CEIG - Spanish Computer Graphics Conference (2015)
Jorge Lopez-Moreno and Mateu Sbert (Editors)

Efficient and Robust Position-Based Fluids for VFX

Iván Alduán[1,2], Angel Tena[2] and Miguel A. Otaduy[1]

[1]URJC Madrid, Spain
[2]Next Limit Technologies

Figure 1: *A non-manifold self-intersecting animated dragon falls violently on a pool of water generating violent splashes with high detail. This is an example of a badly-conditioned model, though common in visual effects (VFX) and 3D animation studios, which our solver handles robustly.*

Abstract

Designing a fluid simulator with VFX production pipelines in mind is a difficult task where goals like efficiency, robustness and scalability compromise each other. Many impressive fluid simulation methods have been presented in research papers before, but often they do not meet the production and flexibility demands of artists working on actual VFX production pipelines. In this paper we present a particle-based fluid simulation framework, based on the well known Position-Based Fluids (PBF) method, designed to address VFX production demands. Our framework puts special care on data structure design and implementation details. It highlights cache-efficient GPU-friendly data structures, an improved Z-index sort spatial voxelization technique, tuned-up simulation algorithms, and collision treatment based on VDB fields. Altogether, they empower the artist with a very efficient fluid solver, but also with the robustness and versatility needed for simulating very diverse scenes and effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

Research on fluid simulation has a long history in computer graphics, yet animation artists keep demanding methods that enable larger and richer simulations. Even though all new methods contribute to the improvement and evolution of the field, their adoption within the artist community is subject to other practical factors imposed by production requirements.

Computational efficiency is one of those factors. Better simulation times give artists the ability to iterate quickly over a shot and reach faster the desired look. But efficiency often competes with other practical factors such as constraints on the simulation domain, memory consumption, resolution settings, or controllability.

Constraints on simulation domains, such as the use of fixed domains, are factors that compromise versatility. Artists seek simulation methods that support arbitrary scenes, such as open domains with complicated boundaries.

High memory consumption and resolution constraints are factors that compromise scalability. Artists seek scalable methods that support high-resolution simulations with highly detailed effects under practical memory costs.

Finally, controllability limitations are the factors that

DOI: 10.2312/ceig.20151202

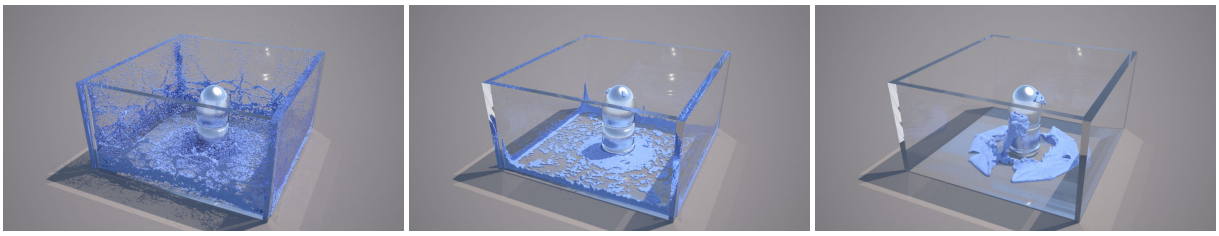

Figure 2: *With our robust XSPH viscosity algorithm, we are able to produce a large range of viscous behaviors. In the snapshots, a sphere of fluid falls over a bullet-shaped object, with behaviors that range from inviscid on the left to toothpaste-like viscous on the right.*

compromise the artists' ability to express their creativity. Artists seek user-oriented methods that allow them to explore in an intuitive manner a diverse set of fluid behaviors.

In practice, artists will even take simulation methods beyond the limits that they were designed for, and expect them to work under diverse and stressful conditions. One typical example is to interact the fluid with solid objects represented using non-manifold meshes, degenerated geometry, and/or self-intersecting geometry. Artists seek robust methods that reach reasonable results even in worst-case scenarios.

This paper presents a particle-based fluid simulator designed to address VFX production demands. The simulator builds on the PBF method by Macklin et al. [MM13]. We propose a combination of fluid model extensions, data structures, and collision handling methods that produce an efficient yet robust and versatile fluid simulation method, suitable for simulating very diverse scenes and effects.

Section 2 reviews previous work on particle-based fluid simulation. Next, Section 3 describes our extension of the PBF method, focusing on the robust coupling of particles of different resolutions, and models for viscosity and surface tension. We incorporate constraint stiffness regularization into PBF, as well as intermediate solver steps to enable high-viscosity fluids. Section 4 describes the treatment of fluid-solid collisions with complex geometry using VDB sparse fields. Section 5 describes our approach to accelerate neighbor searches with a modified Z-index sort algorithm. And Section 6 discusses our data structures for efficient particle management on a highly parallel implementation. Finally, Section 7 and Section 8 show our results and discuss avenues for future work.

