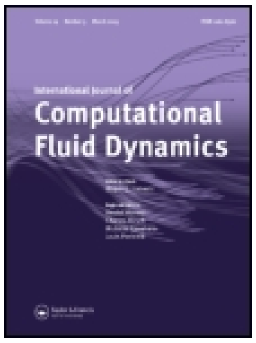
Computer animation challenges for computational fluid dynamics

Article *in* International Journal of Computational Fluid Dynamics · July 2012

DOI: 10.1080/10618562.2012.721541

Won-Sook Lee
University of Ottawa
122 PUBLICATIONS **1,232** CITATIONS
SEE PROFILE

Catherine Mavriplis
University of Ottawa
77 PUBLICATIONS **827** CITATIONS
SEE PROFILE



International Journal of Computational Fluid Dynamics

ISSN: 1061-8562 (Print) 1029-0257 (Online) Journal homepage: http://www.tandfonline.com/loi/gcfd20

Computer animation challenges for computational fluid dynamics

Mauricio Vines , Won-Sook Lee & Catherine Mavriplis

To cite this article: Mauricio Vines , Won-Sook Lee & Catherine Mavriplis (2012) Computer animation challenges for computational fluid dynamics, International Journal of Computational Fluid Dynamics, 26:6-8, 407-434, DOI: 10.1080/10618562.2012.721541

To link to this article: http://dx.doi.org/10.1080/10618562.2012.721541

International Journal of Computational Fluid Dynamics
Vol. 26, Nos. 6–8, July–September 2012, 407–434

Computer animation challenges for computational fluid dynamics

Mauricio Vines[a]*, Won-Sook Lee[a] and Catherine Mavriplis[b]

[a]*School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada;* [b]*Department of Mechanical Engineering, University of Ottawa, Ottawa, Canada*

(*Received 27 March 2012; final version received 8 August 2012*)

Computer animation requirements differ from those of traditional computational fluid dynamics (CFD) investigations in that visual plausibility and rapid frame update rates trump physical accuracy. We present an overview of the main techniques for fluid simulation in computer animation, starting with Eulerian grid approaches, the Lattice Boltzmann method, Fourier transform techniques and Lagrangian particle introduction. Adaptive grid methods, precomputation of results for model reduction, parallelisation and computation on graphical processing units (GPUs) are reviewed in the context of accelerating simulation computations for animation. A survey of current specific approaches for the application of these techniques to the simulation of smoke, fire, water, bubbles, mixing, phase change and solid–fluid coupling is also included. Adding plausibility to results through particle introduction, turbulence detail and concentration on regions of interest by level set techniques has elevated the degree of accuracy and realism of recent animations. Basic approaches are described here. Techniques to control the simulation to produce a desired visual effect are also discussed. Finally, some references to rendering techniques and haptic applications are mentioned to provide the reader with a complete picture of the challenges of simulating fluids in computer animation.

Keywords: computer animation; numerical simulation; visualisation; graphical realism; incompressible fluids

1. Introduction

Fluid simulation has attracted many researchers and practitioners in areas related to computer animation in the past few decades due to the increased demand for realistic and efficient simulations. These are employed in a wide range of applications ranging from interactive training environments and videogames to high-quality and complex simulations for the film industry. Fluid simulation in computer animation is a challenging area due both to the complex phenomena governing fluid behaviour and to the animation requirements that are quite different from those in the mechanical engineering field. The most important differences are:

- As opposed to mechanical engineering simulations, where the main focus is to achieve physical accuracy, in computer animation the main goal is to achieve visual plausibility. Also in many scenarios, animators require having control over the simulation, such as making smoke adopt a specific shape.
- In mechanical engineering, performance is not necessarily critical and simulations may run for long periods of time. In computer animation, the required update rates are much stricter: most interactive applications run at least at 24 updates per second.
- Whereas in mechanical engineering, simulations may be required to run for a predefined amount of time, such as in the study of steady flows, in computer animation simulations must be able to run stably for arbitrary periods of time. In interactive applications such as training systems and videogames, it is usually the user who defines the simulation time.

Due to the complexity of fluid phenomena, there have been very few successful attempts at developing simple *ad hoc* algorithms to reproduce fluid behaviour. Procedural liquids introduced in Fournier and Reeves (1986) are among the few exceptions, where sinusoidal waves are superimposed to reproduce the appearance of a liquid surface. In Peachey (1986), sinusoidal waves are also combined with particles to simulate spray. A large body of work in fluid simulation is focused on methods to solve the Navier–Stokes equations. These techniques are further developed to meet requirements of visual realism, efficiency, stability and animation control.

Foster and Metaxas (1996) introduce to computer animation the first physically based fluid simulations

*Corresponding author. Email: mvine059@site.uottawa.ca

ISSN 1061-8562 print/ISSN 1029-0257 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/10618562.2012.721541
http://www.tandfonline.com

based on solving the Navier–Stokes equations on Eulerian grids. This approach is further developed in Stam (1999), where the first fluid simulations for interactive applications are produced. Since then, grid methods have been the backbone of simulations in computer animation due to straightforward differentiation on grids. However, loss of detail due to discretisation and high computational costs are two major drawbacks of these methods. Several improvements have been made to address these issues. *Adaptive mesh refinement* is used to increase grid resolution in areas of visual interest such as a liquid surface, while grid resolution is decreased in other areas. This helps to reduce the amount of cells in the simulation space while achieving visually plausible simulations. Unstructured meshes have been used to conform to solid boundaries accurately and efficiently divide the space. Here, the *finite volume* method is used and the Navier–Stokes equations are integrated over grid cell volumes. Performance improvements in Eulerian grids are also obtained through *model reduction* by solving the Navier–Stokes equations on a reduced space obtained with pre-computed flow data. Parallelisation is a well-known method of increasing performance by performing computations on several processors simultaneously. A technique that is easily parallelisable is the *Lattice Boltzmann method*. Here, microscopic behaviour of particles is modelled with collisions and propagation on local volume elements, converging to the solution of the Navier–Stokes equations. Lagrangian particle methods have been proposed as an alternative to grid methods due to their efficiency. The most prominent is *Smooth Particle Hydrodynamics*, where flow properties are sampled using particles and interpolated on the space using smoothing kernels. Implementing advection in particle simulations is straightforward; however other processes such as enforcing fluid incompressibility are not. *Hybrid grid and particle methods* have been employed to produce simulations with accurate advection and incompressibility enforcement. All these methods are discussed in Section 2.

Fluid simulation in computer animation is focused mostly on incompressible fluids, such as gases at low speed and liquids. The most common simulations are smoke, fire and water. Smoke is commonly represented as a passive density field which is advected, driven by a buoyant force due to a temperature gradient. Similar techniques are used for fire, which is not incompressible, but has been modelled with these techniques as well by explicitly simulating gas dilation. Flames have also been modelled in a non-physical way as deformable objects under the influence of a wind field allowing animation control. The main focus on modelling water is tracking the free surface, which is the area of visual interest. This is performed using heightfields or level sets. Other processes like mixing, melting and immiscible fluids interaction have also been addressed in computer animation. Furthermore, turbulence is added to increase visual realism using either vortex methods or turbulent energy models with noise particles. A high level of visual plausibility of fluid simulations is achieved by simulating the interaction between solids and fluids and this becomes another focus of attention in computer animation. Also simulation control has been addressed in different ways in order to achieve a specific visual result. A review of these scenarios with their specific techniques is presented in Section 3.

In computer animation there are further issues to address. One of these is producing a realistic visual representation of the simulation data in a process known as *rendering*. Also, fluid simulations in computer animation can be made more efficient by parallelising calculations, which can be done by performing computations on a graphical processing unit (GPU) instead of the central processing unit (CPU). Graphical processing units are highly specialised parallel processors designed to perform geometric and colour calculations and may be used for general purpose computations as well. Fluid simulations have also been incorporated to advanced human computer interfaces such as haptic devices. These devices allow the user to *feel* a virtual object by stimulating the sense of touch. Most haptic devices used to interact with virtual fluids display force feedback to the user, which has to be computed at a rate of 1 KHz – a very challenging update rate for fluid simulations. We discuss techniques related to address these issues in computer animation in Section 4.

2. Main simulation methods

In this part, we review the most commonly used simulation methods for fluids in computer animation.

2.1. Grid-based methods

Foster and Metaxas (1996) introduce the earliest method for fluid simulation using the Navier–Stokes equations in computer animation using forward Euler time integration with a relaxation scheme to compute pressure. Here, a simulation of $50 \times 15 \times 40$ grid cells runs at 2.22 updates per second on a Silicon Graphics Crimson R4000 computer. Stam (1999) addresses stability issues allowing larger time steps. Most current grid solvers are based on this method, and we highlight its main features. The momentum equation's time step is split into each of its terms (external forcing, advection, diffusion, pressure gradient), which are solved sequentially. Incompressibility or zero divergence is then enforced at the last step.

This is a first-order accurate algorithm and its results are adequate for computer animation. External forcing is solved using forward Euler integration. The advection term is solved with a semi-Lagrangian approach known as back-tracing that is used to advect other properties such as temperature and smoke density as well. One downside of this approach is the numerical dissipation it introduces due to interpolation. The diffusion term is solved implicitly by backwards Euler integration commonly with the conjugate gradient (CG) method. The intermediate velocity field is not necessarily solenoidal. Incompressibility is enforced in a process known as *pressure projection*.

The most expensive operations are solving the Poisson equations for diffusion and pressure projection as they require solving a linear system, compared with advection which only requires a particle tracing step and interpolation for each velocity sample. Two main improvements in this method are focused on performing efficient computations for solving the Poisson equations and reducing numerical dissipation due to interpolation in the semi-Lagrangian method.

2.1.1. *Adaptive grids*

One problem that arises in applying grid methods is the trade-off between efficiency and visual results. High detail fluid simulations are achieved by using high resolution grids at a high computational cost. Multigrid methods have been applied in computer animation for speeding up the evaluation of differential equations. These are based on the observation that many relaxation methods smooth high-frequency error very fast and low frequencies are smoothed very slowly. Solving an iteration of a relaxation method on a coarser grid, computational costs can be significantly reduced. Then, the strategy is to evaluate the solution of the whole problem in a hierarchy of discretisations. The highest detail problem is solved via an approximation to a coarser grid which is obtained by downsampling. The coarser grid is solved recursively by an even coarser grid. Solutions are interpolated to a finer grid and are used as initial guess for the next step of the iterative relaxation method.

In certain scenarios such as simulating a pool of water, a crucial region is the liquid surface due to its visual importance. A high resolution grid improves the visual quality of the simulation, but this is not required in areas far from the surface. Adaptive mesh refinement consists of enhancing detail only in areas of visual interest such as object boundaries and liquid surfaces, reducing the computational cost in other areas by keeping a coarser grid. This has been applied both to regular meshes as in Losasso *et al.* (2004) and unstructured tetrahedral meshes as in Batty *et al.* (2010).

Losasso *et al.* (2004) use a staggered grid, where each cell may be refined into eight new cells. All scalar values except pressure are stored at the nodes or corners of the cell to facilitate computations. Coarsening corresponds to creating a new cell where values are obtained by averaging the values of the corresponding fine grid cells. Refinement is achieved by adding new nodes, whose values are obtained by averaging as well. Despite the increased resolution, one problem in the octree structure is the discrete jump in grid size which does not allow a smooth transition between resolutions and introduces distortions in the flow. Batty *et al.* (2010) address this problem by using an unstructured tetrahedral grid for discretisation thereby increasing the visual quality of the flow obtained. This is implemented on a 4-core Intel i7 860 processor, requiring 31 s per time step for simulations employing 400 K to 1.1 M tetrahedrons.

Chentanez and Müller (2011) reduce the grid complexity in liquid simulations by the use of *tall cells* in regular, structured grids. Liquid cells that are far from the surface are merged along the vertical direction only. These tall cells are topped by a layer of regular-size cells that provide the required level of detail at the liquid surface.

2.1.2. *Model reduction*

One way to reduce computational costs of simulations in computer animation is to perform some pre-computation steps in advance and use this information at run-time. Treuille *et al.* (2006) and Wicke *et al.* (2009) adopt this strategy. Treuille *et al.* (2006) combine a flow pre-computation stage with a model reduction approach. Each field in the grid can be seen as an n-dimensional vector, where n is the number of grid cells. The fluid equations are projected into a reduced space of dimension $m < n$ to be solved efficiently. A Galerkin projection is used to obtain the fluid evolution equations in the reduced space. An orthonormal matrix $\mathbf{B}$ is found such that if $\mathbf{u}$ is a vector in $\mathbb{R}^n$ and $\mathbf{r}$ its corresponding vector in $\mathbb{R}^m$, then $\mathbf{u} = \mathbf{Br}$ and $\mathbf{r} \approx \mathbf{B}^T\mathbf{u}$. A linear equation of the form $\partial\mathbf{u}/\partial t = \mathbf{Mu}$ can be expressed in the reduced space by the following Galerkin projection:

$$\frac{\partial \mathbf{r}}{\partial t} = \mathbf{B}^T\mathbf{MBr}.$$

The matrix $\mathbf{B}$ is found through the following pre-computation process. Solenoidal velocity examples are obtained from several high-resolution grid simulations of fluid interacting with solids with free-slip boundary condition. These velocities form a basis $\mathbf{U}$. The orthonormal reduced basis $\mathbf{B}$ is chosen as the first m eigenvectors of $\mathbf{U}\mathbf{U}^T$ since this minimises the reconstruction error:

$$E = ||\mathbf{U} - \mathbf{B}\mathbf{B}^T\mathbf{U}||_F^2.$$

Treuille *et al.* (2006) show that this basis is divergence-free and satisfies the free-slip condition by analysing the linear properties of both constraints. In each simulation step, the momentum equation is split and each linear term is solved directly using the Galerkin projection above. The advection term is linearised by assuming velocities are constant over a time step. This allows the creation of a matrix $\mathbf{A_u}$ that represents the advection given a specific velocity field. Advection is projected into the reduced space in two steps: first the matrix $\mathbf{A_u}$ is represented as a linear combination of advection matrices for the velocity basis in the reduced space as follows:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{A_u} \equiv (r_1\mathbf{A}_{\hat{\mathbf{u}}_1} + r_2\mathbf{A}_{\hat{\mathbf{u}}_2} + \cdots + r_m\mathbf{A}_{\hat{\mathbf{u}}_m})\mathbf{u}.$$

Here, $\hat{\mathbf{u}}_k$, $k = 1, 2, \ldots, m$ represents the basis velocity vectors of the reduced space. Each of the advection matrices is projected then into the reduced space yielding:

$$\frac{\partial \mathbf{r}}{\partial t} = (r_1\mathbf{B}^T\mathbf{A}_{\hat{\mathbf{u}}_1}\mathbf{B} + r_2\mathbf{B}^T\mathbf{A}_{\hat{\mathbf{u}}_2}\mathbf{B} + \cdots + r_m\mathbf{B}^T\mathbf{A}_{\hat{\mathbf{u}}_m}\mathbf{B})\mathbf{r} = \mathbf{A}'\mathbf{r}.$$

This equation is solved exactly using the eigendecomposition $\mathbf{A}' = \mathbf{E}\Lambda\mathbf{E}^{-1}$:

$$\mathbf{r}(t + \Delta t) = \mathbf{E}e^{\Delta t\Lambda}\mathbf{E}^{-1}\mathbf{r}.$$

Wicke *et al.* (2009) indicate that this time integration is unconditionally stable. The diffusion term is represented in the reduced space directly with a Galerkin projection.

