbulent, thus better looking, behavior [17, 18].

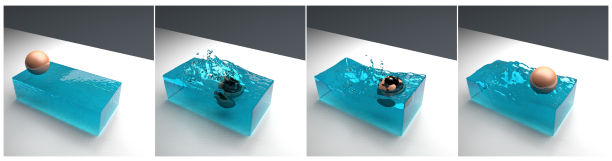

Figure 1: A ball falling inside a tank. — The ball is pushed upward due to the pressure wave it created.

The second kind of method, which encompasses particle based simulations, does not make use of the Navier-Stokes equations either, but are nevertheless based on physical considerations. Indeed, they are able to exhibit realistic fluid behaviors without resorting to an explicit description of motions. These motions naturally spring from the local forces in action between the particles [13, 2, 9]. It should be noted that these models are already able to deal with convincing interactions between solids and liquids. This arises from the fact that both entities are simulated using the same lagrangian representation [10]. However this kind of simulation comes with a high computation cost stemming from the huge number of forces that must be computed between the many particles needed to obtain realistically looking motion.

To avoid such a plethora of particles, physical macroscopic modeling may be used. This is what is employed in a number of different works, from heavily simplified simulation models to thorough ones. These simplifications can be limitations to the simulable behaviors, such as non turbulent fluids [20, 19], or fluids that can be represented by a heightfield [11, 15]. In any case, it should be noted that physical considerations are taken into account, either explicitly by the solver or implicitly by the underlying simulated equations.

The last class of simulators extends the former one. Indeed, the Navier-Stokes based simulators are undoubtedly physically based, but what makes them specific is that they use a comprehensive fluid model to its maximum extent: they don't put any restrictions on the prop-

erties of the simulated fluid, nor on the motions that can be simulated starting from these equations. The only limitation arises from the allowed computational cost, which can be quite substantial when large scale but precise simulations are required. Despite much less demanding than lagrangian systems, these simulators can thus still be costly. The cost, along with the problem of liquid tracking and surface determination, explains why these simulators were first developed to run on two dimensional domains [1], before being expanded to full three dimensional domains [8]. Another interesting point of these simulation schemes is that movements can be easily controlled by animators without ruining the physical believability of the whole animation [6].

Further development led to the solution of the two aforementioned problems related to these methods: computation speed was improved using a more stable evolution scheme [16], and a visually appealling surface representation was devised to allow realistic visualization of liquids [7, 5]. This last point is not to be underestimated due to the fact that current liquid simulations often rely on marker particles to discriminate between the fluid and its surrounding environment, thus not readily exhibiting a clear and good looking surface to be visualized.

However, if these schemes are able to handle a fluid inside its static environment fairly well, they don't provide any means to deal with true reciprocal interactions between fluid and arbitrary solids, except for one work applied to the explosion simulations [21], and another work restricted to heightfield represented water [15]. The first work is not directly relevant to liquid simulation though, because it does not deal with free surface liquid while the heightfield nature of the second work is highly limitating its use.

As a consequence, no animation system based on the Navier-Stokes equations is able to automatically deal with a simple ball that splashes into water and then oscillates due to the waves it just created without lots of animator's work. This extraneous work therefore puts a damper on the usage of liquids in computer graphics productions.

In this paper, a new method allowing to link a commonly used eulerian fluid simulation to a lagrangian solid simulation is proposed. While not new to fluid mechanics [3], such a method is new to the field of computer graphics. This way, arbitrary objects can be realistically incorporated into fluid simulations without resorting to any external intervention, as shown in Figure 1. Resulting motions of fluid and solids automatically appear physically correct since the strong coupling between the two different entities are handled in a coherent way. This is a definite improvement on previous work, where only unidirectional actions between fluids and solids were considered, so solids either responded to fluid's motion but without any action on it, or vice-versa.

The new method presented in this paper relies essentially on the definition of a new force, computed by taking into account both of the fluid and the solids, to capture the interaction. This new force is then simultaneously introduced in both models, thus guaranteeing coherent behaviors. Though this method may appear simple, it was found to be computationally effective while producing satisfactory results, as can be seen in the included pictures and videos.