2. Review of Particle-Based Fluids

Particle systems have been used for animation since the early days of computer graphics [Ree83]. Due to their Lagrangian nature combined with the absence of connectivity requirements, smoothed particles were first used to simulate elements like fire and smoke [SF95], highly deformable bodies [DG96], and viscous fluids like lava [SAC*99]. Soon after, Smoothed Particle Hydrodynamics (SPH) has become one of the most popular techniques for fluid animation. 3D free-surface particle-based fluid simulation examples at interactive rates were achieved first by Müller et al. [MCG03]. One of the major drawbacks of early SPH methods is the challenge to model highly incompressible fluids. Becker et al. proposed some modifications to achieve better incompressibility [BT07] at the cost of smaller time-steps. Thanks to the versatility of SPH, these formulations have been adapted to support multiphase simulations [MSKG05], viscoelastic fluids [CBP05], or granular media [LD09].

The stiffer the system becomes, the less feasible it is to use purely explicit formulations. Solenthaler and Pajarola [SP09] introduced the PCISPH method, which iterates pressure adjustments to project the density to acceptable values. PCISPH has also been extended to support other materials, such as granulars [AO11] or melting objects [DGP12]. As an alternative to SPH, Premoze et al. proposed the use of the Moving-Particle Semi-Implicit method [PTB*03].

Although PCISPH comes from a physically motivated derivation, it largely resembles an iterative solver for constrained optimization too. The connection between particle fluids and constrained optimization was introduced to graphics by Bodin et al. [BLS12]. Macklin et al. [MM13] recently realized that these constraints could be reformulated to be solved in a position-based manner, gaining additional stability. PBF has been extended to support different media and its interactions [MMCK14].

In each section of the paper we will discuss additional work related to each one of our contributions. For more information on relevant publications in the field, we refer to the state of the art report by Ihmsen et al. [IOS*14].

3. Extensions to Position-Based Fluids

As we have anticipated, our solver uses as baseline the PBF method by Mackin et al. [MM13]. In this section we present several modifications and additions. First, we show how to couple particles of different resolutions in a robust manner. Our current solution does not support dynamic adaptivity of particle resolution, but in practical situations it is convenient

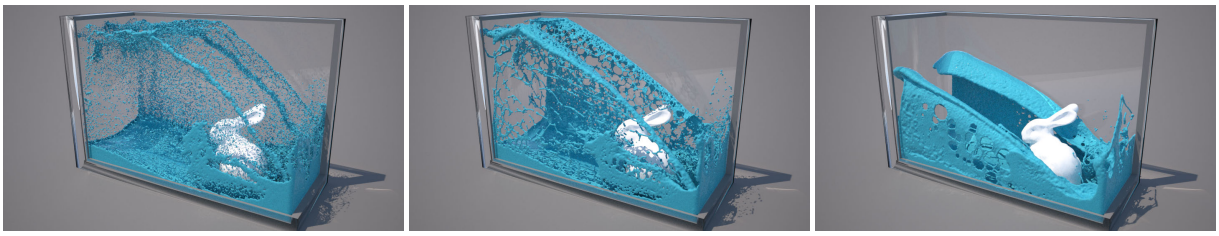

Figure 3: *With our method, we can control the degree of surface tension. In the snapshots, a dam breaks over Stanford's bunny, with growing surface tension from left to right.*

to initialize different fluid regions with different resolutions, e.g., one per fluid emitter. Then, we present contributions for viscosity and surface tension models, which extend the types of behaviors that can be simulated in a controllable way.

3.1. Robust Incompressibility

To enforce incompressibility, PBF defines one constraint C_i per particle, which measures the deviation between the current SPH-based density from the fluid's rest density [BLS12]:

$$C_i = \frac{\sum_j m_j W(\mathbf{p}_i - \mathbf{p}_j, h)}{\rho_0} - 1 = 0. \tag{1}$$

In this expression, $\mathbf{p}_i$ and $\mathbf{p}_j$ represent particle positions, m_j is the particle's mass, W is the SPH kernel, and h is the kernel size. Unlike standard PBF, we allow particles of different resolutions by retaining the influence of particle mass.

The constraint gradients take the form:

$$\nabla_{\mathbf{p}_k} C_i = \frac{1}{\rho_0} \begin{cases} \sum_j m_j \nabla W(\mathbf{p}_i - \mathbf{p}_j, h) & \text{if } k = i \\ -m_k \nabla W(\mathbf{p}_i - \mathbf{p}_k, h) & \text{if } k \neq i \end{cases} \tag{2}$$

Following the PBF approach, we aim to find a set of Lagrange multipliers $\{\lambda_i\}$ which, once applied as position corrections along the constraint gradient, satisfy the constraints in Eq. (1). To compute these λ_i, we regularize the constraint stiffness in a way similar to Solenthaler et al. [SP09]. For each fluid, we compute a constraint stiffness k, using a prototype filled neighborhood:

$$k = \frac{1}{\sum_k |\nabla_{\mathbf{p}_k} C_i|^2}. \tag{3}$$

Then, on each PBF iteration λ_i is updated as:

$$\lambda_i = -\beta k C_i. \tag{4}$$

The coefficient β can be 1.0 or a different positive constant to modulate constraint relaxation.