One drawback of this method is the restricted set of simulations that can be obtained from the precomputed velocity fields. Wicke *et al.* (2009) address this issue by employing modular reduced models that can be tiled to obtain a full simulation. The simulation space is divided in sub-domains and a reduced basis is obtained per sub-domain. Simulation is performed in each reduced model and additional constraints in the reduced space are used to account for discrepancies in the boundary values of neighbouring domains. A comparison of performance of both approaches using the full simulation and model reduction is presented in Table 1. Examples of simulation results using model reduction are presented in Figure 1.

2.2. Lattice Boltzmann method

One desirable feature in simulation methods is the possibility of parallelising the computations. A scenario where this can be achieved is when computations are performed locally in a reduced region. For this reason, Wei *et al.* (2002) introduce the Lattice Boltzmann method to computer animation for fire simulation. Wei *et al.* (2004a) further develops this method for smoke simulation. The Lattice Boltzmann method is based on cellular automata, and models the fluid from the point of view of the microscopic interaction of packets of molecules. These particles move in a discrete lattice. A lattice specifies the connections between neighbouring cells and therefore defines the possible directions for microscopic particles. Complex lattices are represented in terms of simpler sub-lattice configurations. At the core of this method are propagation and collision rules. A time step simulation consists of updating a packet distribution based on a collision operator and propagating distribution values. This propagation step is discrete and represents a jump from one simulation node to another one. Macroscopic quantities of density and velocity are calculated from distributions. The collision operator is chosen such that mass and momentum are conserved. Wei *et al.* (2004a) define this collision operator based on an equilibrium packet which depends on macroscopic velocity and density (Qian *et al.* 1992). From the equilibrium distribution, the

Table 1. Performance comparison of simulation approaches using model reduction.

Approach	Hardware	Simulation nodes	Seconds per update
Treuille *et al.* (2006) full simulation	3.6 GHz Intel Xeon CPU	$256 \times 64 \times 128$ grid	26
Treuille *et al.* (2006) reduced model[a]		Same as above with 32 reduced bases	0.0017
Wicke *et al.* (2009) full simulation	1.1 GHz AMD Opteron CPU, 16 GB RAM	$248 \times 196 \times 71$ grid	191
Wicke *et al.* (2009) reduced model[b]		Same as above with 14 subdomains and 16 bases	0.024

Note: [a]More than 400 hours of pre-processing were employed. [b]33 hours of pre-processing were employed.

Figure 1. Models used for testing model reduction for fluid simulation in computer animation. Left: Leaves example from Treuille, A., Lewis, A., and Popović, Z. "Model reduction for real-time fluids," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2006, Vol. 25:3. © 2006 ACM, Inc. http://doi.acm.org/10.1145/1179352.1141962. Right: Spacecraft example simulated using the method by Wicke *et al*. (2009) available online at: http://graphics.cs.cmu.edu/projects/modular_bases/ Images courtesy of Carnegie Mellon Graphics Lab.

collision operator is defined with a relaxation time scale. Chen and Doolen (1998) show that the particles' behaviour described with this method approximates that of the Navier–Stokes equations at a macroscopic level. This method has also been employed in other fluid simulations. Zhao *et al*. (2006) develop this method for melting simulation and Park *et al*. (2008) for mixing.

As each particle distribution update is performed locally, the Lattice Boltzmann method can be easily parallelised. Most implementations are performed on parallel graphics hardware. Wei *et al*. (2002) report performance increase by a factor of 50 with this kind of implementation. Notice also that flow is computed without solving for pressure. These two advantages make this approach attractive for interactive applications. Thürey and Rüde (2004) extend this approach by incorporating free surfaces using level sets and a mass tracking method. They illustrate their technique with liquid drop motion. Wei *et al*. (2004b) apply the Lattice Boltzmann method to interaction with different objects. Chu and Tai (2005) use this method to simulate the absorption and dispersion of ink in paper incorporating deposition, liquid percolation and transport in the simulated media. Simulation results obtained using the Lattice Boltzmann method are shown in Figure 2.

2.3. Finite volume method

In the *Finite Volume* method, an irregular, unstructured mesh is used to conform to objects' shapes avoiding distortions due to discretisations over a regular mesh. The finite volume method is based on solving the integral of the Navier–Stokes equations over a specified volume V and applying Stokes' theorem to obtain surface integrals. Again, the momentum equation is split to achieve efficient solutions. The space is usually modelled as an unstructured tetrahedral mesh. Velocities are sampled at tetrahedron faces whereas scalar values are sampled at cell centres or at each tetrahedron's vertices. One of the main advantages of this method is the comparative simplicity with respect to regular grid methods to obtain high quality results with relatively low resolutions for complex solid objects' geometries. Feldman *et al*. (2005) adapt the finite volume method to simulate fluids on tetrahedral meshes that deform by applying a modified semi-Lagrangian advection

Figure 2. Simulation results using the Lattice Boltzmann method. Animation of liquid inside a box. Reprinted from Proceedings of VMV 2004, Thuerey, N. and Rude, U. Free surface Lattice-Boltzmann fluid simulations with and without level sets, pp. 199–207, Copyright (2004), with permission from IOS Press.

method that accounts for the mesh deformation. Wendt *et al.* (2007) uses this method for modelling fluid interaction with complex boundary geometry. Klingner *et al.* (2006) use a tetrahedral mesh that conforms to solid boundaries and it is recalculated at each time step as solid boundaries move. At each time step, fluid properties stored on the mesh are transferred to the new grid that is generated.

Miklós (2004) employs the finite volume method for wave simulation. A height-field approximation is used to model a liquid surface and an additional layer is used to represent the overturning of waves. Brochu *et al.* (2010) apply a finite volume method in adaptive tetrahedral mesh simulations. Sin *et al.* (2009) construct a mesh from particles and perform pressure projection similarly to the finite volume method. Chentanez *et al.* (2007) generate a tetrahedral grid from a triangular mesh that defines the surface using semi-Lagrangian contouring (see Section 3.1.1). Incompressibility is enforced locally on each tetrahedron. To avoid loss of volume due to features smaller than a grid size, artificial thickening is applied to areas with small features.

2.4. *Fourier transform-based solvers*

There are specific scenarios where high detail simulations can be obtained at low computational costs. Such is the case of simulating flow with periodic boundary conditions. Although this kind of flow does not exist in nature, it can be used to represent large bodies of water such as the ocean. Ocean wave modelling is probably where Fourier Transform methods have been applied most extensively in computer animation. The ocean surface is modelled as superimposed sinusoidal waves which are efficiently decomposed using the fast Fourier transform method (FFT) for efficient computation. Random wave amplitudes produce visually plausible results.

A wave is characterised by its wave vector **k** and its magnitude, the wave number k. The direction of a wave is perpendicular to the wave crest; the larger the wave number, the smallest the space between waves. A function is represented as a sum of waves, and the Fourier transform assigns the corresponding weight to each. Tessendorf (2001) presents a technique which has become the base for many current methods. Here, the ocean surface is modelled as the summation of complex sinusoids for different wave vectors. Waves are propagated in both positive and negative directions in Fourier space. It is possible to relate the wave number k and its frequency. In deep waters for example, this relationship is given by the wave number and the gravitational acceleration as $\omega^2(k) = gk$. Producing a visually attractive appearance of the ocean surface is achieved by generating random Fourier amplitudes h_0 for waves as follows:

$$h_0(\mathbf{k}) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P_h}.$$

Here, ξ_r and ξ_i are two random variables with Gaussian distribution and P_h is a modified Phillips spectrum, which is a semi-empirical ocean wave fluctuation model employed in oceanography. Tessendorf (2001) uses the following:

$$P_h(\mathbf{k}) = A\frac{e^{-1/(kL)^2}}{k^4}||\hat{\mathbf{k}} \cdot \hat{\mathbf{w}}||e^{-(kl)^2}.$$

Here, A is a simulation parameter and **w** is the wind direction; L corresponds to the largest possible wave and $l < L$ is a cut-off amplitude used to improve convergence.

Cieutat *et al.* (2003) further develop this method by incorporating interaction with solids, specifically ships. Chiu and Chang (2006) enhance the approach in Tessendorf (2001) with adaptive surface tessellation. Examples of ocean waves produced with Fourier Transform-based techniques are presented in Figure 3.

Stam (2001) also applies Fourier Transform-based methods achieving efficient and simple simulation of fluids with periodic boundary conditions. Lapierre *et al.* (2003) develop a Fourier domain advection step. More recently, Long and Reinhard (2009) present a 3D simulation model using cosine and sine transforms simulating free-slip boundary conditions.

2.5. *Smoothed particle hydrodynamics*

Although traditional grid methods allow accurate simulations, they require extremely high resolutions to handle thin features in liquids such as splashes and foam, considerably increasing the computational cost of such simulations. This motivates modelling the fluid using particles. These are either emitted to model an inflow, or initialised. For efficiency and to ensure particles are not only emitted, particles are removed after a time limit. A particle encodes its position, velocity, mass and the forces acting on it.

Desbrun and Cani (1996) introduce the technique of smoothed particle hydrodynamics (SPH) to computer graphics to simulate motion of deformable objects. Here, a continuum is simulated with a limited set of discrete particles by interpolating quantities stored in each particle over locations where no particle is present using a smoothing kernel function. A kernel is a finite support, normalised radial basis function that defines each particle's contribution to the computations in arbitrary locations.

Two properties of this kind of simulation that make this technique attractive are: (1) by keeping the amount of particles and the mass of each particle constant, mass conservation is automatically satisfied and (2) as particles move with the fluid, the time derivative of the particle's velocity corresponds to the substantial derivative of the velocity field, and therefore the advection term does not need to be calculated explicitly. Other terms in the Navier–Stokes equations are computed by calculating the influence of particles on each other using the smoothing kernels. Computing the pressure term has been addressed in different ways. Müller *et al.* (2003) employ an ideal gas equation to relate pressure and density, which is calculated from particles. This approximation is used for liquids and gases and results are those of a compressible flow. Premože *et al.* (2003) enforce incompressibility by solving a Poisson equation. This yields accurate results, however at a high computational cost. Becker and Teschner (2007) employ the Tait equation to relate density and pressure on liquid simulations obtaining flows with minimum compression. A performance comparison for the above approaches is presented in Table 2. Solenthaler and Pajarola (2009) propagate density fluctuations throughout the fluid and pressure is adjusted in a prediction-correction scheme. Adams *et al.* (2007) further improve performance by adaptively increasing particle resolution at liquid surfaces achieving a speed-up of computation time by a factor of 2.8 to 7.7 in their experiments. Solenthaler and Gross (2011) simulate flow at two scales. Physical fluid properties are determined at a low resolution simulation, whereas areas of visual interest are simulated at higher resolution. Boundary conditions are defined for areas of different resolutions and force feedback is computed between them to achieve a consistent simulation. Computational costs in this approach are reduced by a factor of up to 6.7. Goswami and Pajarola (2011) determine convergence conditions and advection approximations based on identification of active and passive particles to speed up SPH simulations.

There are several challenges associated with SPH simulations. Irregular particle distribution and extraction of smooth surfaces for liquids using particles are both major problems as they may affect the visual quality of the simulation results. Also, defining a proper smoothing kernel radius to obtain a visually realistic simulation is not trivial for every scenario. These are among the major limiting factors for the application of SPH to large simulations.

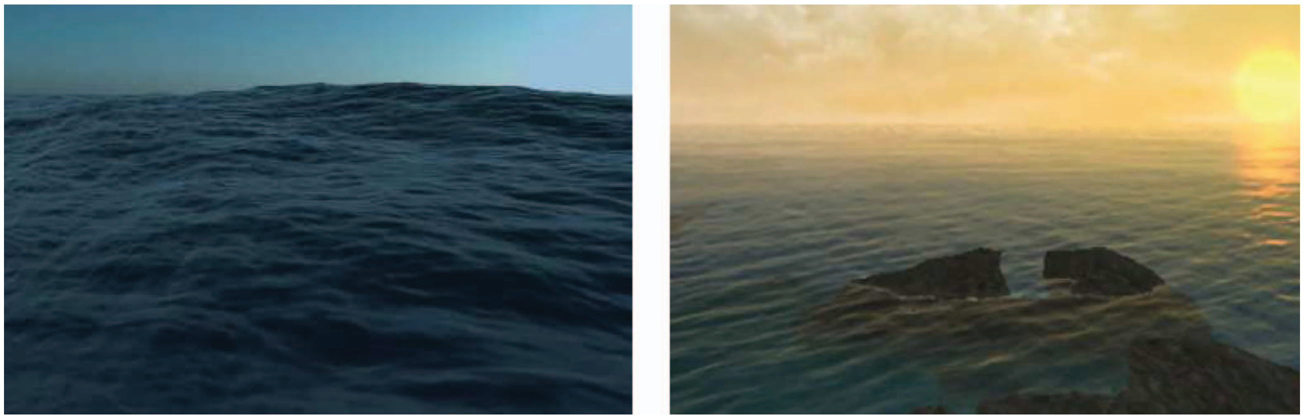


Figure 3. Ocean simulations obtained using Fourier Transform methods. Left: ocean rendered using the method from Tessendorf, J., 2001. Simulating ocean water. In: SIGGRAPH 2001 course notes (Course 47). ACM. Copyright (2001) J. Tessendorf. Right: results obtained using the method in Chiu and Chang (2006). © 2006 IEEE. Reprinted, with permission, from Chiu, Y.F. and Chang, C.F., 2006. GPU-based ocean rendering. In: 2006 IEEE international conference on multimedia and expo, 9–12 July, Toronto, Ontario, Canada. IEEE, pp. 2125–2128.

Table 2. Performance comparison of particle simulation approaches.

Approach	Hardware	Simulation nodes	Updates per second
Müller *et al.* (2003)	1.8 GHz Pentium IV CPU	1300 particles	25
Premože *et al.* (2003)	1.7 GHz Pentium IV CPU, 512 MB RAM	10 K particles	1
Becker and Teschner (2007)	3.4 GHz Pentium IV CPU	200 K particles	0.13

2.6. Hybrid methods

As mentioned earlier, one problem with the semi-Lagrangian advection method proposed by Stam (1999) is smoothing due to interpolation from the grid values. This is reflected as an additional viscosity which dampens the fluid. Hybrid methods have been applied to address this issue, by solving advection using particles and enforcing incompressibility using the Eulerian grid. The most common hybrid approach is the particle-in-cell (PIC) method. Zhu and Bridson (2005) apply this for simulating sand, and Horvath and Geiger (2009) for fire. Particles encode the fluid properties; these are averaged into the grid, where all terms of the Navier–Stokes equations are calculated, except advection. Then, the particles are advected using the grid velocity. The PIC method is improved into the fluid implicit particle (FLIP) method as follows. Particle information is not directly updated with data from the grid. Instead, the variation of a quantity is computed from the grid, and this in turn is used to update particle information. This produces high quality results, as it does not introduce numerical dissipation. Zhu and Bridson (2005) note this approach introduces noise, and a combination of PIC and FLIP updates is used in practice. An example flow obtained by this hybrid approach is shown in Figure 4. This idea has been applied in the context of SPH simulations as well. Raveendran *et al.* (2011) interpolate particle velocities to a coarse grid, where they solve a Poisson equation for pressure. Then, pressure values are interpolated back to particle positions, where pressure forces between particles are computed.