The remaining of the paper is organized as follows: section 2 describes both the separates fluids' simulation model and the solids' one. Section 3 details the coupling scheme between these two models. Section 4 presents limitations of the method. Section 5 provides results while section 6 analyses the proposed method as well as provides some hints about improvements that could be made.

2 Underlying simulations

Before describing how the interaction between liquids and solids is handled, a description of the two distinct elementary simulation models is given.

2.1 Liquid simulation

The liquid related part of the simulator is based on the Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where a fluid of density ρ and viscosity ν is described: $\mathbf{u}$ stands for the velocity field, p for the pressure field, and $\mathbf{F}$ for the external forces, such as gravity. Equation 1 expresses momentum conservation, while equation 2 states that the liquid is incompressible, which means that volume must be conserved over time.

Since the simulator must be able to cope with a liquid that does not fill all the simulation space, a way to discrimate between the liquid and its surrounding empty environment must be devised. Moreover, full three dimensional motions must be achievable, so heightfield-like solutions, where waves are not able to overturn, are not relevant. The way to solve these two problems is to resort to *Markers And Cells* simulation. In this kind of simulations, velocity and pressure are evaluated on cells of a rectilinear static grid that defines the simulation domain, meanwhile liquid is tracked as a cloud of markers moving inside this grid, as shown in Figure 2. These massless

markers are passively advected according to the underlying velocity field, which is updated over time using the Navier-Stokes equations. Whether a given cell of the grid is considered full or empty of fluid by the simulator depends on the presence of at least one marker inside the considered cell.

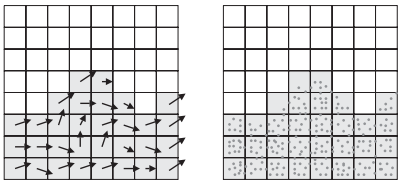

Figure 2: Fluid Simulation. — Realized using a grid carryind fluid velocity and many markers to delineate the fluid presence.

It should be noted that the grid does not sample both the pressure and the velocity fields at the same location but instead makes use of staggered sampling. The different components of the velocity field are not sampled on the same position either. Focusing on the velocity sample taken on a cell, each component is sampled on the center of the orthogonal faces to the considered component orientation, whereas pressure is sampled on the center of the cell. Such a scheme is chosen due to better stability properties compared with a scheme where the samples are taken from the same location. A more detailed description of the simulation grid can be found in articles that introduced the method in computer graphics [8].

Fluid evolution is achieved using the method proposed in [7]. The different parts of equation 1 are simulated using different strategies depending on their nature. The convection term $-(\mathbf{u} \cdot \nabla)\mathbf{u}$ is handled by a semi-lagrangian integration process, the viscous term $\nu\nabla^2\mathbf{u}$ is solved using direct differentiation while the external force term $\mathbf{F}$ contribution is based on simple Euler integration. Mass conservation $\nabla \cdot \mathbf{u} = 0$ is enforced by means of a projection step inside the target divergence free space. This projection is computed by solving an adequately built equations system using a conjugate gradient solver. This last step takes care of the pressure related term $-1/\rho\ \nabla p$ of equation 1 too.

Apart from liquid simulation, liquid visualization must also be addressed. In order to get a surface, a marching cube algorithm [12] is used. The underlying data field of the marching cube is built by counting the fluid markers inside each cell of a fine grid and smoothing these values. While not creating the best looking surfaces, this strategy was chosen among others because it is quite fast and easy to implement while it gives satisfactory smooth surfaces. However, a major drawback of such a surface extraction system is that artificial fluid dissipation is introduced by the smoothing step. Nonetheless attention must be drawn to the fact that this dissipation is only visual, and no fluid at all is lost at the simulation level using this technique. This dissipation can be seen on Figure 2 where some of the droplets visually disappear during the simulation.

2.2 Solid simulation

Although fluid is simulated through a rather complex process, solids are much easier to handle. While solids can be represented as rigid bodies delineated by a surface made of triangles, they can also be represented as a set of linked point masses [14]. This latter model, for which a simple example is given in Figure 3, is especially well suited to animation. It is used in the presented method because of its flexibility to handle both rigid and non-rigid bodies, its easy manipulation, and the way it fits well in the interacting composite model.