Particle positions can be corrected $\mathbf{p}_i += \Delta\mathbf{p}_i$ with:

$$\Delta\mathbf{p}_i = \frac{1}{\rho_0} \sum_j m_j (\lambda_i + \lambda_j) \nabla W(\mathbf{p}_i - \mathbf{p}_j, h). \tag{5}$$

Our regularization of the constraint stiffness yields simulations that are computationally less expensive, while the overall fluid behavior is very similar to more accurate PBF models. The regularization is particularly inaccurate for surface particles, i.e., particles with few neighbors, but this is not a major problem in practice, as the behavior of such particles is dominated by surface tension.

3.2. Viscosity

Macklin et al. [MM13] used XSPH [SB12] to maintain coherent motion in their simulations. Beyond this goal, we wish to accommodate robust handling of controllable viscosity, and thus support a wider range of fluid effects. Then, the fluid simulator must be able to support large viscosity values. Unfortunately, we have found that the XSPH velocity smoothing technique easily makes the fluid unstable and gains momentum if the damping coefficient c is larger than 0.5. Mixing explicit forces for viscosity with PBF also results in stability issues.

To model viscosity robustly, we propose to apply the resolution-independent XSPH model [Mon94] in an iterative manner. For damping values c larger than 0.5, we divide the application of the full viscosity into N stable XSPH iterations until they add up to the full desired viscosity behavior. Each of these N iterations is computed as:

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \frac{c}{N} \sum_j \frac{m_j}{\rho_j} (\mathbf{v}_j^n - \mathbf{v}_i^n) \cdot W(\mathbf{p}_i - \mathbf{p}_j, h). \tag{6}$$

With our easily modified XSPH algorithm, the fluid remains stable even under extreme viscosity without the need to decrease the time step, as shown in Fig. 2.

3.3. Surface Tension

Particle-based fluid simulators are often used for the animation of small-scale liquid shots scenarios, such as the glass-filling example in Fig. 4. At such small scales, surface tension becomes a very important factor among fluid forces, hence it becomes compulsory in order to achieve realistic behavior.

In PBF, the major source of attractive forces is the correction imposed by the bilateral incompressibility constraint at

Figure 4: *Filling some glasses with wine shows the ability of our solver to produce realistic small-scale scenes with emerging surface-tension details, but it also showcases the efficient support for unbounded simulation domains.*

low-density regions. Indeed, this attraction may be so large that Macklin et al. [MM13] added a tensile instability repulsion force to improve results. For small-scale effects, where one expects surface tension to be dominant and hence the attractive behavior should be apparent, their repulsion forces are sufficient. For large-scale effects, however, surface tension should not be apparent, and the attractive behavior in PBF is excessive despite the addition of repulsive forces. A valid alternative for reducing attractive behavior is to convert the incompressibility constraint into an inequality constraint [AO11, MMCK14].

Inspired by the work of Alduán and Otaduy [AO11] on friction simulation, we achieve a controllable surface tension behavior (see Fig. 3) through the combination of two elements: (i) limits over the range of valid incompressibility attractive forces, and (ii) a configurable stiffness for the tensile instability force. But the key to artist-friendly controllability is to expose only one parameter, a surface-tension coefficient κ, and based on this we set the maximum attractive force $f(\kappa)$, and the tensile stiffness $g(\kappa)$. We have used linear functions for both $f(\kappa)$ and $g(\kappa)$.

Limits on the attractive forces are simply implemented as a constraint $\lambda_i \leq f(\kappa)$, which is applied after the computation of the Lagrange multiplier in Eq. (4). The tensile instability force is a correction s_{corr} that is subtracted from the Lagrange multipliers in the position correction in Eq. (5) (See [MM13] for details). By making the tensile stiffness dependent on the surface-tension coefficient, the correction is computed as:

$$s_{corr} = -g(\kappa)\left(\frac{W(\mathbf{p}_i - \mathbf{p}_j, h)}{W(\Delta q, h)}\right). \tag{7}$$

4. Collision Detection Using VDB

Many attractive fluid phenomena emerge as the result of collisions with objects. For this reason, the ability to collide with any kind of geometry robustly and efficiently is one of the major requirments for a VFX fluid simulator.

4.1. Review of Collision Detection Methods

Different methods have been proposed for representing object boundaries. One of the most adopted solutions samples boundary geometry with particles, which interact with the fluid through penalty forces. This approach has been applied both to rigid [Mon05, AIA*12] and deformable objects [MST*04, ACAT13]. Penalty forces may be unstable under large time steps and difficult to control. Direct forcing approaches alleviate this problem [BTT09, IAGT10]. However, sampling is difficult to apply when fluids of different resolution interact with the objects.

Alternatively, it is possible to compute the interactions directly with triangle meshes. However, particle-mesh collision detection is costly and time steps must be small to avoid penetrations.