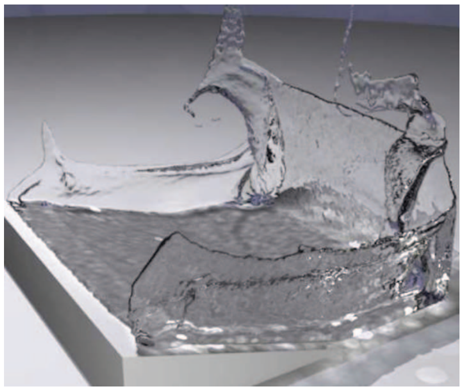


Figure 4. Results of simulating sloshing liquids using particle methods using a hybrid PIC-FLIP particle simulation (see Section 2.6) from Zhu, Y. and Bridson, R. "Animating sand as a fluid," ACM Transactions on Graphics – Proceedings of ACM SIGGRAPH 2005, Vol. 24:3, © 2005 ACM, Inc. http://dx.doi.org/10.1145/1073204.1073298/

3. Computer animation fluid applications

3.1. Liquids

The main issue in simulating liquids is to track the location of the surface due to its visual relevance. One alternative proposed early on by Foster and Metaxas (1996) is to seed particles in the fluid and use them to track the location of the fluid. After several simulation steps however, particle positions may be uneven, producing gaps in the fluid surface. Also, extracting a smooth surface for rendering from particles is not a trivial task. In the following, we outline some common methods to model a free surface that produce visually appealing results.

3.1.1. *LevelSet methods for water surface simulations*

The level set technique is based on the construction of an implicit surface function ϕ whose zero-set represents the liquid surface. This function is created from simulation samples, either grid locations or particles. Usually, the level set function values are set to be negative inside the fluid and positive outside. Given an implicit surface defined in the simulation space, calculating the signed distance function can be done by numerically approximating $||\nabla\phi|| = 1$. Geometric methods are based on searching the closest point on the surface for grid locations around the surface. For each grid location, the signed distance to the surface is calculated and this is used to update the signed distance for other grid locations. A popular method to compute the signed distance is the fast marching algorithm described in Sethian (1999).

The level set is advected with the fluid velocity using the semi-Lagrangian scheme in Stam (1999), which smooths the level set. Since this method depends on the grid resolution, liquid is prone to losing fine detail and volume due to flow irregularities. To avoid this problem, the level set method is combined with particles to avoid excessive smoothing. Foster and Fedkiw (2001) propose seeding particles only near the fluid surface. Each particle is given a radius and the spherical shell of the particles defines an isocontour of the fluid surface. Once the level set is defined, both the level set and the particles are advected. Particles near the boundary provide additional information which the level set alone cannot capture. Areas of high curvature of the level set indicate fluid splashing and particle information is used to locally modify the level set.

Enright *et al.* (2002) further develop the above approach by seeding particles on both the liquid and empty sides of the surface, and storing their distance to the free surface. Particles that are advected far from the surface on the opposite side are considered *escaped*.

They are used to indicate where the level set is inaccurate and then the level set is modified accordingly.

Several authors have applied level sets for simulating liquids. Adams *et al*. (2007) use level sets to render liquids modelled with SPH. Losasso *et al*. (2008) develop large simulations with high-quality interaction of solids and fluids by coupling SPH simulations and level sets. Losasso *et al*. (2006b) and Kang *et al*. (2010) have applied level sets to model interfaces between different liquids. Mihalef *et al*. (2009), Hong and Kim (2005) and Hong *et al*. (2008) demonstrate the level set technique for modelling bubbles. Examples of liquids modelled with level sets are shown in Figure 5.

Bargteil *et al*. (2006) present the Semi-Lagrangian contouring method to track liquid surfaces. Here, a distance function is defined such that its zero set corresponds to the liquid surface. At each time step, this function is advected using a semi-Lagrangian scheme. Then, a triangular mesh is constructed from the zero-set of the distance function. This explicit polygonal mesh representation of the liquid surface avoids undesirable artefacts when advecting a triangular mesh directly, such as self-intersecting portions of the mesh.

Wojtan *et al*. (2009, 2010) also address evolving an explicit polygonal liquid surface. Here, a distance field is defined and the liquid surface corresponds to its zero-set. Through the evaluation of the distance field, topological changes, such as merging and splitting of bodies of fluid, are detected and the surface is modified accordingly. Yu *et al*. (2012) apply this idea to SPH simulations, where the liquid surface is advected directly using the velocity stored in particles. Then, this surface is projected into an implicit surface defined by the particles. Surface tension is incorporated to the SPH particles as an additional force computed from the mean curvature of the liquid surface mesh. Kim *et al*. (2006) employ a level set to track the main body of liquid on a grid simulation using a hybrid PIC/FLIP method. Particles are then used to track fluid mass and volume in sub-grid features, and the physical interaction between them and the main body of liquid is simulated. Misztal *et al*. (2010) apply explicit surface tracking in tetrahedral meshes. Here, the tetrahedral mesh adapts to the liquid surface evolution. Then, surface tension and solid boundary conditions are solved as an optimisation problem.

3.1.2. *Height field approximations*

Scenarios like the surface of a lake can be modelled with very simple techniques that produce real-time simulations that are adequate for interactive applications. Kass and Miller (1990) introduce several simplifications of the fluid equations to achieve realistic and very efficient water surface motion. First, the surface is assumed to be a height field. Second, only horizontal

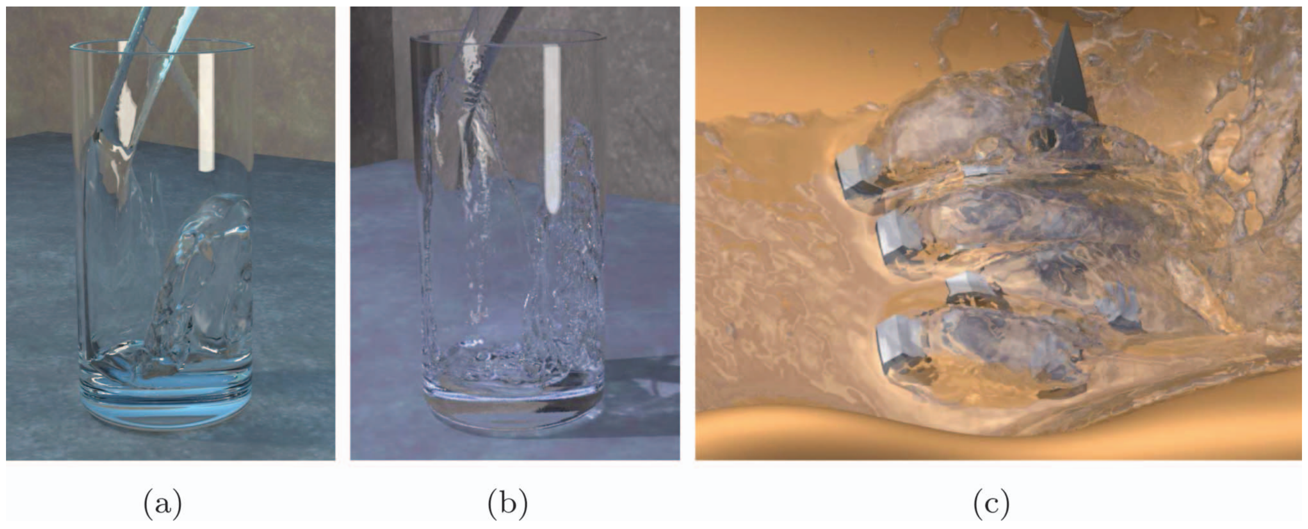


Figure 5. Example water simulations using level set techniques: (a) results from Enright D., Marschner S., Fedkiw R. "Animation and rendering of complex water surfaces," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2002, Vol 21:3 © 2002 ACM, Inc. http://doi.acm.org/10.1145/566654.566645. (b): The same scene as in (a) simulated with SPH from Losasso *et al*. (2008). © 2008 IEEE. Reprinted, with permission, from Losasso, F., Talton J., Kwatra N., Fedkiw R. "Two-way coupled SPH and particle level set fluid simulation". IEEE Transactions on Visualization and Computer Graphics, Vol. 14:4. pages 797–804. (c): Water level set constructed using adaptive particles from Adams B., Pauly M., Keiser, R., Guibas L. "Adaptively Sampled Particle Fluids," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2007, Vol 26: 3, © 2007 ACM, Inc. http://doi.acm.org/10.1145/1276377.1276437.

components of the flow velocity are assumed to be relevant. Third, the horizontal velocity of the waves is approximately constant. In these cases, the Navier–Stokes equations can be simplified in 2D to:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + g\frac{\partial h}{\partial x} = 0,$$
$$\frac{\partial d}{\partial t} + \frac{\partial ud}{\partial x} = 0.$$

Here, $h(x)$ is the height of the surface and $d(x)$ is the depth, defined as $d(x) = h(x) - b(x)$. Here, $b(x)$ is the height of the container at the bottom of the fluid body. These equations can be further simplified by ignoring the advection term and linearising around a constant value of h. By differentiating and combining the resulting equations, a second order wave equation for h is obtained. This equation has been applied in computer animation to produce simple waves and it can be easily extended to 3D. This technique allows representation of the liquid surface as a polygonal mesh where each vertex is advanced using the wave equation. Yuksel *et al.* (2007) improve this approach by incorporating particles into the height field calculation to simulate waves interacting with solid objects at very fast update rates. A large simulation consisting of 1681 boats totalling 295,856 polygonal faces floating on a height field of resolution 256 $\times$ 512 and 8 million particles is shown in Figure 6. This scene was simulated on a commodity 2.13 GHz Core 2 Duo processor achieving 4.84 updates per second. Chentanez and Müller (2010) solve a similar form of the above equations on a grid which is combined with particles to produce effects that cannot be replicated within a height field model, such as splashing and overturning waves. These particles carry some mass and momentum which are removed from the main body of liquid. These quantities are re-inserted in the height field as particles fall into the main body of liquid.

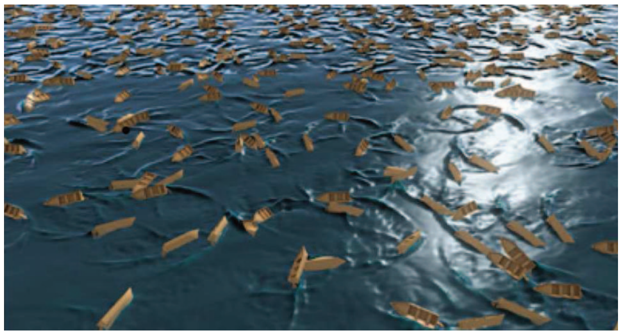


Figure 6. Water surface generated with a height field with solid interaction from Yuksel, C., House, D., Keyser, J. "Wave Particles," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2007, Vol. 26:3, © 2007 ACM, Inc. http://doi.acm.org/10.1145/1276377.1276501.

3.2. *Smoke and fire*

Simulating smoke and fire is achieved by simulating a gas that fills the entire simulation space. Now, both temperature T and smoke density s have to be calculated. Fedkiw *et al.* (2001) evolve temperature and smoke density with an advection–diffusion and source transport equation. This is solved using the same methods developed by Stam (1999).

Buoyancy is taken into consideration by a Boussinesq approximation in the velocity equation. Detailed vertical smoke patterns are obtained by calculating a term that accounts for lost vorticity and introducing it into the flow as an additional external force in the Navier–Stokes equations. This technique is known as *vorticity confinement* and is discussed in Section 3.5.1. An example of the results obtained is shown in Figure 7.

Fedkiw *et al.* (2001) represent smoke density as a passive quantity that does not affect the flow. Bridson (2008) incorporates density variations as an effect of temperature. As fluids expand due to temperature gradients, mass is not conserved if the pressure projection step in Stam (1999) is applied. Bridson (2008) corrects this by incorporating density variations via the divergence (by applying conservation of mass) to the Poisson equation for pressure.

Fire has been extensively studied in computer animation due to its visual features and the demand in the entertainment industry. To model fire, there are basically two problems that have to be addressed: the motion of the flame and the effect of the temperature on the surrounding fluid. Nguyen *et al.* (2002) model the region where the combustion takes place, or core, with a surface represented as a level set. Assuming a constant burn speed S, the evolution equation for the level set becomes:

$$\frac{\partial \phi}{\partial t} + \mathbf{u}_{\text{fuel}} \cdot \nabla \phi = S.$$

Here, the velocity of the fuel, $\mathbf{u}_{\text{fuel}}$, is a simulation parameter. An example of the core used in this approach is shown in Figure 7.

Fluid expansion is simulated when crossing the flame front. This expansion can be expressed as the following volume variation:

$$\Delta V = \left(\frac{\rho_{\text{fuel}}}{\rho_{\text{burnt}}} - 1\right)S.$$

Here, ρ_{burnt} is the density of the burnt gas, which is assumed to be smaller than the density of the fuel. The velocity of the burnt products is computed from the fuel velocity plus the speed of the volume variation in the normal direction of the flame front. As in the case of smoke, the expansion process is taken into account in the pressure projection step. A weighted average of the density of the burnt products and the fuel can be considered, and the divergence value adjusted accordingly. As soon as the burnt products cross the flame front, they enter the simulation with a predefined temperature $T_{\max}$. Then, the decay of temperature is modelled by a black-body radiation conservation equation with a constant to control the amount of radiation in the simulation.

Zhao *et al.* (2003) propose a model of expanding flames on solid fuels that are discretised using a grid. Flames are modelled as particles following a wind velocity field computed with the Lattice Boltzmann method. The fire front evolves tangentially to the surface of the fuel. This approach is efficiently implemented on a GPU achieving 14.2 updates per second for a 67 × 128 × 67 grid with an object consisting of 58,837 voxels. An external wind field allows control over the animation, directing the fire in a specific direction and consuming an object in a specific manner. An example of the results obtained with this approach is shown in Figure 8.