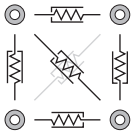

Figure 3: Solid Simulation. — Example of a 2D square: 4 nodes are linked using viscous springs.

A spring-mass solid, as one can infer from its name, is made of a set of masses linked by interactions. These cohesion interactions are chosen to be not only elastic but visco-elastic, which means that the intensity f of the force between two given particles i and j is not only dependant on their distance but on their closing speed too:

$$f = -k_{coh} \cdot (|\mathbf{x}_i - \mathbf{x}_j| - d_{coh}) \\ -z_{coh} \cdot \frac{d}{dt}|\mathbf{x}_i - \mathbf{x}_j|.$$

The force $\mathbf{F}$ is obviously aligned with the direction implied by the positions of the two particles:

$$\mathbf{F} = f \cdot \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|}.$$

To evolve the solid over time, the position of each mass is updated independently of the others according to Newton's second law taking account of all the applied forces:

$$\mathbf{x} = \frac{1}{m} \iint \sum_i \mathbf{F}_i \, dt^2.$$

The $\mathbf{F}_i$ forces encompass all the applied forces to the mass, coming either from the internal or external interactions, such as gravity or obstacle collision forces. The

aforementioned flexibility of spring-mass solids arises from this separate time evolution process of every mass: since their relative position is allowed to change related to the stiffness of interactions, all kind of solids can be built, from extremely plastic to the most sturdy ones.

3 Interfacing the two models

To allow a true reciprocal interaction between the two different eulerian and lagrangian models prevously described in sections 2.1 and 2.2, an interface to bind them is needed.

This interface is chosen to stand as a representation of the fluid for the solids, that is, to build a lagrangian description of the eulerian world. In fact, looking at the fluid simulator, this very representation is already available through the use of the massless markers whose role is to delineate the presence of fluid. Thus, the interface is chosen to be built out of these markers. This way, no new entity is introduced inside the model, helping to avoid unnecessary overcomplication of the whole composite simulator.

Indeed, these markers are already able to easily, yet precisely, discriminate where fluid is actually located inside the environment. This is an extremely important fact since the position of the interface is part of the way it affects both models. This position dependance is undoubtly needed due to the free-surface nature of the simulated liquids. Indeed, since solids can be half-submerged, it would be a rough oversimplification to assume that all of their surrounding environment is composed only of fluid. Moreover, the point cloud nature of this interface will allow linkage of the interface with the two different models.

Another interesting property of this interface is that since it is defined as a set of points, it is trivial to define a restricted interface as a subset of all the markers. This property is of the utmost interest since it allows one to easily manipulate an almost steady interface made of a constant number of points even if the number of fluid markers is increased to improve fluid representation. This allows an animator to avoid altering dynamic properties of the interface between a low resolution prototyping animation and a full resolution production one.

The composite simulator is thus formed of three logical entities:

- The eulerian Navier-Stokes based fluid simulator;
- The lagrangian spring-mass solids simulator;
- The interface that must be able to flow data between them, translating these data when they are not readily interpretable by the destination model.

The final coupling is asymmetric at a coarse level, as can be seen on the synthetic decription depicted in Figure 4 and is comprised of two parts. On the one hand, the connection between the interface and the solid model is symetric through the definition of a force equally reintroduced in the two entities. On the other hand, the binding between the fluid model and the interface is an assymetric one, where velocity information are transferred from liquid toward the interface while forces information flow the reverse way. This scheme has been chosen instead of the converse one, where velocity would flow from solids to liquid and force from liquid to solids, since it allows the aforementioned interface definition which is experimentally found to be effective and efficient.

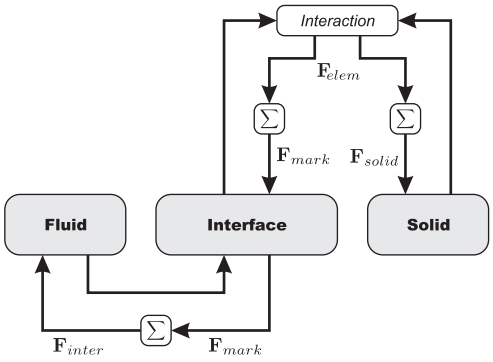

Figure 4: Overview of the coupling scheme.