Yet another type of representation is distance fields [JBS06]. The advantage of distance fields is that distance queries are straightforward and inexpensive [FSG03]. In PBF, collision queries need to be executed on each substep iteration, not just once per step, hence fast collision queries are crucial for efficiency. Additionally, dense distance fields can be treated as raw pointers or 3D textures, which simplifies their integration in GPU implementations [HKK07].

4.2. VDB for Fluid Collision Detection

However, production scenes are often large and contain fine details, hence dense distance-field representations would become the memory bottleneck and fail to meet the artist's resolution demands. Instead, we propose the use of adaptive distance fields [FPRJ00]. Specifically, we use narrow-band distance fields stored using VDB grids [Mus13]. The VDB grid is a tree-like data structure with different branching granularity per tree level, support for unbounded domains, activation masks, and efficient cached access.

OpenVDB is the open-source implementation of VDB provided by *Dreamworks, LLC* [Dre15]. This implementation also provides the guidelines on how to compute robust distance fields from geometry, with support for non-manifold surfaces, self-intersections, degenerate faces, and meshes without normals. Such geometric difficulties are common in the models produced by artists (see Fig. 5) and cause simulation problems if they are not properly handled.

Using VDB grids as the base data structure, we have implemented three different modes of distance field computation, which provide the versatility needed to interact with any kind of geometry. For objects with clear inside-outside definition, we provide *solid-inside* and *solid-outside* rasterization modes. For open objects we provide a *shell* rasterization mode which creates an unsigned distance field and then

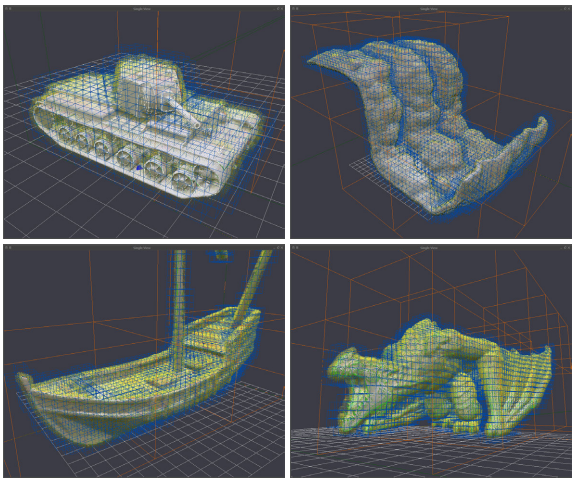

Figure 5: *Robust computation of distance fields in our simulator. The tank and the boat are non-manifold self-intersecting geometries. The waterfall is an example of shell-mode rasterization. The wings of the dragon have no volume, hence they need an additional surface offset to be well represented.*

dilates the zero-valued isosurface. On top of these modes, the user can configure the rasterization cell size, a surface offset for defining the collision isosurface, and a domain offset to define the narrow band where the distance field is computed.

The simulation of collisions with non-static geometry requires a velocity field too. For rigid bodies, the computation of velocity information is straightforward. For hand-animated deformable bodies, such as the dragon in Fig. 1, vertex velocities can be inferred through finite differences from positions in consecutive frames. Then, we propose an efficient way of converting vertex velocities into a velocity field by extending the distance field computation.

In addition to the distance field, we store in the narrow band an auxiliary field with indices to the closest primitives. This auxiliary field is initialized together with the distance field, on the voxels intersected by surface primitives. The auxiliary field is expanded on the narrow band together with the distance field itself at almost no additional cost. The process is demonstrated in Fig. 7. Using the auxiliary field, we compute the velocity field as follows. We copy the VDB tree topology of the distance field into an empty VDB vector field. Then, per active voxel, we obtain the index of the closest primitive from the auxiliary grid. We compute the actual point in the primitive that is closest to the voxel, use its barycentric coordinates to interpolate vertex velocities, and write the resulting velocity at the voxel. Each voxel's calculation is independent, hence the process is trivial to parallelize. Fig. 6 shows two examples of velocity field computation.

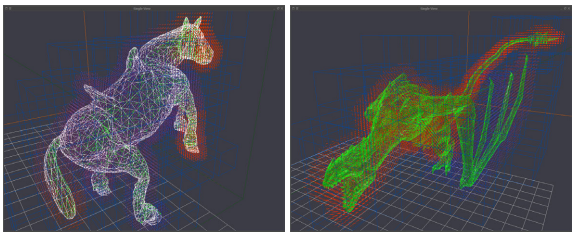

Figure 6: *Visualization of the velocity field for horse and dragon hand-animated characters (red indicates higher velocities).*

The approach described above for the computation of the velocity field can be adapted to compute any secondary field needed by the simulator. For example, if variable friction coefficients are defined on the geometry as a texture, we can reuse the auxiliary field to identify the closest primitive to any voxel, and then compute the barycentric and texture coordinates of the closest point to obtain the friction coefficient for the voxel.