Bridault *et al.* (2007) use a multi resolution approach to handle multiple simultaneous flame simulations, such as several candles on a table. Small flame details are simulated with a small grid which is

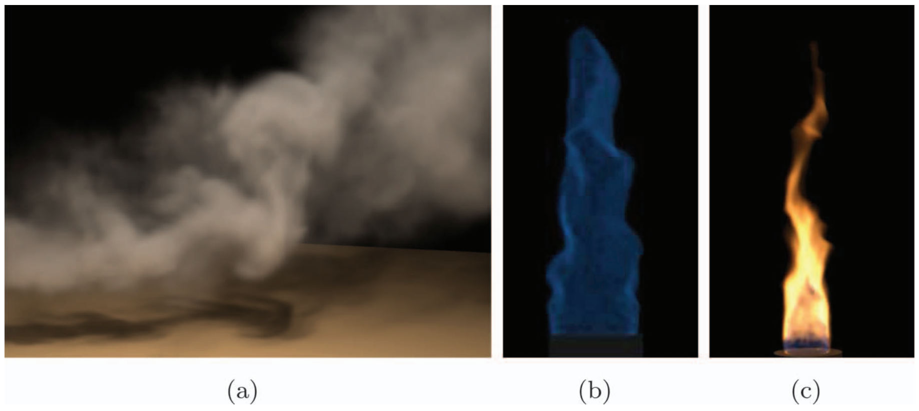


Figure 7. (a) Smoke results from Fedkiw R., Stam J., and Jensen H. "Visual simulation of smoke," SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, © 2001 ACM, Inc. http://doi.acm.org/10.1145/383259.383260. (b) Example combustion core and (c) flame result from Nguyen D., Fedkiw R., Jensen H. "Physically Based Modeling and Animation of Fire," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002, Vol. 21:3, © 2002 ACM, Inc. http://doi.acm.org/10.1145/566654.566643

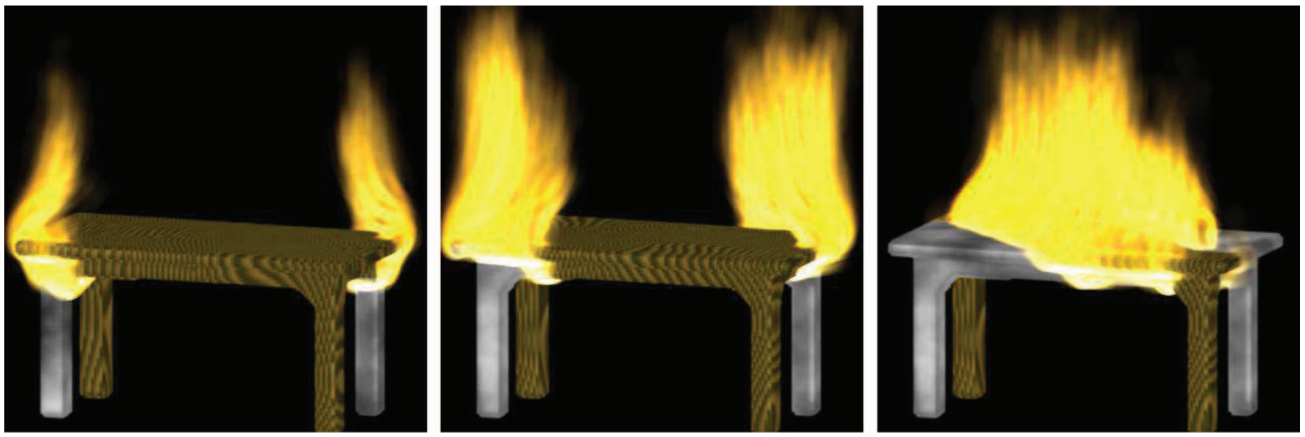


Figure 8. Snapshots of animation frames of a wooden table consumed by flames from Zhao *et al.* (2003). Notice the tangential advancement of the fire front due to the presence of fuel in areas not yet burnt. © 2003 IEEE. Reprinted, with permission, from Zhao, Y., Wei, X., Fan, Z., Kaufman, A., Qin, H. 2003. "Voxels on fire," Proceedings of the 14th IEEE visualization 2003 (VIS'03), VIS '03. Washington, DC: IEEE Computer Society, pages 271–278.

embedded in a larger, coarser grid for global control of the simulation, for instance, wind entering through a window. Flames located far from the camera are simulated with particles as they do not require a high level of detail. A performance of 25 updates per second is achieved in 153 grids for up to 16 flames on a an Intel Core 2 Duo 6300 processor, before applying their multiple resolution approach which allows up to 256 flames.

More realistic production of fire is usually performed as an off-line process. For example, Hong *et al*. (2007), simulate fire using a complex Chapman–Jouguet detonation model for the front velocity, obtaining complex patterns that enhance the visual quality of flames. An example of the resulting flames is shown in Figure 9. Horvath and Geiger (2009) perform a coarse simulation step using PIC and FLIP methods with a hierarchical decomposition. Here, lower resolution grids are computed with a low-pass filter. Then, velocities on the coarse grid are interpolated to locations of a higher resolution grid and are subtracted from the original values. This way, only high frequency details are kept at each grid, forming a hierarchy of different levels of detail where incompressibility is enforced at each level. Artists can increase the amount of rotational motion through vorticity confinement (See Section 3.5.1).

Achieving simulation control, as often required in the film industry, is achieved by simulating fluids departing from physically based models. One common alternative is to represent a flame by a skeleton and a surface surrounding it. The skeleton consists of a series of connected nodes which move following a flow. Beaudoin *et al*. (2001) employ a user-defined flow and flames do not affect the fluid motion. Lamorlette and Foster (2002) simulate flame separation by breaking the skeleton, and noise is added to improve the flame's visual appearance. Fuller *et al*. (2007) and Vanzine and Vrajitoru (2008) follow the same approach and add noise in three octaves of the Kolmogorov spectrum at 30 updates per second for a lattice with 135 points. Examples of flames produced with the above skeleton and noise techniques are presented in Figure 10.

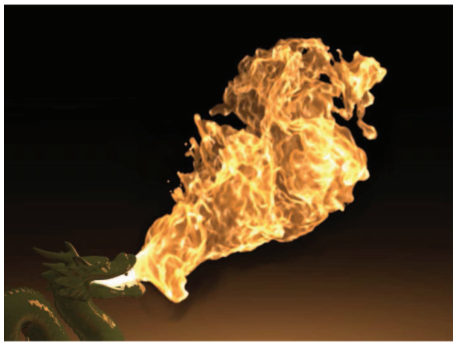


Figure 9. Fire simulation from Hong, J.-M., Shinar, T., Fedkiw, R. "Wrinkled Flames and Cellular Patterns," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2007, Vol. 26:3, © 2007 ACM, Inc. http://doi.acm.org/10.1145/1276377.1276436.

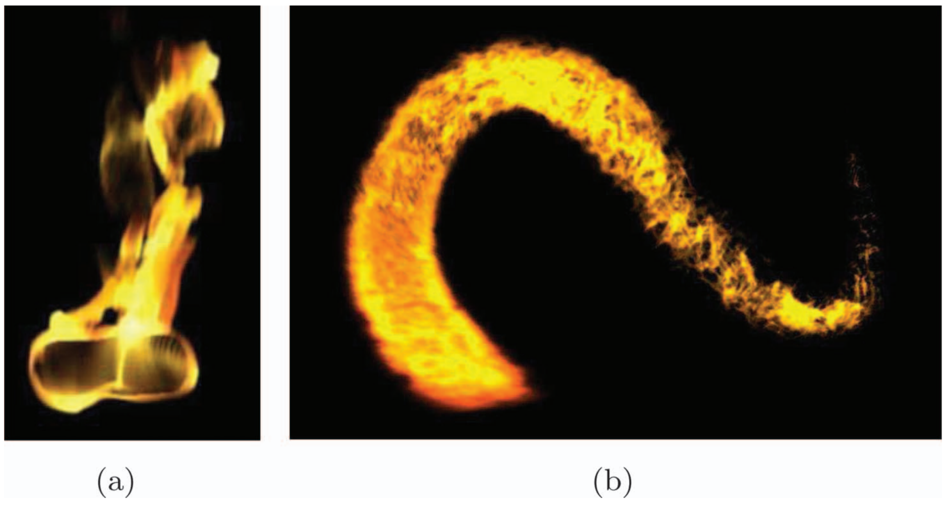


Figure 10. Examples of fire simulation. (a): Torch flame from Lamorlette, A., Foster, N. "Structural Modelling of Flames for a Production Environment," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2002, Vol. 21:3, © 2002 ACM, Inc. http://doi.acm.org/10.1145/566654.566644. (b): Flame results from Fuller, A.R., Krishnan, H., Mahrous, K.M., Hamann, B. and Joy, K.I. "Real-time procedural volumetric fire," in: Cohen, J.D., Turk, G. and Watson, B.A., eds., Proceedings of the 2007 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D), ACM Press, New York, New York, pages 175–180. © 2007 ACM, Inc. http://doi.acm.org/10.1145/1230100.1230131.

Along with fire, explosions are interesting phenomena to simulate due to their visual features. Yngve *et al.* (2000) focus their attention on simulating shock waves using a viscous compressible fluid simulation, neglecting molecular vibration energy, dissociation and ionisation. Fluid equations are solved using a fractional time step along with an internal energy evolution equation derived from the First Law of Thermodynamics and an ideal gas state equation. Feldman *et al.* (2003) propose a different approach by simulating explosions with an incompressible fluid simulation grid and a particle simulation for fuel and soot. Flow divergence is positive where fluid is expanding and this value is used to adjust the pressure computation. Fuel and soot particles' temperature depends on the absorbed and emitted heat energy.

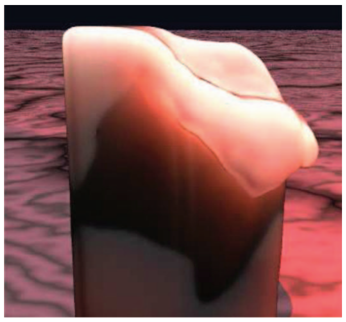


Figure 11. Detail of melting wax from Carlson, M., Mucha, P., Brooks Van Horn, III, R., Turk, R. "Melting and flowing," Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 167–174, © 2002 ACM, Inc. http://doi.acm.org/10.1145/545261.545289.

3.3. *Viscous fluid solvers*

Viscosity is usually dealt with using an implicit method to solve the diffusion term in the Navier–Stokes momentum equations. An important challenge in this context is modelling free-surface viscous fluids, such as goo. Batty and Bridson (2008) introduce a variational method to solve viscosity with boundary conditions focused on minimising the change in velocity and the rate of viscous dissipation. Other cases where viscous phenomena require special attention are melting and mixing which are discussed below.

3.3.1. Melting

In some cases, it is required to simulate and visualise melting of objects, such as wax or ice when heated. Terzopoulos *et al.* (1989) introduce a Lagrangian approach, where solid objects are represented by a lattice of particles. Fluids are represented by particles that are not included in any lattice. Material softening is simulated by modifying the elastic stiffness of inter-particle connections.

Carlson *et al.* (2002) employ user-defined fixed temperature threshold and relationship between viscosity and temperature. When the temperature of the solid is above the threshold it liquefies, otherwise it remains solid. Variable viscosity is sampled at the centre of each cell, as for pressure and these values are interpolated onto a staggered grid. The geometric mean is used for interpolation to avoid flow slowing down in low viscosity regions near high viscosity ones. Reintroduction of dampened velocity in fluid blobs formed on splashes avoids artificial reduction of the blob's velocity. An example of the results using this method is shown in Figure 11.

Losasso *et al.* (2006a) propose an alternative approach based on combining an Eulerian grid to model the fluid with a Lagrangian representation of the solid objects. Solid objects are triangular or tetrahedral meshes and a level set is constructed both for solids and liquids. Particles are created inside the solid object. As the material erodes so does the level set and particles leaving the solid are added to the particle (liquid) level set simulation, generating a layer of fluid at the solid surface.

Wei *et al.* (2003) propose an approach based on cellular automata. Here, each cell stores the matter state (solid or liquid), the amount of liquid, up to a maximum, and the accumulated energy. Heat conduction is determined from the gradient of temperature between neighbouring cells. Heat energy is accumulated on each cell until it reaches the melting point. A set of rules is defined to determine the fluid behaviour. Motion of a liquid cell due to gravity is determined by the capacity for more liquid in the cell below. Spreading patterns are determined randomly, and heat energy is dissipated due to friction. Fluid excess can be distributed upwards as well, simulating the piling of viscous fluid.

Müller *et al.* (2004b) propose a general framework for modelling elastic and plastic objects on a framework built on a SPH simulation. Zhao *et al.* (2006) model melting objects using the Lattice Boltzmann method. Here, modified update rules for particle package distributions account for phase changes and interfaces between multiple phases. A performance comparison for different approaches and hardware is shown in Table 3.

Solenthaler *et al.* (2007) model both solids and fluids using SPH. Here, elastic solids are modelled by incorporating an additional elasticity force to the particles. Modelling rigid solid behaviour is achieved by computing all the forces acting on the solid and by restricting the motion to translations and rotations.

Table 3. Performance comparison of melting simulation approaches.

Approach	Hardware	Simulation nodes	Updates per second
Carlson *et al.* (2002)	2.0 GHz Pentium 4 CPU	$35 \times 28 \times 28$ grid	2
Wei *et al.* (2003)	2.53 GHz Pentium 4 CPU	$141 \times 94 \times 141$ grid	9.5
Müller *et al.* (2004b)	2,8 GHz Pentium 4 Laptop	200 volume elements	27
Zhao *et al.* (2006)	3.0 GHz Pentium Xeon CPU	64^3 grid	0.4
	nVidia GeForce 6800 Ultra GPU		5.5

Depending on the temperature of a particle, it is considered liquid or solid. In the latter case, it can correspond to a rigid or deformable solid, where the elasticity constant and the viscosity are interpolated, allowing simulation of melting objects.

3.3.2. *Mixing*

Mixing presents another challenging application for computer animation. The goal is to properly represent and visualise mixing fluids with different densities and viscosities such as water and honey. Zhu *et al.* (2006) introduce a Lattice Boltzmann equation for mixing two miscible fluids. Park *et al.* (2008) develop a framework for simulating miscible and immiscible fluids of different densities also using the Lattice Boltzmann method. The evolution of phase separation of immiscible fluids is modelled using the Cahn–Hillard equations. Both the Navier–Stokes and the Cahn–Hilliard equations are represented using the Lattice Boltzmann method by modifying the collision operator to account for energy changes due to mixing.

Kang *et al.* (2010) introduce a method to handle both miscible and immiscible fluids on a regular Eulerian grid. Here, both density and viscosity are assumed to be variable and the individual density of each fluid in the mixture is assumed constant. It is shown that in this case, the resulting velocity field is solenoidal and the regular Navier–Stokes equations are used. Density evolution is simulated for each fluid in the mixture using volume fractions. Since the volume in each cell is constant in a regular grid, the density in a cell is defined as the sum of the densities present in the cell weighted by the volume fraction they occupy. A signed distance function $\phi(\mathbf{x})$ is constructed from volume fractions to model the interface between immiscible fluids. The Navier–Stokes equations are split in the same fashion as in Stam (1999), but the solution of these equations is slightly different in this case. Advection of the volume fractions is performed using the back and forth error compensation and correction (BFECC) method proposed in Kim *et al.* (2007), which is a more accurate advection scheme than the semi-Lagrangian method in Stam (1999). The BFECC method is based on correcting the numerical errors in an advection scheme and is used to achieve more accurate results than with semi-Lagrangian advection. For this, the quantity q in a cell (i, j, k) is advected backwards in time and then forward in time to obtain a quantity q'. Then, the error of the advection scheme is considered to be $1/2(q - q')$. The factor $1/2$ appears as there are two advection steps involved. This error is used to correct the scheme: in a semi-Lagrangian method this would be done by advecting backwards and subtracting the estimated error.

The diffusion step assumes that the volume fractions f are constant within a cell. Also the kinematic viscosity v in each cell is computed as a weighted average of the viscosities of the participating fluids, leading to the following equation:

$$\frac{\partial \mathbf{u}}{\partial t} = \nabla v(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \quad \text{where} \quad v = \sum_f \alpha^f \frac{\mu^f}{\rho^f}.$$

Here, μ corresponds to the dynamic viscosity. This equation is discretised to assemble a symmetric linear system, similar to the one obtained for diffusion in solving the Navier–Stokes equations. Miscible fluid diffusion is modelled using Fick's second law for diffusion. To avoid miscible diffusion between immiscible fluids, at each interface between immiscible fluids a boundary condition is enforced. Kang *et al.* (2010) note that for high diffusion coefficient, simulations become unstable because of explicit time integration; therefore this step is solved implicitly. To calculate the pressure projection, density is calculated from fractions of each group of miscible fluids. For example, water and ink would form one single group and oil would be in a different one. A performance comparison of these approaches is presented in Table 4.