3.1 Interface / Solids relationship

The relation between the interface and the solids is handled in an efficient way by the use of a force exerted between each marker i of the interface and each nodal mass j of the solids. This elementary force $f_{elem_{ij}}$ is viscoelastic, such as the internal cohesion force of solids, but with a limited radius of influence:

$$\begin{aligned}\bar{f}_{elem_{ij}} &= -k_{int} \cdot (|\mathbf{x}_i - \mathbf{x}_j| - d_{int}) \\ &\quad -z_{int} \cdot \frac{d}{dt}|\mathbf{x}_i - \mathbf{x}_j|,\end{aligned}$$

$$f_{elem_{ij}} = \begin{cases} \bar{f}_{elem_{ij}} & \text{if} \quad |\mathbf{x}_i - \mathbf{x}_j| \leq R \\ 0 & \text{if} \quad |\mathbf{x}_i - \mathbf{x}_j| > R \end{cases} .$$

This way, solids are naturally affected by the nearby interface, which is itself determined by liquid movements. The limited interaction distance between solid and interface guarantees that objects are only influenced by closely located liquid motions, and not by far, thus physically irrelevant, liquid movements.

The radius R is closely related to the way solids are modeled: its size must be big enough to avoid holes in the solids while correctly approximating their surfaces.

No general guideline can be given to setup the stiffness k_{int} and the viscosity z_{int}, except that k_{int} must be stiff enough to repel water markers from the interior of the solid. Unless interior is preserved from fluid, objects will be filled with fluid, preventing a buoyancy force to develop. This buoyancy force automatically stems from pressure difference in the fluid around the object due to gravity.

Given this force definition, the complete interaction force can be computed for each node belonging to a solid $\mathbf{F}_{solid}$, as well as for all of the interface markers $\mathbf{F}_{mark}$, as the sum of the forces in which the entity is involved:

$$\mathbf{F}_{solid_i} = \sum_j \mathbf{F}_{elem_{ij}}, \quad \mathbf{F}_{mark_j} = \sum_i \mathbf{F}_{elem_{ij}}.$$

This resulting force is directly taken into account by the solids as one of the many gathered forces, such as gravity or internal cohesion. No further consideration is necessary since this raw expression of the new force perfectly fits the evolution scheme of solids.

However, if a lagrangian simulator can make direct use of this force, an eulerian simulator can not. Although not of a straightforward use, this force should not be wasted since it carries all the information to update the eulerian simulator to make it react to the solids. The interaction force is thus stored on the interface marker basis to be reintroduced in the eulerian simulator after suitable transformation.

3.2 Interface / Fluid relationship

On the other side of the interface, half of the relation between the interface and the fluid is already built. Indeed, since the interface is formed out of the fluid advected markers, the interface is already under the influence of fluid in a physically realistic way, and fluid velocity is able to affect solids through the interface.

The last link that must be devised to complete the interaction scheme is the link between the interface and the fluid, to keep the solid to liquid force flowing from interface to the fluid model. Recalling the last term of equation 1, the Navier-Stokes equation related to momentum conservation, external forces, such as the interaction one, can be incorporated inside the simulation. However, unlike gravity, the interaction is spatially dependant, which means that last term of the the equation should be updated to $\mathbf{F}(\mathbf{x})$ instead of $\mathbf{F}$, as depicted in Figure 5.

To populate this force field $\mathbf{F}_{inter}(\mathbf{x})$, a simple yet effective method, summarized in Figure 6 is used. Forces stored at interface markers are summed on a per computational grid cell basis, so as to obtain the force $\bar{\mathbf{F}}_{inter}$ globally applied to the fluid located inside each cell C:

$$\bar{\mathbf{F}}_{inter}(C) = \sum_{i \in C} \mathbf{F}_{mark_i}.$$

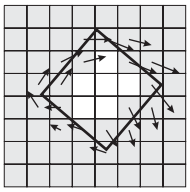

Figure 5: The interaction is defined as a spatially dependant force field for the fluid simulator.

The key point of this method is that partially filled cells do not require special handling with respect to completely submerged ones. Indeed, the relative number of interface markers per cell is automatically able to handle this filling discrepancy.