As a limitation, the VDB grid representation is a complex data structure designed to perform well on CPUs, but to date it is too complex to be fully supported on GPUs.

5. Neighbor Search Acceleration

Particle-based fluid solvers require identifying, for every particle, all the neighbors inside a predefined radius. Acceleration algorithms are necessary to avoid the brute-force computation with quadratic cost. In this section, we present an efficient solution that extends the Z-index sort method, but overcomes its simulation domain-size limitations and high memory cost.

5.1. Review of Neighbor Search Methods

Spatial subdivision data structures such as uniform grids can be used to accelerate neighbor queries, but they may become a memory bottleneck. Teschner et al. [THM*03] proposed the use of spatial-hashing to overcome this limitation, but cache-hit rates of this technique are low and hash-collisions cause additional inefficiency.

To avoid the embedding of the geometry into acceleration data structures and gain memory efficiency, index sort approaches maintain during the simulation a cell-ordered array of particle indices, while a dense map points to each cell's range into this array [OD08]. Recently, several authors proposed Z-index sort as a way to accelerate neighbor search queries for SPH, both on multi-core CPUs [IABT11] and on GPUs [GSSP10].

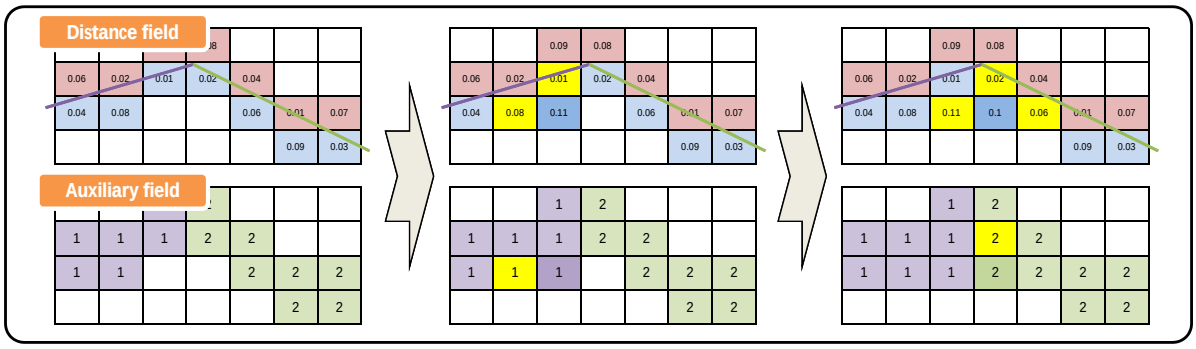

Figure 7: *Illustration of the process to compute our closest-primitive-index auxiliary field along with the expansion of the distance field. Left: the distance field and the auxiliary field are initialized by rasterizing surface primitives. Middle: the fields are expanded into a new voxel, and we compute the distance value from the values at the two neighbors marked in yellow, and we copy the closest-primitive index from the neighbor with shortest distance. Right: the fields are expanded into yet another voxel, and this time the distance is computed from three neighbors.*

5.2. Improved Z-Index Sort

The Z-index sort algorithm assigns unique indices to cells based on a space-filling Z-curve, which provides good cache locality and can be easily computed by bit-interleaving. Goswami et al. [GSSP10] proposed the use of 32-bit Z-indices with 10 bits available per axis. Although GPU friendly, this solution limits the resolution of the acceleration grid to 1024 cells per axis. In production scenes not restricted to a fixed container, this limitation would result in excessively coarse grids and poor neighbor-search performance.

Instead, we propose to use 64-bit indices, which yields 21 bits per axis (See Fig. 8). This results in an acceleration grid with up to 2^{21} cells per axis, which is far more than necessary by any of today's production scenes. Nonetheless, our choice is GPU-friendly, as GPUs support 64-bit integer computations.

To accelerate the evaluation of the Z-index for each particle, we precompute look-up tables per coordinate that facilitate the bit offsetting. At run-time, a bit-wise *or* operation with values from these tables gives us the Z-index.

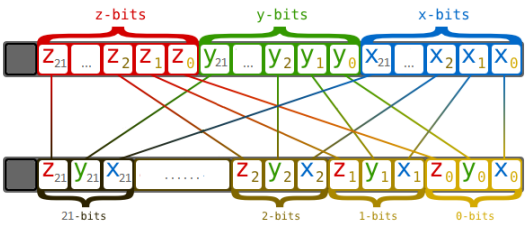

Figure 8: *Cell indices of a Z-curve represented using 64 bits, computed by bit-interleaving with 21 bits available per axis and 1 bit unused.*

The Z-index sort method also requires a grid data structure that stores, for each cell, the range of ordered particles belonging to that cell. Previous work used a dense grid, but with a 64-bit index representation this option is not feasible. Instead, we propose to use a VDB grid to support the Z-Index sort method, thus enabling sparse storage of unbounded domains. To parallelize the construction of the VDB grid, we partition the particles into the number of available cores, create one grid per core, and fill each grid storing on each occupied cell the index of the first particle in that cell. Then, we merge the grids with a reduce operation.