3.4. *Multiphase fluids*

In several scenarios, different fluids in different phases interact with each other as for example in combustion, reaction of different fluids and bubbly flows. Combustion and fire form a special case that has been reviewed in Section 3.2, so that in this section we focus on liquid interaction.

3.4.1. Bubbles

Probably one of the most visually attractive cases of multiphase fluids is that of bubbles, which enhance the visualisation and simulation of liquids. Greenwood and House (2004) advance bubbles following an external flow which is not altered by the bubbles. Fluid is simulated employing a grid method and the surface is tracked using the particle level set in Enright *et al.* (2002). Particles are used to identify areas where air pockets are trapped and bubbles are formed. Bubbles are modelled as spheres whose radii follow a Gaussian distribution. To model bubble popping, it is assumed that bubbles with larger radius pop sooner than small bubbles with a given probability based on the bubble lifetime, radius and a user defined parameter. Foam is formed from bubbles that are in contact. Attraction and repulsion forces are computed as spring forces. Instead of simulating the accurate foam structure, bubbles that are attracted simply overlap. Forces acting on the bubble are due to pressure and viscosity and can be modified by the animator using user-defined constants. Friction between bubbles close to the surface is calculated from spring forces and the average velocity of surrounding bubbles as foam is formed.

Similarly, Hong *et al.* (2008) calculate the background flow using a grid with adaptive refinement and bubbles are modelled using SPH. The pressure force is computed with an SPH kernel, and bubbles are subject to a vorticity confinement force which is obtained at the mass centre of each pair of SPH particles. Lift and drag forces on bubbles are determined from the velocity field on the grid. The volume of the bubble is computed as $V_i = m_i/\rho_i$, where both the particle mass and the density at a particle's location are determined from the SPH simulation. Buoyancy is computed proportionally to the bubble's volume and attractive forces allow clustering nearby bubbles. An example of the results obtained with this technique is shown in Figure 12. Ihmsen *et al.* (2011) simulate bubbles and liquids with SPH where each phase is solved separately. Then, both phases are coupled by computing drag from buoyancy and an attraction force between particles or *cohesion force*, allowing merging of nearby particles. Bubbles are generated in surface areas with large differences in velocity.

Hong and Kim (2005) focus on computing the bubbles' pressure profiles. A bubble's surface is modelled by a level set. Across this interface, there is a discontinuous jump in pressure caused by surface tension in the bubble's surface. The ghost values technique is used to properly differentiate the pressure field. The fluid pressure is extrapolated inside the bubble and the pressure inside the bubble is extrapolated into the fluid. Pressure jumps are incorporated easily into the simulation by modifying only the right

Table 4. Performance comparison of fluid mixing approaches.

Approach	Hardware	Simulation nodes	Seconds per update
Park *et al.* (2008)	Two Xeon Node processors with four cores of 3.3 GHz	$140 \times 70 \times 80$ grid	3.3
Kang *et al.* (2010)	Intel R Core2 Quad Q8400 CPU	$100 \times 200 \times 100$ grid	180

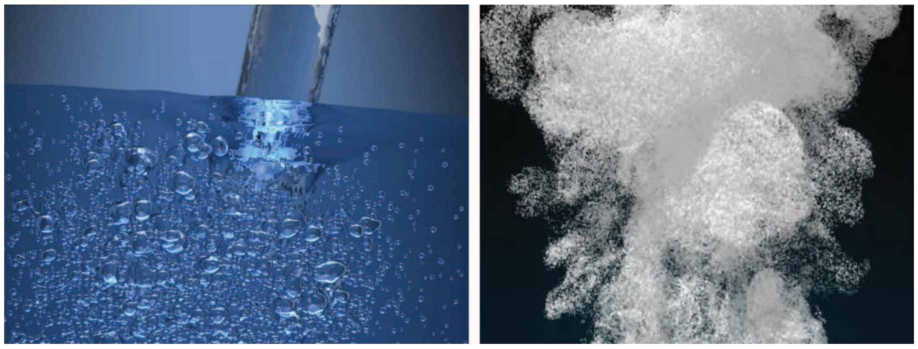


Figure 12. Bubble simulations. Left: Detail of bubbles generated by water falling on a tank from Hong, J.-M., Lee, H.-Y., Yoon, J.-C., Kim, C.-H. "Bubbles alive," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2008, Vol. 27:3, © 2008 ACM, Inc. http://doi.acm.org/10.1145/1360612.1360647. Right: Detail of 1.57 million bubbles simulated as dispersed flow from Kim, D., Song, O.-Y., Ko, H.-S. "A Practical Simulation of Dispersed Bubble Flow," ACM Transactions on Graphics (TOG) – Proceedings of SIGGRAPH 2010, Vol. 29:4, © 2010 ACM, Inc. http://doi.acm.org/10.1145/1778765.1778807.

hand side of the pressure update equation. Ghost values are used as well to compute viscous forces from velocity.

Mihalef *et al.* (2009) use a marker level set method to track the surface. Here, marker particles are seeded on the surface only and are used to indicate where the level set suffers from excessive smoothing. Droplets and bubbles are generated in cells that (a) are not at the liquid surface and (b) are located in high curvature areas. These are filtered with a threshold value relating to the local mean curvature and the relative velocity of gas and liquid. Bubbles are considered to be spherical and their velocity is calculated as the sum of the background flow and the Stoke's law value for particles falling in a viscous fluid. Droplet velocities are calculated from Newton's law adding a drag force of the form $\mathbf{f} = -\alpha(\mathbf{u} - \mathbf{U})^{\beta}$. Here, $\mathbf{U}$ is the air velocity, α is a simulation parameter and $\beta = 1.0$ for Reynolds numbers smaller than 1, and $\beta = 1.28$ for high Reynolds numbers. This provides a unified framework for handling bubbles and droplets product of liquid interaction.

Kim *et al.* (2010) model bubbles as a dispersed flow. Previous approaches work well for a small amount of bubbles, but scale poorly as the number of bubbles increase. Instead of computing values for each bubble interface, volume fraction information is used. This corresponds to the spatial average of each fluid, using level set and bubble particle information. A *fraction field* is constructed from the level set and is used for computing the density required in the pressure projection step from the gas and liquid densities. Random motion of millions of bubbles, affected by the wake that each one generates as they move is performed through discrete random walks. Particles have a defined probability of breaking up. If this is the case, two particles of half the volume are created, and scattered according to the Schlick's phase function in Blasi *et al.* (1993). Results from this technique are shown in Figure 12.

3.4.2. *General immiscible fluids*

In general, modelling non-miscible fluids requires paying special attention to properly tracking and evolving the fluids interfaces. Losasso *et al.* (2006b) extend the level set method to model the interface between several different fluids. Different level sets are used for each fluid. A geometric projection method is introduced to correct overlapping and empty spaces, based on evaluating whether points are submerged in more than one fluid or in a void space. If the case is a point submerged in multiple fluids, it is considered to be in the fluid in which the point is immersed the deepest. If a point is in the void, it is assumed to be in the fluid whose level set is the closest. This is done by evaluating the value of level sets for different fluids at specific locations and correcting inconsistencies by subtracting to each the average of both level sets values.

This approach for correcting multiple level sets is combined with level set particles in order to obtain more accurate boundaries. Each fluid is modelled using a standard Eulerian grid solver. The Poisson problem for pressure is modified to account for different fluids and surface tension which induces jumps in pressure across interfaces in a similar way to the approach of Hong and Kim (2005). An example of the results obtained with this technique is shown in Figure 13.

Kang *et al.* (2010) use a similar approach in a more general framework to handle both miscible and immiscible fluids, however they combine the level set method with a volume fraction approach to enforce incompressibility. Bao *et al.* (2010) employ volume fraction methods to determine interfaces between different fluids in an *onion-skin* model. Adaptive mesh refinement for enhancing visual details is used at interface locations.

Multiphase flow has been simulated using SPH as well. Solenthaler and Pajarola (2008) handle density discontinuities at multiple fluid surfaces using a particle density measure rather than the fluid density encoded in particles and equations for pressure and viscous forces are modified accordingly. This avoids velocity field artefacts and numerical instabilities in traditional SPH simulations of multiple fluids.

3.5. *Turbulence and detail preservation*

Physically correct simulation of turbulence is a challenging problem in computational fluids dynamics

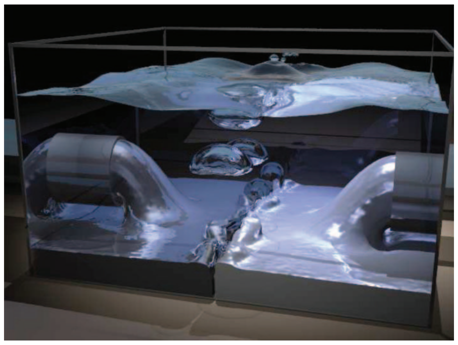


Figure 13. Details of multiphase liquid simulation. Four immiscible fluids interacting: two being poured inside a third and bubbles are generated as resulting reaction, from Losasso, F., Shinar, T., Selle, A., Fedkiw, R. "Multiple interacting liquids". ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2006, 25:3, © 2006 ACM, Inc. http://doi.acm.org/10.1145/1141911.1141960.

as well as in computer animation. In the following, we review some of the most important advances in this area.

3.5.1. *Vorticity confinement*

An important drawback of grid-based methods is the loss of fine detail due to grid resolution. Also, as semi-Lagrangian methods average velocity values, an artificial viscous effect due to numerical dissipation appears in simulations. This becomes a problem in simulations of fluids that naturally exhibit a high degree of detail and fine vortices, such as smoke. Fedkiw *et al.* (2001) introduce the *vorticity confinement* technique to address this problem without introducing non-physical methods to reproduce details such as adding random noise. Small scale structure is assumed to be due to vorticity ω. Lost vorticity due to numerical dissipation is reintroduced to the fluid as an additional force:

$$\mathbf{f} = \varepsilon h(\mathbf{N} \times \omega)$$

where $\varepsilon > 0$ is an animation parameter that controls the amount of vorticity force re-introduced and h is the size of a grid cell. Here, $\mathbf{N}$ is a normalised vorticity location vector defined to point to high vorticity regions:

$$\mathbf{N} = \frac{\nabla||\omega||}{||\nabla||\omega||||}.$$

An example of swirling smoke produced with this technique is shown in Figure 7. Nguyen *et al.* (2002) use this technique for fire simulation and Losasso *et al.* (2004) for simulations with adaptive meshes. Selle *et al.* (2005) use vorticity confinement to add sub-grid detail obtained from a vortex particle simulation running parallel to a grid simulation. Hong *et al.* (2008) employ vorticity confinement to achieve realistic bubble motion in an SPH simulation.

3.5.2. *Vortex methods*

Vortex methods have been used in computer animation due to their ability to model complex flow with very sparse data and minimal numerical dissipation. Since Lagrangian particle methods achieve better performance in fluid simulations, modelling fluids using vortex methods in a Lagrangian framework is an attractive alternative. Gamito *et al.* (1995) introduce vortex methods for the first time in computer animation for 2D fluid simulation. However, it is only recently that these methods have been applied to high quality 3D smoke animation.

The key idea in vortex methods is to model a flow from its vorticity. The vorticity evolution equation is obtained by applying the curl operator to the Navier–Stokes equations, which eliminates the pressure term for constant density flows and divides the advection term into a vorticity advection term and a vortex stretching term. To solve the former, the velocity field has to be recovered from the vorticity field on the whole fluid region. This is done using the Biot–Savart law.

Park and Kim (2005) model vorticity with vortex blobs, whereas Angelidis *et al.* (2006) and Weißmann and Pinkall (2010) employ closed filaments. This kind of discrete structures simplifies the evaluation of the Biot–Savart law as the integral over the whole fluid domain is evaluated only on particles, or along curves.

Since the Biot–Savart formula is singular at filament or particle locations, alternate kernels are used. Angelidis *et al.* (2006) employ a fourth order polynomial kernel with limited support. Weißmann and Pinkall (2010) apply a smoothed convolution of the Biot–Savart formula to both vortex filaments and passive smoke particles. This is shown to be equivalent to a global advection step using the original formulation of the Biot–Savart law.

Park and Kim (2005) compute the vorticity stretching term by explicit numerical evaluation. Angelidis *et al.* (2006) invoke Kelvin's theorem to account for the vortex stretching term by modifying the magnitude of vorticity as the filament enlarges and therefore keeping circulation constant. This approach has been used successfully to represent high quality smoke.

Weißmann and Pinkall (2010) reproduce smoke behaviour around an object using vortex rings for smoke simulation and a vortex sheet to simulate solid surface vorticity. This vorticity is released into the main flow as filaments to produce vortex shedding. To avoid performance degradation due to a large number of filaments, vortex filaments are reconnected based on an energy functional minimisation. In this way, the amount of filaments in the simulation is kept low. A performance comparison of the approaches in Angelidis *et al.* (2006) and Weißmann and Pinkall (2010) is presented in Table 5.

Selle *et al.* (2005) propose a hybrid approach using Eulerian grids and vortex particles. A background velocity field is computed on an Eulerian grid. This is used in a vortex particle simulation for advecting and stretching vortex particles. To avoid numerical instability, the vortex stretching term is normalised. The vorticity of each particle is reintroduced in the grid by performing two steps. First, vorticity is interpolated at grid locations using a compact-support Gaussian kernel. Second, vorticity confinement is used to update

Table 5. Performance comparison of vortex methods' approaches.

Approach	Hardware	Simulation nodes	Seconds per update
Angelidis *et al.* (2006)	Pentium 4 2.4 GHz CPU; 256 MB RAM	73,357 smoke particles.	4
Weißmann and Pinkall (2010)	GeForce 8800 Ultra GPU	Approx. 1 M smoke particles.	0.25

the grid velocity with the vorticity from the particles. Here, it is noted that adding vortex particles added less than a 5% of computational time, making this a suitable technique to increase visual realism with very little additional computational cost. An example of the results obtained with this technique is shown in Figure 14.

Elcott *et al.* (2007) present a different approach to solve the vorticity transport equation. Here, vorticity is solved on an unstructured tetrahedral mesh, where velocity, divergence and vorticity spin are treated as fluxes, which are computed for each tetrahedron. Physical quantities are represented as integral values that are stored at vertices, edges, triangles and tetrahedra.

More recently, Lagrangian vortex methods have been employed to model hot buoyant smoke. Pfaff *et al.* (2012) achieve this by incorporating a baroclinic term in the flow equations and simulating the interface between the hot gas and surrounding fluid using a vortex sheet. Kim *et al.* (2012) simulate the creation of vortex particles in regions of high baroclinicity, introducing turbulence in a background flow.

Besides obtaining visually realistic representations of flows, computer graphics techniques have been applied to produce flow visualisations for analysis in engineering. For example, Sadlo *et al.* (2006) present vortical flow visualisation methods based on sampling fluid properties along flow pathlines.

3.5.3. Procedural turbulence

Different methods have been employed to add turbulence to simulation to enhance its visual appeal. Stam (2003) simulate a background flow to which a small scale velocity features are added. Small scale details are random vectors generated using Fourier synthesis. Turbulence is controlled by the animator by modifying an energy equation based on Kolmogorov's energy spectrum. Neyret (2003) employ advection and an energy cascade model to generate high-resolution turbulent visual representations of flow motion as textures, which are used to enhance detail of visual surfaces. Kim *et al.* (2008) generate high-resolution flows employing a low resolution grid simulation, and adding high turbulence detail, which is generated using *wavelet noise*. This corresponds to a noise function defined for a limited band of frequencies. This noise is added to the main flow according to an energy spectrum, which is obtained by computing the flow kinetic energy for different spectral bands. A user can control turbulent motion by defining the range of frequencies on which noise is added.