Once these per-cell forces have been computed, the force field is ready to use. Due to the staggered nature of the grid, the actual force $\mathbf{F}_{inter}$ used during time evolution of the velocity carried by a given face inside the grid is obtained by averaging the two forces stored in the neighboring cells. Using the Kronecker's delta δ_{pq}, for coordinates denoted as $n = 0, 1, 2$, this can be expressed as:

$$\begin{aligned}\mathbf{F}_{inter}(\mathbf{x}_{(i,j,k)+\delta/2}) &= \bar{\mathbf{F}}_{inter}(C_{i,j,k})/2 \odot \delta \\ &+ \bar{\mathbf{F}}_{inter}(C_{(i,j,k)+\delta})/2 \odot \delta\end{aligned}$$

where $\delta = [\delta_{0n}, \delta_{1n}, \delta_{2n}]$ and where $\odot$ stands for component-wise multiplication: $(x_i) \odot (y_i) = (x_i \cdot y_i)$.

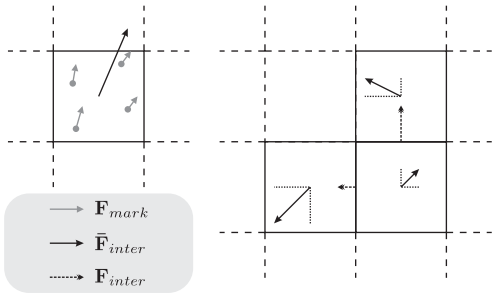

Figure 6: Processing of raw per-marker interaction forces to take account of interaction into eulerian simulator. — Left: Forces of markers are summed on a grid cell basis. Right: Adjacent cell forces are used to compute force required in speed evolution of faces.

4 Limitations of the method

Despite the clear improvement of allowing a simple two way interaction, the method still suffers from some drawbacks.

Indeed, the discretization of the force field is extremely coarse and imprecise. This imprecision can sometimes

lead to "spurious currents" that may appear at the interface between the solids and the fluid. However, it was found that these currents are most of the times unnoticeable and without tangible influence on the liquid surface position. It should be noted though that this problem is intrinsic to the discretization of a force field in eulerian simulation methods and that the Computational Fluid Dynamics community is still trying to address this problem.

Another limitation of the method is its relatively difficult setup. Indeed, multiple interlocked parameters need to be devised, each having dramatic influence on the results.

5 Results

This section presents results obtained from the previously described simulator. Pictures are rendered using the Radiance photo-realistic renderer [1]. The liquid surface is extracted from the simulation using the technique described in section 2.1. All simulations were run on an Athlon XP 1800+ with 512MB RAM. Meaningful parameters of the different examples are gathered in Table 1. The timestep used for all the simulations is $1/240$ second, but an export of the surface is only performed every $1/24$ second.

This small timestep is due to the solids introduced in the simulation. Indeed, they must have small displacements during a timestep compared to the size of a cell of the fluid simulation grid. This is a requirement if one wants to avoid instabilities and to produce meaningfull results.

Figure 7 illustrates buoyancy. On the beginning of the simulation, both the fluid and the ball are completely steady, but buoyancy allows the ball to surface.

Figure 8 shows a ricochet of a ball thrown in a tank.

Figure 9 involves a high speed jet striking a cube. The cube is made of an elastic material that deforms but quickly regains its shape such as rubber. It is modeled as thousand nodes arranged in a regular cubic lattice scheme. When the jet strikes the cube, water is able to temporarily tear the cube, showing that the interaction force can be precisely localized in space, even down to a sub-object level. This can be seen in the close up shots depicted in Figure 10. Meanwhile, the cube is able to deflect the tip of the jet, before being put into movement. It should be noted that the noisy appearance that fluid develops in complex motions areas is not induced by the simulator itself but is rooted in the fact that the fluid surface evolves in a too convoluted mesh to be rendered smoothly.

Figure 11 depicts a U-shaped container, half filled with water, where two different cubes are dropped on top of the water, in the two branches of the U container. The red cube is five times as heavy as the yellow one and nearly three times as rigid. The container has a square cross-section, almost identical to the cross-section of the two cubes. With such a situation, when a cube hits the fluid, the movement is entirely transmitted at the opposite end of the fluid without loss of momentum, like in a hydraulic brake system. The interaction between the cubes and the fluid is clearly visible looking at the lighter yellow cube, on the left of the device. When it first hits the water, it pushes the fluid upward in the opposite part of the container, but when the second heavier cube hits the water a bit later, it is itself propelled upward by the water. The two way interaction is thus clearly demonstrated: solid to fluid forces are seen first and then fluid to solids forces are visible.