With the proposed modifications, our Z-Index sort method for neighbor-search acceleration is able to support virtually any kind of scene with sparse particle distributions.

6. Particle Data Management

In this section we present details on our particle data management. The choices we made aim to find a good compromise between the efficiency of the simulator and the versatility of the proposed tools.

In production scenarios, particle data may vary in number and size. The particle count could grow dynamically to tens of millions of particles, or drastically disappear due, e.g., to age-based particle killing. Concerning particle data size, even if the particle properties needed for the simulation are fixed, artists like the possibility to add arbitrary channels of information to the particles, which will aid them later in the rendering or compositing process.

For these reasons, we depart from the fixed-size particle data-structure approach used, e.g., by Macklin et

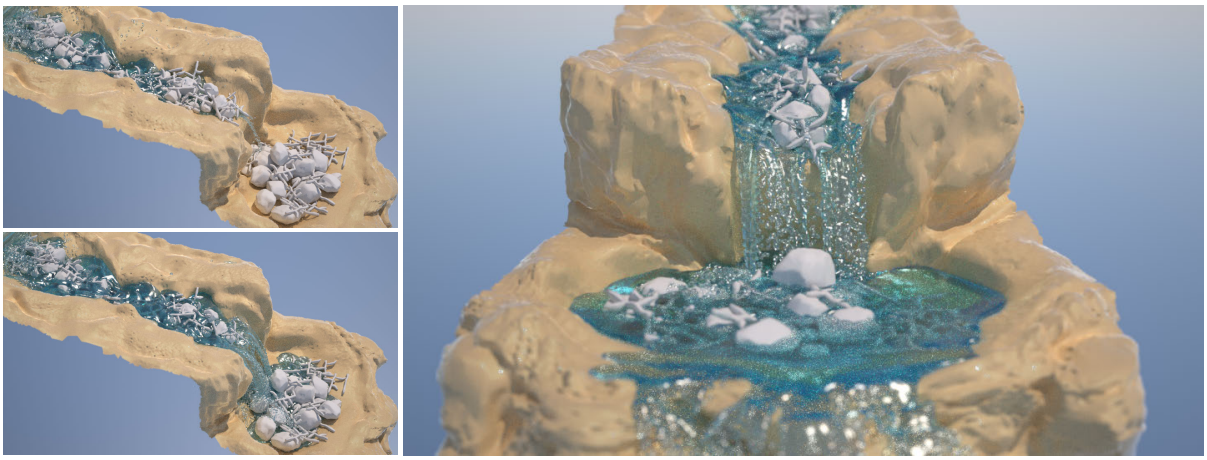

Figure 9: *Our collision detection method based on distance fields and VDB (see Section 4) provides fast and very accurate interaction with detailed geometry even in middle- or large-scale scenarios like the one shown in the images. This scene also showcases robust support of no-volume or shell geometry, often difficult to manage using distance fields.*

al. [MMCK14], and propose a data structure based on independent raw arrays of information, each one representing a property of the fluid. With this data structure, it is trivial to add as many channels of information as desired by the artist, cache efficiency during the simulation is higher, and GPU acceleration can be easily supported by transferring to the GPU only the required channels.

However, with raw arrays of information, extra care needs to be taken to keep efficiency and intelligent use of resources as the simulation evolves, because memory copy operations and data reorganization inside the arrays could seriously affect performance. To minimize large data copy and transfer operations, we change the size of arrays only at predefined capacities chosen by experimentation, balancing memory use and cost of reallocation. To avoid excessive data reorganization as particles disappear, removal operations are handled with an *active-particles* mask, then regularly the arrays are compacted to eliminate gaps, and if occupancy falls below half, the arrays shrink to a lower predefined capacity.

To conclude, we support two different modes of memory operation. When the simulation is paused and the artist interacts with the application and/or constructs the scene, buffers are stored in regular memory. When the simulation starts, buffers are transferred to another pointer to *pinned* or non-pageable memory. The use of pinned memory is more efficient for CPU-GPU heterogeneous executions, but it is risky because it prevents the operating system from reclaiming the reserved memory. As soon as the simulation stops, buffers are returned to regular storage.

7. Results

We have implemented our solver in a completely parallel manner using Intel TBB. On top of TBB, we have created our own custom parallelization model to support symmetric optimizations on all pairwise computations. This gives additional performance gain as demonstrated by Dominguez et al. [DCGG13].

The iterative nature of PBF makes our solver adequate for handling extreme incompressibility. This is shown in Fig. 1, where as soon as the dragon touches the liquid surface, the liquid is repelled and violent splashes emerge. The proposed fluid model can also be configured to produce diverse behaviors, as demonstrated in Fig. 2 and Fig. 3. It also supports a wide range of scenarios, from small-scale to large-scale domains, with their corresponding characteristic effects.