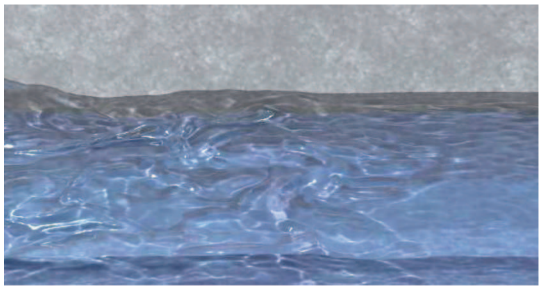


Figure 14. Example of water simulation using a hybrid approach with a grid and vortex particle simulation in Selle *et al.* (2005). Vortex particles are added to the left of the simulation space to display the difference in the liquid surface shape. Image from Selle A., Rasmussen N., Fedkiw R. "A vortex particle method for smoke, water and explosions," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2005, Vol. 24:3, © 2005 ACM, Inc. http://doi.acm.org/10.1145/1073204.1073282.

Yoon *et al.* (2009) propose a different way to use particle methods for turbulence simulation. Here, a background flow is generated using the approach in Stam (1999) using a coarse grid. From this coarse grid, a high resolution grid is obtained via interpolation which is combined using vorticity confinement with a vortex particle simulation. Then, the high resolution vorticity grid is blended with the low resolution grid with the background flow for increased detail.

Narain *et al.* (2008) simulate a background flow on an Eulerian grid and turbulent energy evolution and cascading are simulated. The energy cascade is simulated from the energy distribution at different scales in the Kolmogorov spectrum. This energy is used to generate *curl noise* particles. Curl noise corresponds to the addition of irregular flow patterns using a Perlin noise function as in Bridson *et al.* (2007). An irregular spinning flow is obtained by applying curl to a Perlin noise vector. This flow then is added to a background flow to simulate turbulence. Narain *et al.* (2008) produce highly detailed turbulent flow by simulating energy and transferring it as noise particles.

This turbulent flow is produced at a much lower cost than using an equivalent fine grid. The turbulent energy distribution for a specific wave number k is assumed to be $E(k) \propto k^{-5/3}$.

To facilitate simulation, it is assumed that turbulence is isotropic and eddies are not correlated to the background flow. Wave numbers k_i are defined as the inverse of the wave length, i.e. $k_i = 1/l_i$ and at each octave of the Kolmogorov spectrum, the wave length is half the wave length in the previous octave, i.e. $l_i = l_{i-1}/2$. The wave number k_0 is a user defined parameter.

Energy evolution is modelled based on an energy creation term G, an energy transfer rate to higher octaves Π and a diffusion term D as follows:

$$\frac{\partial E_0}{\partial t} + \mathbf{u} \cdot \nabla E_0 = G(E_0, \mathbf{u}) - \Pi(E_0) + D(E_0).$$

Turbulence is first produced at the scale of the background flow, which determines the *inertial scale* and then it is transferred to subsequent scales. Eddy creation is assumed to be due to a viscous effect that turbulence has on the main flow. The rate of turbulent energy production at the inertial scale used is:

$$G(E_0, \mathbf{u}) = \frac{E_0^{1/2}}{k_0} \sum_i \sum_j \left(\frac{\partial \mathbf{u}_i}{\partial x_j} + \frac{\partial \mathbf{u}_j}{\partial x_i} \right)^2.$$

Here, sub-indices i and j indicate the ith and jth components of velocity, respectively. The factor $E_0^{1/2}/k_0$ corresponds to the eddy viscosity number and it determines the rate of eddy creation.

Energy is then transferred to smaller scales. The rate of this energy transfer from the inertial scale to higher ones is modelled by the cascade: $\Pi(E_0) = k_0\sqrt{E_0}^3$. Energy is dissipated at the highest scales, which is simulated by simply stopping the spectrum transfer for a certain scale k_ν. Here, the energy is simply dissipated and the energy at any scale above k_ν is assumed to be zero. This cut-off scale is defined as $k_\nu = \left(\Pi(E_0)/\nu^3\right)^{1/4}$.

Energy is also spatially transferred to neighbouring cells as a diffusion process. This diffusion process is modelled as:

$$D(E_0) = \nabla \cdot (l_0 \sqrt{E_0} \nabla E_0).$$

Here, the advection term is performed with the same scheme as the fluid solver for the background flow. Once the energy at each level is determined, turbulence is introduced through noise particles using a pre-computed 3D Perlin noise vector for each octave. An example of simulation results is shown in Figure 15.

Pfaff *et al.* (2009) focus on vorticity creation at solid boundaries. Vorticity in the boundary layer is estimated from the universal law of the wall and is assumed to depend on the velocity and material constant only: $\omega = \beta(\mathbf{u}_s \times \mathbf{n})$, where β is a user-defined parameter accounting for both skin friction and fluid viscosity and $\mathbf{u}_s$ is the tangential component of the flow velocity at the solid surface. This boundary vorticity or *artificial boundary layer* is used to generate particles to model turbulent flow. The vorticity at the boundary layer is pre-computed by time-averaging the tangential component of the flow velocity around a solid for different flow directions. Run-time turbulence synthesis is performed by adding vortex particles controlled by an energy model similar to the one in Narain *et al.* (2008).

To determine the areas of vorticity creation, the artificial boundary layer data is used. The anisotropic part of the Reynolds stress tensor is used as an

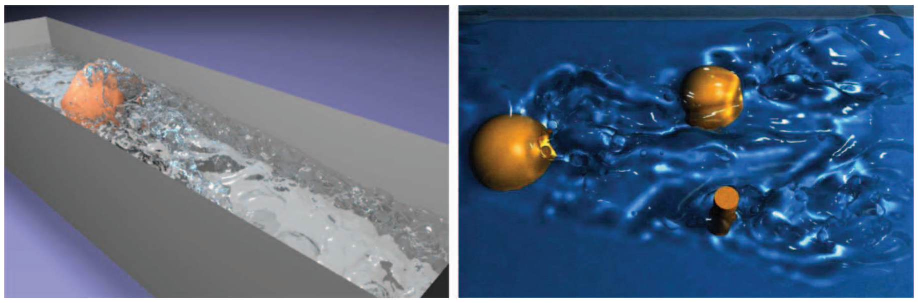


Figure 15. Left: Results obtained with procedural turbulence from Narain, R., Sewall, J., Carlson, M., Lin, M. C. "Fast Animation of Turbulence using transport and procedural synthesis," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH Asia 2008, Vol. 27:5, © 2008 ACM, Inc. http://doi.acm.org/10.1145/1409060.1409119. Right: Detail of results obtained by synthetic turbulence from Pfaff N., Thürey A., Selle A., Gross M. "Synthetic Turbulence using Artificial Boundary Layers," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH Asia 2009, Vol. 28:5, © 2009 ACM, Inc. http://doi.acm.org/10.1145/1618452.1618467.

indicator of areas where vorticity should be created. The turbulent-viscosity hypothesis is adopted to compute the Reynolds stress tensor by assuming that the tensor depends on a turbulent viscosity term and the strain rate. By applying a series of simplifications, the magnitude of the stress tensor is approximated as: $||a_{ij}|| \approx 2l_m^2||\omega||^2$ using a *mixing length* (l_m) model. Vorticity is evaluated using the artificial boundary layer information found in the pre-computation step. Vorticity is then created in run-time with the following probability density:

$$p = c\Delta t \frac{||a_{ij}||}{||\mathbf{u}||^2}.$$

Here, c is a constant to control vorticity creation. Vortex particle dynamics is governed by a vorticity transport equation. Particles are assigned a radius, which is modified as part of the energy transfer process. Particles with an associated wave number in the inertial scale are decayed into several particles with smaller wave numbers. In the model-dependent range, aligned particles that are closer than a radius distance are merged into a stronger particle with largest radius. Particle dissipation is achieved by eliminating particles. An example of simulation results is shown in Figure 15.

Zhao *et al.* (2010) pre-compute different random force fields in the Fourier domain following the Kolmogorov spectrum. One of these force fields is randomly selected and added to a grid simulation as an external force. This can be done directly on the simulation grid, or on a refined grid obtained by interpolation from the original simulation, or on a SPH particle simulation. This is called *random forcing* and its purpose is twofold: add random, chaotic turbulent behaviour and reduce computational time by not synthesising turbulence on run-time. Performance comparison for procedural turbulence methods is shown in Table 6.

3.6. Solid–fluid coupling

In many scenarios in computer animation, fluids interact with solids. Simulation of this phenomenon is a key component to achieve visual realism in graphical fluid simulations. One-way coupling of solids and fluids can be performed in two cases: solid-to-fluid and fluid-to-solid coupling. In solid-to-fluid coupling, the solid object acts as a boundary condition for the fluid, but is unaffected by the fluid motion. Here, the solid is static or it has prescribed motion. Such is the case of water simulations in Foster and Fedkiw (2001) and Enright *et al.* (2002). In fluid-to-solid coupling, the solid takes the velocity from the flow but it does not affect the fluid motion. Such is the case of a particle that is passively advected in a flow which can be used to model a very light object being dragged in the flow as in Foster and Metaxas (1996).

In two-way solid fluid coupling, both solid and fluid are affected by each other's motion. In the following, we review the main techniques used in computer animation to simulate two-way solid fluid coupling.

3.6.1. *Two-way solid–fluid coupling for computer graphics applications*

The simplest case of this kind of coupling as described in Bridson (2008) is called weak coupling. Here, the coupling process is performed in two stages: first the flow is solved applying solid–fluid boundary

Table 6. Performance comparison simulations for procedural turbulence simulations.

Approach	Hardware	Simulation nodes	Seconds per update
Yoon *et al.* (2009) full simulation	Intel Quad Core 2.4 GHz CPU; 2 GB RAM	$120 \times 360 \times 120$ grid	59
Yoon *et al.* (2009) combined simulation		$30 \times 90 \times 30$ background grid; $120 \times 360 \times 120$ grid from particles	9.2
Narain *et al.* (2008)	Intel Xeon 2.8 GHz CPU; 8 GB RAM	$128 \times 32 \times 32$ grid	45
Pfaff *et al.* (2009)[a]	3.0 GHz Intel Core i7 CPU	$100 \times 25 \times 60$ grid	10
Zhao *et al.* (2010) full simulation	Intel Core 2 6300 1.86 GHz, 4 GB RAM	$64 \times 32 \times 50$ grid	1.5
Zhao *et al.* (2010) combined simulation		$32 \times 16 \times 25$ background grid; $64 \times 32 \times 50$ grid for turbulence	0.052
Zhao *et al.* (2010) SPH simulation		4096 particles	0.019

[a]This employs a pre-computation step requiring 59 s per update per database parameter.

conditions, obtaining both the fluid pressure and the shear stress tensor. Both are used to compute pressure and viscous drag on the solid object.

Carlson *et al.* (2004) propose a different approach where solid objects are modelled as a fluid with a higher density in an Eulerian grid. This is referred by some authors as an *immersed boundary method*. Both solid and fluid are simulated using the Navier–Stokes equations for motion. The solid object rigidity is enforced by imposing a zero rate of strain in the solid object cells. Most of the computational power required to perform solid–fluid coupling in this approach is spent solving the fluid equations and the level set for liquid surfaces.

Klingner *et al.* (2006) adopt a similar approach with tetrahedral meshes where both solid and fluid accelerations are computed as a coupled system in a single pressure projection step. Chentanez *et al.* (2006) present an approach where the motion of a Lagrangian solid and a fluid are solved simultaneously by solving a single system of equations for the fluid pressure projection and the solid velocity. Batty *et al.* (2007) further develop the idea of solving both fluid and solid object velocities simultaneously adopting a variational approach to solve both solid and fluid velocities in a single pressure update step.

Robinson-Mosher *et al.* (2008) further enhance the approach in Batty *et al.* (2007) by incorporating interaction between fluids and deformable objects and thin shells. Here, the transfer of momentum between the solid object and the fluid is modelled explicitly. As thin objects do not conform to the grid, fluid mass and momentum is mapped to the solid surface. The conservation of momentum is enforced at the solid surface and the results are interpolated back to the grid. An example of the results obtained with this technique is shown in Figure 16.

Guendelman *et al.* (2005) propose a method to simulate the interaction of liquids and infinitesimally thin objects such as cloth. When interpolating and calculating differentials, leakage is avoided by tracing a ray from a fluid location to the point of interest (e.g. backtraced particle position for semi-Lagrangian advection), and determining whether both points are separated by a thin shell. If so, interpolation and differential expressions are modified to account for the solid presence. Pressure differences in the fluid are transferred to the solid locations to compute solid–fluid coupling.

Kwatra *et al.* (2010) present a method to simulate swimming characters. Here, a character is represented as an articulated body follows motion recorded in advance. In this method, the force the character requires to match the pre-recorded motion is computed considering a two-way coupled liquid simulation.

Müller *et al.* (2004a) perform solid fluid coupling in an SPH simulation. Coupling particles and solids is reduced to simple particle coupling by creating boundary particles at the solid surface. Then, the influence of the fluid on the solid is computed from the surface particles.

Akinci *et al.* (2012) also employ boundary particles and compute their relative contributions to account for irregular particle distributions. Here, simulations are also enhanced by incorporating friction and drag at the solid surface.

3.7. Fluid control

One of the most challenging aspects of fluid simulation is allowing animators to control the simulation to achieve a specific visual effect. Such is the case of smoke or fire adopting a desired shape. A limited level of control can be achieved by using an external wind

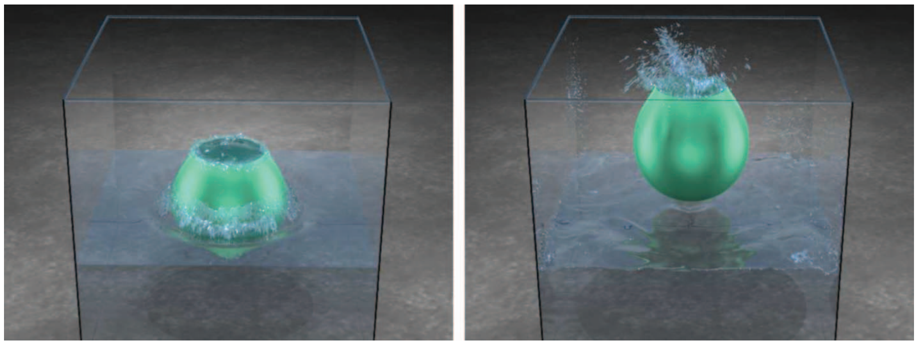


Figure 16. Example of coupling fluids and solids in Robinson-Mosher *et al.* (2008). A thin shell initially submerged is pulled out of water. As the bag rises, it contracts and expands inducing additional motion in the contained liquid. Images from Robinson-Mosher, A., Shinar, T., Gretarsson, J., Su, J., and Fedkiw. "Two-way coupling of fluids to rigid and deformable solids and shells," ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2008, Vol. 27:3, © 2008 ACM, Inc. http://doi.acm.org/10.1145/1360612.1360645.

velocity field as in Beaudoin *et al.* (2001) for fire, and defining the trajectory axis of a series of vortex rings as in Angelidis *et al.* (2006).