[1] http://radsite.lbl.gov/radiance/HOME.html

6 Conclusions and future works

A new method that allows realistic interaction between a free surface fluid and solids has been presented. This method proved to be easy to implement, computationally affordable and efficient to fulfill its role.

This method is believed to provide a definite improvement in the visual realism of animations involving fluid by providing a simple way to add more life in these scenes.

Since the method does not involve deep modifications of classic eulerian fluid simulators, it should be possible to integrate this extension into more sophisticated simulators that give better representation of the fluid's surface to further increase realism of animation without much trouble.

Improvements could also be made by taking steps to insure that simulation grid refinement does not affect dynamic properties of the interaction, in order to allow an efficient transition between prototype and production animations. Moreover, more intuitive parameters would be of great help during the setup phase of the simulation.

Another possible enhancement lies in the solid representation. Indeed, not all objects are easy to represent as a set of nodal masses fitted with spherical interaction fields. It would thus be interesting to extend the definition of the interaction force to an interaction between fluid's markers and triangles, to deal with complex triangulated meshes in an obvious way.

References

[1] Jim X. Chen and Niels Da Vitoria Lobo. Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. *Graphical models and image processing: GMIP*, 57(2):107–116, March 1995.

Figure	Simulation grid size	Particles number or introduction rate	Solid nodes	Interaction particles subset	Avg frame simulation time	Avg surface reconstruction time
7	$40 \times 50 \times 40$	4 676 065	1	1 / 8	45.4 sec	9.5 sec
8	$100 \times 40 \times 20$	1 600 000	1	1 / 16	23.2 sec	4.1 sec
9	$40 \times 40 \times 20$	800 000 / sec	1000	1 / 8	12 sec	9 sec
11	$20 \times 40 \times 5$	250 000	2000	1 / 1	8 sec	1.6 sec

Table 1: Parameters of the different simulations.

[2] N. Chiba, S. Sanakanishi, K. Yokoyama, I. Ootawara, K. Muraoka, and N. Saito. Visual simulation of water currents using a particle-based behavioural model. *The Journal of Visualization and Computer Animation*, 6(3):155–171, July–September 1995.

[3] Eric Climent and Martin R. Maxey. Numerical simulation of random suspensions at finite reynolds numbers. *International Journal of multiphase flows*, 2001.

[4] D. S. Ebert, W. E. Carlson, and R. E. Parent. Solid spaces and inverse particle systems for controlling the animation of gases and fluids. *The Visual Computer*, 10(4):179–190, March 1994.

[5] Douglas P. Enright, Stephen R. Marschner, and Ronald P. Fedkiw. Animation and rendering of complex water surfaces. In *SIGGRAPH 2002 Conference Proceedings*, pages 736–744, 2002.

[6] N. Foster and D. Metaxas. Controlling fluid animation. *Computer Graphics International 1997*, June 1997.

[7] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH 2001 Conference Proceedings*, pages 23–30, 2001.

[8] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, September 1996.

[9] Patrick Fournier, Arash Habibi, and Pierre Poulin. Simulating the flow of liquid droplets. In *Graphics Interface*, pages 133–142, June 1998.

[10] A. Habibi, A. Luciani, and A. Vapillon. A physically-based model for the simulation of reactive turbulent objects. In *Winter School of Computer Graphics 1996*, 1996.

[11] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. *Computer Graphics*, 24(4):49–57, August 1990.

[12] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[13] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.

[14] Gavin S. P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH 88 Conference Proceedings*, pages 169–178, 1988.

[15] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *Computer Animation '95*, pages 198–205, 1995.

[16] Jos Stam. Stable fluids. In *SIGGRAPH 99 Conference Proceedings*, pages 121–128, 1999.

[17] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *SIGGRAPH 93 Conference Proceedings*, pages 369–376, 1993.