Figure 9 shows a long, large-scale shot, where a river flows into a cascade, and water falls and accumulates naturally forming a lake. Once the lake overflows, the river flows further. This scene demonstrates the benefits of our VDB-

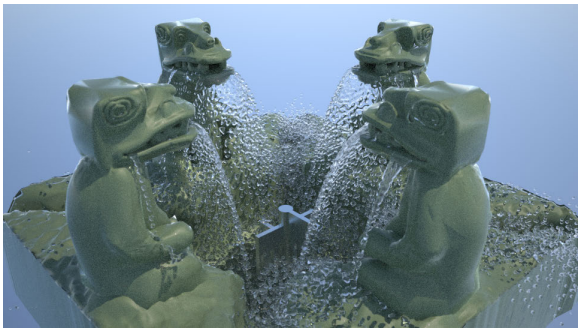

Figure 10: *In this fountain of idols, particles are dynamically created and deleted, but our particle data management handles this situation efficiently.*

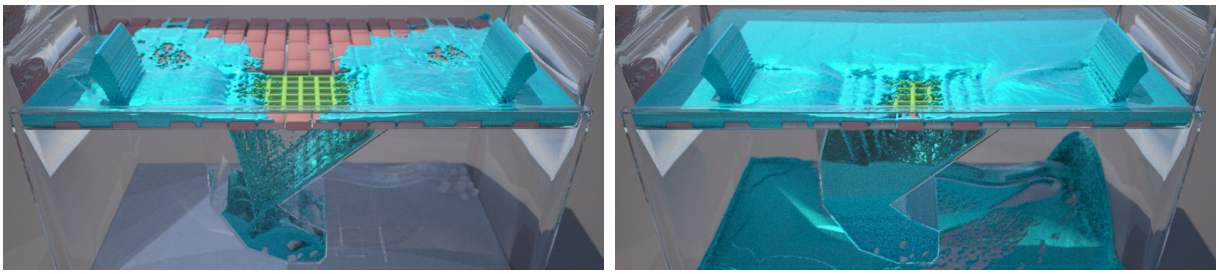

Figure 11: *This mid-scale sewer drain scene showcases a potentially complex scenario for an animated feature film. The liquid covers the road filling the gaps between cobblestones, falls through the grid-like drain, accumulates in a garbage-filled holder, and the drain overflows as the exit pipe is partially clogged and cannot release all the incoming flow.*

based distance-field collision detection. By using a memory-efficient sparse distance-field representation we are able to capture all the fine detail of the original geometry and perform quick particle-boundary collision queries.

Figure 4 shows a small-scale simulation example, where fine surface-tension details are key for the scene's realism. This scene also shows an unbounded scenario where particles fall away. The voxelization technique presented in Section 5 enables fast neighbor search operations in this scenario. The Z-Index sort algorithm preserves locality, and with the sparse storage provided by VDB it maintains memory consumption low.

Fig. 10 shows a scene where particles are dynamically created and deleted. This is a worst-case scenario for our particle data storage based on raw arrays, but our array reallocation policy and mask-based particle removal ensure simulation efficiency.

Finally, Fig. 11 shows a scenario suited for an animation film, with many different geometry elements, and where the fluid naturally evolves around intricate geometry creating interesting effects.

Iteration times and memory consumption for several of the scenes are presented in Table 1.

Scene	#particles ($\times 10^3$)	time step (ms)	memory (MB)
Idols	947	649	465
Wine	1452	1158	585
Drain	1594	2338	921
Waterfall	2049	1839	959
Dragon	6390	3868	1833

Table 1: *Statistics and performance for several scenes. The table indicates peak values for particle count, time-step performance, and memory performance.*

8. Conclusions

In this paper we have presented a production-oriented particle-based fluid simulation framework able to achieve a good compromise between efficiency, scalability, robustness and versatility. The examples shown in the paper demonstrate just a small number of the effects we can already achieve with our solution. Yet there are many features we wish to incorporate to our simulator.

Even though all the pieces of the simulator were designed targeting heterogeneous computing systems, as outlined in Section 5 and Section 6 full GPU acceleration support for our simulator is still work-in-progress.

Currently, we only support one-way coupling of fluids with rigid and deformable objects, as discussed in Section 4. We would like to extend the simulator to handle two-way coupling, being able to simulate at the same time object and fluid dynamics interacting with each other.

Finally, although not shown in our results, our PBF solver is general enough to support any type of constraint. In a similar spirit to the Autodesk Nucleus solver [Sta09], arbitrary types of constraints with different levels of importance can be incorporated to the iterative solver by external programmers. Similar to other work [MMCK14], the ability to simulate different materials such as granulars, gases or non-Newtonian fluids is a very interesting avenue for the future.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, and Next Limit RealFlow's team members for their great effort and support, specially LuisM. [Mor15] for his help with demo production. This research was partially funded by the Spanish Ministry of Education, Culture and Sports (FPU fellowship ref. AP2010-4118), and grants from the Spanish Ministry of Economy (TIN2012-35840) and the European Research Council (ERC Starting Grant no. 280135 Animetrics).