Treuille *et al.* (2003) propose a more complex approach for control of smoke simulations using *keyframes*. These are specific simulation states identified by a user-defined smoke density distribution and velocity field. Fluid evolves in time to match keyframes by solving an optimisation problem whose solution defines a wind force that drives the simulation to the defined keyframes. McNamara *et al.* (2004) further develop this approach to allow liquid control as well by incorporating sources and scalable gradient calculation. Fattal and Lischinski (2004) propose a method to evolve smoke density to a predefined configuration using a modified flow equation instead of an optimisation framework. The flow equation incorporates an additional driving force which controls the transport of smoke density to a specific configuration, and a smoke gathering term to reduce the effects of dissipation.

Thürey *et al.* (2006) propose a different approach, where a multi-scale approach is adopted using the Lattice Boltzmann method. A coarse-level flow field is obtained by removing high frequencies in from the velocity field. Control forces are added to this coarse velocity field using particles, minimally affecting fine details. More recently, Nielsen and Bridson (2011) control liquid simulations by guiding the simulation using an auxiliary shape which separates the region where high resolution is required to model the liquid surface from the inner part of the liquid where coarser resolutions can be used.

4. Additional considerations

4.1. Rendering

In the previous sections, we have discussed different methods to simulate fluids for computer animation. Once simulation data is obtained, either as an off-line process or in real time, a 2D visual representation of the simulation data and other objects in the simulation space is required to be displayed to the user. This process is known as *rendering*. This is achieved by modelling the behaviour of light considering phenomena as reflection and refraction. This corresponds to calculating the amount of light that is directed from a point in the simulation space based on the total emitted light and the incident light on that point on the simulation space. In many cases in fluid simulation, it is required to model the behaviour of light when travelling through a volume of fluid. Therefore, it is necessary to model the extinction, scattering and absorption of luminous energy. The light behaviour in this case is modelled based on radiative transfer.

Different solutions to determine the amount of light that reaches the observer from the simulation model objects have been proposed and they can be seen as ways to approximate the solution of the rendering equation. While this subject is beyond the scope of this paper, we briefly list the principal techniques of *ray casting, ray tracing, surface reconstruction, slicing*, texture *mapping* and *splatting* and we refer the reader to the corresponding references for more details.

Ray casting is a computationally inexpensive technique where a ray is cast towards the simulation for each pixel in a 2D image. The colour of the pixel is determined by an object that the ray's trajectory intersects. Inacio *et al.* (2010) and Crane *et al.* (2007) employ this method to render liquid surfaces and gases. Fraedrich *et al.* (2010) adaptively sample particle simulation data onto a view aligned grid.

In *Ray tracing*, more complex ray trajectories are computed by allowing reflection, refraction and scattering. Nguyen *et al.* (2002) employ this method for rendering smoke and Fedkiw *et al.* (2001), Foster and Fedkiw (2001), Enright *et al.* (2002), Frisvad *et al.* (2007) and Harada *et al.* (2007) for liquids. Carlson *et al.* (2002) and Hong *et al.* (2008) apply this method for rendering melting wax and bubbles, respectively. Ray tracing yields more appealing results than ray casting, but at a greater computational cost.

Liquid surfaces can be reconstructed by traversing a volume and defining polygonal patches forming a boundary as in Lorensen and Cline (1987). Kang *et al.* (2010) and Müller *et al.* (2003) apply this method for rendering liquid surfaces and Beaudoin *et al.* (2001) for flame layers. This method is computationally expensive and it is used mostly for high-quality rendering. Yu and Turk (2010) reconstruct smooth surfaces from particle-based fluid simulations by representing particles with anisotropic smoothing kernels. The anisotropy scale is computed from the distribution of nearby particles and the analysis of its covariance tensor. A smooth surface is then reconstructed from the isosurface of a density field determined by these kernels.

A volume can be intersected by several planes to obtain *slices* which are rendered to the user. Fuller *et al.* (2007) and Horvath and Geiger (2009) use this technique for rendering fire.

Mapping a texture to an object's surface can drastically change the visual appearance of an object. Wang *et al.* (2003) employ this method to simulate the ocean's appearance. Projecting and blending the contribution of each volumetric simulation element into a 2D plane is known as *splatting*. Wei *et al.* (2002) and Müller *et al.* (2003) use this technique in fire and liquid simulations, respectively.

4.2. Parallel flow computation

Parallelisation has been used to simulate large scale simulations in computer animation, mostly in the film industry, by performing computations on farms of workstations. The high cost of this kind of implementation makes it impractical for most applications.

With increasingly available multi-core CPUs, one possibility to speed-up fluid simulations is to take advantage of this feature, distributing computation load on each core. Ihmsen *et al.* (2011) adopt this approach by employing optimised data structures that allow efficient simulations with 12 million particles.

A low cost alternative is to transfer computations to the GPU, due to their increasing power even in commodity hardware, reducing the overhead from the CPU. Operations in GPU are efficiently processed in parallel, for instance by processing simultaneously several pixels in one pass. The capabilities of the GPU have led researchers to transfer costly computations to the GPU by representing the flow properties as graphical elements. Early approaches for physical simulation on the GPU use graphics primitives, particularly textures. A texture is a 2D bitmap image. In traditional graphics applications, a texture is mapped to an object to give it a specific appearance. The colour information in the texture is numerical and can be used as an array to store simulation data as proposed in Wu *et al.* (2004). Graphical processing unit programming supports 32 bits floating point precision, allowing to take advantage of the parallel processing pipeline of the GPU for general computations.

A typical GPU program used to simulate fluids receives a texture storing numerical simulation data. Each pixel corresponds to a grid cell in the simulation. The GPU processes each pixel in parallel and another texture is produced as output. This is directly applied to solve flow velocity using the splitting method proposed in Stam (1999), where the result of each simulation step is stored in a new texture. Passively advected quantities like smoke density or temperature are stored in textures as well. Liu *et al.* (2004) and Crane *et al.* (2007) solve 3D simulations using a 3D texture, which stores volumetric data instead of pixel information. Wei *et al.* (2004b) stores each slice of a 3D texture as a 2D texture and process the flow using a set of 2D textures. Particle simulations can also be performed on a GPU by storing the particles positions and velocities in floating point textures.

Numerical methods can be implemented in the GPU through a piece of code called fragment shader. This program is loaded in the GPU and defines the changes of colour in a texture pixel, which in the case of simulation, encodes velocity, density or pressure values. Wu *et al.* (2004) and Liu *et al.* (2004) employ this technique for simulating smoke; Harris *et al.* (2003) for simulating clouds, and Yang *et al.* (2009) for fast fluid force computations on solids.

One simulation technique that has been successfully implemented in GPU is the Lattice Boltzmann method, which is discussed in Section 2.2. The local nature of the computations in each grid cell allows taking advantage of the parallel capabilities of the GPU. A performance comparison of simulations in GPU is presented in Table 7.

The only limitation of this kind of parallelisation improvement is the simulation size, as memory in graphics processing units is much more reduced than the memory for CPU use. As new technology is developed, this constraint may not be a problem for real-time applications in the near future.

Graphical processing units can also be employed for general computation (GPGPU) using different frameworks, such as the Compute Unified Device Architecture (CUDA). CUDA implements programming structures that are similar to traditional languages such as Fortran and C/C++, and provides easy data transfer between the CPU and the GPU, while employing the full computational power of the GPU. Amador and Gomes (2010) employ CUDA grid simulations; Zhang *et al.* (2011) employ multiple processing GPUs for SPH simulations, and Chentanez and Müller (2010) implement hybrid grid and particle simulations using this framework. A summary of performance results using CUDA GPU programming approaches is presented in Table 8.

4.3. Advanced human–computer interfaces

As virtual applications increase in complexity, new human–computer interfaces are developed to provide an enhanced user experience while interacting with a

Table 7. Performance comparison of GPU simulations.

Approach	Hardware	Simulation nodes	Updates per second
Wu *et al.* (2004) CPU simulation[a]	Pentium 2.8 GHz CPU	512^2 grid	1.39
Wu *et al.* (2004) GPU simulation[a]	GeForce FX5950 Ultra GPU		20
Wei *et al.* (2004b) CPU simulation	2.53 GHz Intel Pentium 4 CPU	$80 \times 40 \times 40$ grid	2.8
Wei *et al.* (2004b) GPU simulation	nVidia GeForce FX5900 Ultra GPU		11.5

[a]Simulation without diffusion step.

Table 8. Performance comparison of simulation using CUDA.

Approach	Hardware	Simulation nodes	Seconds per update
Amador and Gomes (2010) CPU simulation	Intel Core2 Quad CPU Q6600@2.40 GHz	128^3 grid	1.611
Amador and Gomes (2010) GPU simulation	NVIDIA GeForce 8800 GT Q6600@2.40 GHz		0.164
Chentanez and Müller (2010) CPU simulation	Intel Core i7, 2.67 GHz	900×135 grid	0.0945
Chentanez and Müller (2010) GPU simulation	NVIDIA GTX480	250 K particles	0.018

virtual environment. One recent development in this area is haptics, which are human–computer interfaces that stimulate the sense of touch allowing the user to feel objects in a virtual environment. One way to achieve this is through generating force feedback to the user through the haptic device.

Current methods for haptic interaction with fluids are based on point-based haptic interfaces, i.e. they simulate interacting with objects with the tip of a stick. The haptic device tracks the position in real space of a point, the *end-effector*, and maps it to the virtual space, into the *haptic interface point* (HIP) position.

This method has been applied for interacting with virtual rigid solids. Each simulation step consists of mapping the end-effector's position to the HIP position, computing the haptic forces and then rendering the force feedback to the user. This process has to be performed at a rate of 1 KHz, otherwise the motion is not smooth, but perceived as a series of discontinuous jumps. Graphic and haptic representations of the fluid are computed separately at different update rates in the processor. To simulate a solid object interacting with a fluid, a virtual solid object is created and the net force and torque acting on it are rendered to the user. Haptic force constraints represent a major challenge for fluid simulation in virtual environments, due to the high update rate required. Moreover, the most precise force calculations are obtained from grid simulations, which have a low update rate. Despite of this difficulty, grid methods have been applied to haptics due to the accuracy of force computations, and different methods to obtain a suitable update rate are applied.

Baxter and Lin (2004) present the first haptic fluid simulation for a virtual painting application. To achieve 1 KHz update rate, forces computed at approximately 30 Hz are used as input for a finite impulse response filter which returns a smooth force value considering the 10 previous samples.

Dobashi *et al.* (2006) present a virtual river paddling application with haptic force feedback for irrotational motion. They simulate the liquid surface using the shallow water equations in Section 3.1.2. It is assumed that the relevant forces for haptic force feedback on the solids are: buoyancy, frictional resistance of the water and the pressure difference between the front and the back of the object. Both buoyancy and pressures are approximated from hydrostatic pressure values where the depth of a surface point is measured taking into account the wave height. Frictional resistance and pressure due to motion are approximated from a database of pre-computed velocity values. This database is obtained by simulating the flow around the object for different flow directions. Through the use of a pre-computed database of velocities for run-time computations, real-time update rates for haptic rendering are achieved.

Vines *et al.* (2009) simulate forces from interaction with a 3D pool of liquid and fluid forces using a network of springs, whose motion and forces can be computed in real-time. Yang *et al.* (2009) perform 3D fluid force computations on GPU, by modelling solid and fluid properties as textures and computing the stress tensor in graphics hardware. Although real-time frame rates of 20 updates per second are obtained in this approach, these are not enough to produce smooth haptic force feedback. Here, the force results are filtered to produce smooth motion.

5. Conclusion

The article presents an overview of the main techniques used to simulate fluids in computer animation, where emphasis is placed on visual plausibility and rapid computation for strict update requirements rather than on technical accuracy, yielding a unique blend of tried and true computational fluid dynamics (CFD) techniques and clever shortcuts. From the complexity of turbulence and infinite and seemingly random sea surface simulations to user-controlled smoke, fire and phase-change models, there is ample room for CFD experts to lend a hand to computer animators in developing efficient solvers for a host of computer animation applications, be they for the entertainment industry or training purposes. This is still an area with a great potential for research where new developments are made every year. While the continuous development in hardware and numerical methods brings fluid simulation in computer animation closer to the performance and visual quality required by many applications, we encourage further interdisciplinary collaboration to enrich this area of research.

Acknowledgements

We would like to express our gratitude to the reviewers for their valuable feedback and to Dr. Jochen Lang from the University of Ottawa for his comments on this article.

References

Adams, B., *et al.*, 2007. Adaptively sampled particle fluids. *ACM Transactions on Graphics*, 26 (3), 48:1–48:8.

Akinci, N., *et al.*, 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics*, 31 (4), 62:1–62:8.

Amador, G., and Gomes, A., 2010. A CUDA-based implementation of stable fluids in 3D with internal and moving boundaries. *In: Proceedings of the 2010 international conference on computational science and its applications*, ICCSA'10. Washington DC: IEEE Computer Society, 118–128.

Angelidis, A., *et al.*, 2006. A controllable, fast and stable basis for vortex based smoke simulation. *In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '06, 2–4 September, Vienna, Austria. Aire-la-Ville, Switzerland: Eurographics Association, 25–32.

Bao, K., *et al.*, 2010. Volume fraction based miscible and immiscible fluid animation. *Computer Animation & Virtual Worlds*, 21, 401–410.

Bargteil, A.W., *et al.*, 2006. A semi-Lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 25 (1), 19–38.

Batty, C. and Bridson, R., 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. *In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '08, 7–9 July, Dublin, Ireland. Aire-la-Ville, Switzerland: Eurographics Association, 219–228.

Batty, C., Xenos, S., and Houston, B., 2010. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum*, 29 (2), 695–704.

Batty, C., *et al.*, R., 2007. A fast variational framework for accurate solid–fluid coupling. *ACM Transactions on Graphics*, 26 (3), 100:1–100:8.

Baxter, W. and Lin, M.C., 2004. Haptic interaction with fluid media. *In: Proceedings of graphics interface 2004*, GI '04, London, Canada: Canadian Human–Computer Communications Society, 81–88.

Beaudoin, P., Paquet, S., and Poulin, P., 2001. Realistic and controllable fire simulation. *In: Graphics Interface 2001*, GRIN'01, 7–9 June, Ottawa, Ontario, Canada. Toronto, Canada: Canadian Information Processing Society, 159–166.

Becker, M. and Teschner, M., 2007. Weakly compressible SPH for free surface flows. *In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '07, 3–4 August, San Diego, California. Aire-la-Ville, Switzerland: Eurographics Association, 209–217.

Blasi, P., Le Saec, B., and Schlick, C., 1993. A rendering algorithm for discrete volume density objects. *Computer Graphics Forum*, 12 (3), 201–210.

Bridault, F., *et al.*, 2007. Real-time rendering and animation of plentiful flames. *In*: D.S. Ebert and S. Mérillou, eds. *Proceedings of the Eurographics workshop on natural phenomena*, NPH 2007, September, Prague, Czech Republic. Prague, Czech Republic: Eurographics Association, 31–38.

Bridson, R., 2008. *Fluid simulation for computer graphics*. Natick, MA: A.K. Peters.

Bridson, R., Houriham, J., and Nordenstam, M., 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics*, 26 (3), 46:1–46:4.

Brochu, T., Batty, C., and Bridson, R., 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Transactions on Graphics*, 29 (4), 47:1–47:9.