[18] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH 95 Conference Proceedings*, pages 129–136, 1995.

[19] Henrik Weimer and Joe Warren. Subdivision schemes for fluid flow. In *SIGGRAPH 99 Conference Proceedings*, pages 111–120, 1999.

[20] Jakub Wejchert and David Haumann. Animation aerodynamics. In *SIGGRAPH 91 Conference Proceedings*, pages 19–22, 1991.

[21] Gary D. Yngve, James F. OBrien, and Jessica K. Hodgins. Animating explosions. In *SIGGRAPH 00 Conference Proceedings*, pages 29–36, 2000.

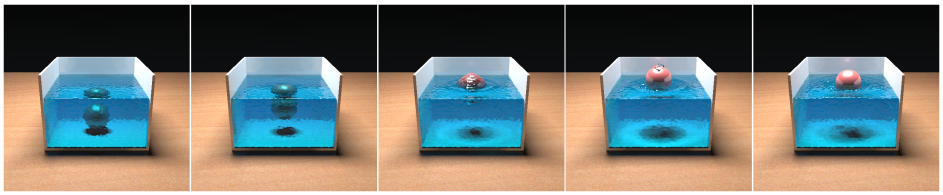

Figure 7: Buyoancy — The ball is initially steady but automatically surface.

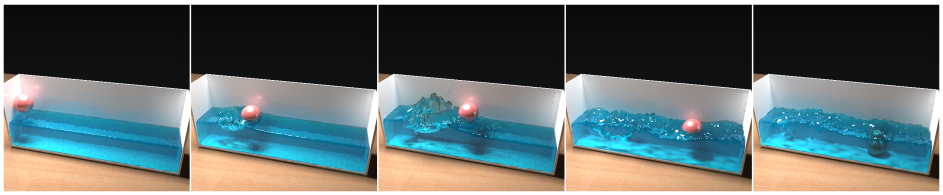

Figure 8: Ricochet of a ball inside a tank.

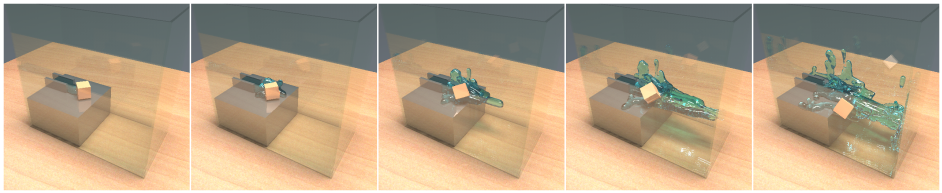

Figure 9: High speed collision of a water jet and a cube.

Figure 10: Close-up of the jet collision with the cube. — In upper row, regular scene is used. In lower row, only the cube is represented.

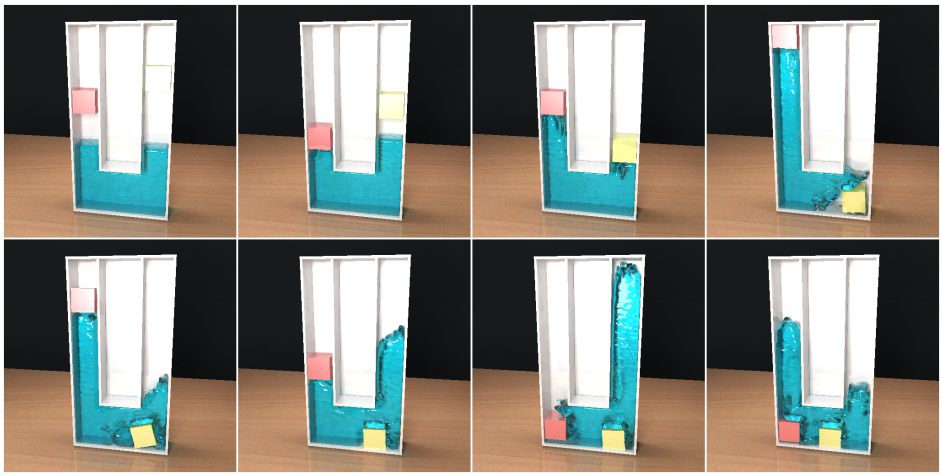

Figure 11: U-shaped container and two cubes.