References

[ACAT13] AKINCI N., CORNELIS J., AKINCI G., TESCHNER M.: Coupling elastic solids with SPH fluids. In *Computer Animation and Virtual Worlds (Proc. CASA 2013)* (2013). 4

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics (SIGGRAPH) 31*, 4 (2012), 62:1–62:8. 4

[AO11] ALDUÁN I., OTADUY M. A.: SPH granular flow with friction and cohesion. In *Proc. of 2011 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2011). 2, 4

[BLS12] BODIN K., LACOURSIERE C., SERVIN M.: Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 516–526. 2, 3

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proc. of 2007 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2007). 2

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for lagrangian rigid-fluid coupling. *IEEE Trans. Vis. Comput. Graph. 15*, 3 (2009), 493–503. 4

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proc. of 2005 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2005). 2

[DCGG13] DOMÍNGUEZ J. M., CRESPO A. J., GÓMEZ-GESTEIRA M.: Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method. *Computer Physics Communication 184*, 3 (2013), 617–627. 7

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the 6th Eurographics workshop on Computer Animation and Simulation* (1996). 2

[DGP12] DAGENAIS F., GAGNON J., PAQUETTE E.: A prediction correction approach for stable SPH fluid simulation from liquid to rigid. In *Proc. Comput. Graph. Intern.* (2012). 2

[Dre15] DREAMWORKS, LLC: OpenVDB C++ library, 2015. URL: http://www.openvdb.org/. 4

[FPRJ00] FRISKEN S. F., PERRY R. N., ROCKWOOD A. P., JONES T. R.: Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of SIGGRAPH '00* (2000), pp. 249–254. 4

[FSG03] FUHRMANN A., SOBOTTKA G., GROSS C.: Distance fields for rapid collision detection in physically based modeling. In *International Conference on Computer Graphics and Vision (Graphicon)* (2003), pp. 58–65. 4

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Proc. 2010 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2010), pp. 55–64. 5, 6

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on GPUs. In *Proceedings of Computer Graphics International* (2007). 4

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum 30*, 1 (2011), 99–112. 5

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for PCISPH. In *Proc. of 7th VRIPHYS* (2010), pp. 79–88. 4

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014 State of the Art Report* (2014). 2

[JBS06] JONES M. W., BAERENTZEN J. A., SRAMEK M.: 3D distance fields: A survey of techniques and applications. *IEEE Trans. Vis. Comp. Graph. 12*, 4 (2006), 581–599. 4

[LD09] LENAERTS T., DUTRE P.: Mixing Fluids and Granular Materials. *Computer Graphics Forum (Eurographics) 28*, 2 (2009), 213–228. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *ACM SIGGRAPH/ Eurographics Sympos. Comput. Anim.* (2003). 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics (SIGGRAPH) 32*, 4 (2013), 104:1–104:12. 2, 3, 4

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Transactions on Graphics (SIGGRAPH) 33*, 4 (2014), 153:1–153:12. 2, 4, 7, 8

[Mon94] MONAGHAN J. J.: Simulating free surface flows with sph. *J. Comput. Physics 110*, 2 (1994), 399–406. 3

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68* (2005), 1703–1759. 4

[Mor15] MORA L. M.: Luis M.'s RealFlow tips and tricks, 2015. URL: https://luismma.wordpress.com/. 8

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proc. 2005 ACM SIGGRAPH/ Eurographics Sympos. on Comput. Anim.* (2005), pp. 237–244. 2

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of fluids with deformable solids. *Comput. Anim. Virt. Worlds 15*, 3-4 (2004), 159–171. 4

[Mus13] MUSETH K.: VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics 32*, 3 (2013), 27:1–27:22. 4

[OD08] ONDERIK J., DURIKOVIC R.: Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics, Statistics and Informatics 4* (2008), 29–43. 5

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. *Computer Graphics Forum (Eurographics) 22*, 3 (2003), 401–410. 2

[Ree83] REEVES W. T.: Particle systems: A technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2*, 2 (1983), 91–108. 2

[SAC*99] STORA D., AGLIATI P. O., CANI M. P., NEYRET F., GASCUEL J.-D.: Animating lava flows. In *Proceedings of the 1999 Conference on Graphics Interface* (1999). 2

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Trans. Graph. (SIGGRAPH) 31*, 4 (2012), 61:1–61:8. 3

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of SIGGRAPH '95* (1995), pp. 129–136. 2

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (2009), 40:1–40:6. 2, 3

[Sta09] STAM J.: Nucleus: Towards a unified dynamics solver for computer graphics. In *11th IEEE Computer-Aided Design and Computer Graphics. CAD/Graphics '09* (2009), pp. 1–11. 8

[THM*03] TESCHNER M., HEIDELBERGER B., MUELLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Proc. of Vision, Modeling, Visualization (VMV '03)* (2003), pp. 47–54. 5