Carlson, M., Mucha, P.J., and Turk, G., 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics*, 23 (3), 377–384.

Carlson, M., *et al.*, 2002. Melting and flowing. *In: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '02, 21–22 July, San Antonio, Texas. New York, NY: ACM, 167–174.

Chen, S. and Doolen, G.D., 1998. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30, 329–364.

Chentanez, N. and Müller, M., 2010. Real-time simulation of large bodies of water with small scale details. *In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '10, 2–4 July, Madrid, Spain. Aire-la-Ville, Switzerland: Eurographics Association, 197–206.

Chentanez, N. and Müller, M., 2011. Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Transactions on Graphics*, 30 (4), 82:1–82:10.

Chentanez, N., *et al.*, 2006. Simultaneous coupling of fluids and deformable bodies. *In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '06, 2–4 September, Vienna, Austria. Aire-la-Ville, Switzerland: Eurographics Association, 83–89.

Chentanez, N., *et al.*, 2007. Liquid simulation on lattice-based tetrahedral meshes. *In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '07, 3–4 August, San Diego, California. Aire-la-Ville, Switzerland: Eurographics Association, 219–228.

Chiu, Y.F. and Chang, C.F., 2006. GPU-based ocean rendering. *In: 2006 IEEE international conference on multimedia and expo*, 9–12 July, Toronto, Ontario, Canada. Washington, DC: IEEE, 2125–2128.

Chu, N.S.H. and Tai, C.L., 2005. MoXi: real-time ink dispersion in absorbent paper. *ACM Transactions on Graphics*, 24 (3), 504–511.

Cieutat, J.M., Gonzato, J.C., and Guitton, P., 2003. A general ocean waves model for ship design. *In: Proceedings of virtual concept*, 5–7 November, Biarritz, France. Ecole Supérieure des Technologies Industrielles Avancées (ESTIA).

Crane, K., Llamas, I., and Tariq, S., 2007. Real time simulation and rendering of 3D fluids. *In*: H. Nguyen, ed. GPUGems 3. Reading, MA: Addison-Wesley, chap. 30.

Desbrun, M. and Cani, M.P., 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. *In:* R. Boulic and G. Hegron, eds. *Eurographics workshop on computer animation and simulation (EG-CAS)*, August, Poitiers, France. Published under the name Marie-Paule Gascuel. New York, NY: Springer-Verlag, 61–76.

Dobashi, Y., *et al.*, 2006. A fluid resistance map method for real-time haptic interaction with fluids. *In: Proceedings of the ACM symposium on virtual reality software and technology*, VRST '06, 1–3 November, Limassol, Cyprus. New York, NY: ACM, 91–99.

Elcott, S., *et al.*, 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics*, 26 (1), 4:1–4:12.

Enright, D., Marschner, S., and Fedkiw, R., 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics*, 21 (3), 736–744.

Fattal, R. and Lischinski, D., 2004. Target-driven smoke animation. *ACM Transactions on Graphics*, 23 (3), 441–448.

Fedkiw, R., Stam, J., and Jensen, H.W., 2001. Visual simulation of smoke. *In: Proceedings of the 28th annual conference on computer graphics and interactive techniques*, SIGGRAPH '01. New York, NY: ACM, 15–22.

Feldman, B.E., O'Brien, J.F., and Arikan, O., 2003. Animating suspended particle explosions. *ACM Transactions on Graphics*, 22 (3), 708–715.

Feldman, B.E., *et al.*, 2005. Fluids in deforming meshes. *In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '05, 29–31 July, Los Angeles, California. New York, NY: ACM, 255–259.

Foster, N. and Fedkiw, R., 2001. Practical animation of liquids. *In: Proceedings of the 28th annual conference on computer graphics and interactive techniques*, SIGGRAPH '01. New York, NY: ACM, 23–30.

Foster, N. and Metaxas, D., 1996. Realistic animation of liquids. *Graphic Models and Image Processing*, 58 (5), 471–483.

Fournier, A. and Reeves, W.T., 1986. A simple model of ocean waves. *In: Proceedings of the 13th annual conference on computer graphics and interactive techniques*, SIGGRAPH '86. New York, NY: ACM, 75–84.

Fraedrich, R., Auer, S., and Westermann, R., 2010. Efficient high-quality volume rendering of SPH data. *IEEE Transactions on Visualization and Computer Graphics*, 16 (6), 1533–1540.

Frisvad, J.R., Christensen, N.J., and Jensen, H.W., 2007. Computing the scattering properties of participating media using Lorenz-Mie theory. *ACM Transactions on Graphics*, 26 (3), 60:1–60:10.

Fuller, A.R., *et al.*, 2007. Real-time procedural volumetric fire. *In: Proceedings of the 2007 symposium on interactive 3D graphics and games*, I3D '07, 29 April–2 May, Seattle, Washington. New York, NY: ACM, 175–180.

Gamito, M.N., Lopes, P.F., and Gomes, M.R., 1995. Two-dimensional simulation of gaseous phenomena using vortex particles. *In: Proceedings of the 6th Eurographics workshop on computer animation and simulation*. Vienna, Austria: Springer-Verlag, 3–15.

Goswami, P. and Pajarola, R., 2011. Time adaptive approximate SPH. *In: Proceedings of the 8th workshop on virtual reality interactions and physical simulations, VRIPHYS 2011*. Aire-la-Ville, Switzerland: Eurographics Association, 19–28.

Greenwood, S.T. and House, D.H., 2004. Better with bubbles: enhancing the visual realism of simulated fluid. *In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '04, 27–29 August, Grenoble, France. Aire-la-Ville, Switzerland: Eurographics Association, 287–296.

Guendelman, E., *et al.*, 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics*, 24 (3), 973–981.

Harada, T., Koshizuka, S., and Kawaguchi, Y., 2007. Smoothed particle hydrodynamics on GPUs. *In: Proceedings of computer graphics international*. Geneva, Switzerland: Computer Graphics Society, 63–70.

Harris, M.J., *et al.*, 2003. Simulation of cloud dynamics on graphics hardware. *In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware*, HWWS '03, 26–27 July, San Diego, California. Aire-la-Ville, Switzerland: Eurographics Association, 92–101.

Hong, J.M. and Kim, C.H., 2005. Discontinuous fluids. *ACM Transactions on Graphics*, 24 (3), 915–920.

Hong, J.M., Shinar, T., and Fedkiw, R., 2007. Wrinkled flames and cellular patterns. *ACM Transactions on Graphics*, 26 (3), 47:1–47:6.

Hong, J.M., *et al.*, 2008. Bubbles alive. *ACM Transactions on Graphics*, 27 (3), 48:1–48:4.

Horvath, C. and Geiger, W., 2009. Directable, high-resolution simulation of fire on the GPU. *ACM Transactions on Graphics*, 28 (3), 41:1–41:8.

Ihmsen, M., *et al.*, 2011. Animation of air bubbles with SPH. *In: GRAPP 2011 – Proceedings of the international conference on computer graphics theory and applications*, 5–7 March, Vilamoura, Algarve, Portugal. Setubal, Portugal: SciTe Press, 225–234.

Inacio, R.T., *et al.*, 2010. Interactive simulation and visualization of fluids with surface raycasting. *In: Proceedings of the 2010 23rd SIBGRAPI conference on graphics, patterns and images*, SIBGRAPI '10. Washington, DC: IEEE Computer Society, 142–148.

Kang, N., *et al.*, 2010. A hybrid approach to multiple fluid simulation using volume fractions. *Computer Graphics Forum*, 29 (2), 685–694.

Kass, M. and Miller, G., 1990. Rapid, stable fluid dynamics for computer graphics. *In: Proceedings of the 17th annual conference on computer graphics and interactive techniques*, SIGGRAPH '90, August, Dallas, TX. New York, NY: ACM, 49–57.

Kim, B., *et al.*, 2007. Advections with significantly reduced dissipation and diffusion. *IEEE Transactions on Visualization and Computer Graphics*, 13 (1), 135–144.

Kim, D., Song, O.y., and Ko, H.S., 2010. A practical simulation of dispersed bubble flow. *ACM Transactions on Graphics*, 29 (4), 70:1–70:5.

Kim, D., *et al.*, 2012. Baroclinic turbulence with varying density and temperature. *IEEE Transactions on Visualization and Computer Graphics*, 18 (9), 1488–1495.

Kim, J., *et al.*, 2006. Practical animation of turbulent splashing water. *In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '06, 2–4 September, Vienna, Austria. Aire-la-Ville, Switzerland: Eurographics Association, 335–344.

Kim, T., *et al.*, 2008. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics*, 27 (3), 50:1–50:6.

Klingner, B.M., *et al.*, 2006. Fluid animation with dynamic meshes. *ACM Transactions on Graphics*, 25 (3), 820–825.

Kwatra, N., *et al.*, 2010. Fluid simulation with articulated bodies. *IEEE Transactions on Visualization and Computer Graphics*, 16 (1), 70–80.

Lamorlette, A. and Foster, N., 2002. Structural modeling of flames for a production environment. *ACM Transactions on Graphics*, 21 (3), 729–735.

Lapierre, S., Entezari, A., and Möller, T., 2003. *Practicalities of Fourier-Domain Fluid Simulation. Technical report*, Simon Fraser University, Canada.

Liu, Y., Liu, X., and Wu, E., 2004. Real-time 3D fluid simulation on GPU with complex obstacles. *In: Proceedings of the computer graphics and applications, 12th Pacific conference*, PG '04 Washington, DC: IEEE Computer Society, 247–256.

Long, B. and Reinhard, E., 2009. Real-time fluid simulation using discrete sine/cosine transforms. *In: Proceedings of the 2009 symposium on interactive 3D graphics and games*, I3D '09, 27 February–1 March, Boston, Massachusetts. New York, NY: ACM, 99–106.

Lorensen, W.E. and Cline, H.E., 1987. Marching cubes: A high resolution 3D surface construction algorithm. *In: Proceedings of the 14th annual conference on computer graphics and interactive techniques*, SIGGRAPH '87. New York, NY: ACM, 163–169.

Losasso, F., Gibou, F., and Fedkiw, R., 2004. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics*, 23 (3), 457–462.

Losasso, F., *et al.*, 2006a. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics*, 12 (3), 343–352.

Losasso, F., *et al.*, 2006b. Multiple interacting liquids. *ACM Transactions on Graphics*, 25, 812–819.

Losasso, F., *et al.*, 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14 (3), 797–804.

McNamara, A., *et al.*, 2004. Fluid control using the adjoint method. *ACM Transactions on Graphics*, 23 (3), 449–456.

Mihalef, V., Metaxas, D., and Sussman, M., 2009. Simulation of two-phase flow with sub-scale droplet and bubble effects. *Computer Graphics Forum*, 28 (2), 229–238.

Miklós, B., 2004. *Real-time fluid simulation using height fields. Semester thesis*. Swiss Federal Institute of Technology, Zurich.

Misztal, M.K., *et al.*, 2010. Optimization-based fluid simulation on unstructured meshes. *In: Proceedings of virtual reality interaction and physical simulation*, *VRIPHYS 2010*. Aire-la-Ville, Switzerland: Eurographics Association, 11–20.

Müller, M., *et al.*, 2004a. Point based animation of elastic, plastic and melting objects. *In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '04, 27–29 August, Grenoble, France. Aire-la-Ville, Switzerland: Eurographics Association, 141–151.

Müller, M., Charypar, D., and Gross, M., 2003. Particle-based fluid simulation for interactive applications. *In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '03, San Diego, California. Aire-la-Ville, Switzerland: Eurographics Association, 154–159.

Müller, M., *et al.*, 2004b. Interaction of fluids with deformable solids: Research articles. *Computer Animation & Virtual Worlds*, 15 (3–4), 159–171.

Narain, R., *et al.*, 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Transactions on Graphics*, 27 (5), 166:1–166:8.

Neyret, F., 2003. Advected textures. *In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '03, 26–27 July, San Diego, California. Aire-la-Ville, Switzerland: Eurographics Association, 147–153.

Nguyen, D.Q., Fedkiw, R., and Jensen, H.W., 2002. Physically based modeling and animation of fire. *ACM Transactions on Graphics*, 21 (3), 721–728.

Nielsen, M.B. and Bridson, R., 2011. Guide shapes for high resolution naturalistic liquid simulation. *ACM Transactions on Graphics*, 30 (4), 83:1–83:8.

Park, J., *et al.*, 2008. A unified handling of immiscible and miscible fluids. *Computer Animation & Virtual Worlds*, 19 (3–4), 455–467.

Park, S.I. and Kim, M.J., 2005. Vortex fluid for gaseous phenomena. *In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '05, 29–31 July, Los Angeles, California. New York, NY: ACM, 261–270.

Peachey, D.R., 1986. Modeling waves and surf. *In: Proceedings of the 13th annual conference on computer graphics and interactive techniques*, SIGGRAPH '86. New York, NY: ACM, 65–74.

Pfaff, T., Thuerey, N., and Gross, M., 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics*, 31 (4), 112:1–112:8

Pfaff, T., *et al.*, 2009. Synthetic turbulence using artificial boundary layers. *ACM Transactions on Graphics*, 28 (5), 121:1–121:10.

Premože, S., *et al.*, 2003. Particle-based simulation of fluids. *Computer Graphics Forum*, 22 (3), 401–410.

Qian, Y.H., D'Humires, D., and Lallemand, P., 1992. Lattice BGK models for Navier–Stokes equation. *EPL (Europhysics Letters)*, 17 (6), 479.

Raveendran, K., Wojtan, C., and Turk, G., 2011. Hybrid smoothed particle hydrodynamics. *In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '11, 5–6 August, Vancouver, British Columbia, Canada. New York, NY: ACM, 33–42.

Robinson-Mosher, A., *et al.*, 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Transactions on Graphics*, 27 (3), 46:1–46:9.

Sadlo, F., Peikert, R., and Sick, M., 2006. Visualization tools for vorticity transport analysis in incompressible flow. *IEEE Transactions on Visualization and Computer Graphics*, 12 (5), 949–956.

Selle, A., Rasmussen, N., and Fedkiw, R., 2005. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics*, 24 (3), 910–914.

Sethian, J.A., 1999. Fast marching methods. *SIAM Review*, 41 (2), 199–235.

Sin, F., Bargteil, A.W., and Hodgins, J.K., 2009. A point-based method for animating incompressible flow. *In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '09, 1–2 August, New Orleans, Louisiana. New York, NY: ACM, 247–255.

Solenthaler, B. and Gross, M., 2011. Two-scale particle simulation. *ACM Transactions on Graphics*, 30 (4), 81:1–81:8.

Solenthaler, B. and Pajarola, R., 2008. Density contrast SPH interfaces. *In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '08, 7–9 July, Dublin, Ireland. Aire-la-Ville, Switzerland: Eurographics Association, 211–218.

Solenthaler, B. and Pajarola, R., 2009. Predictive-corrective incompressible SPH. *ACM Transactions on Graphics*, 28 (3), 40:1–40:6.

Solenthaler, B., Schläfli, J., and Pajarola, R., 2007. A unified particle model for fluid–solid interactions. *Computer Animation & Virtual Worlds*, 18 (1), 69–82.

Stam, J., 1999. Stable fluids. *In: Proceedings of the 26th annual conference on computer graphics and interactive techniques*, SIGGRAPH '99. New York, NY: ACM Press/Addison-Wesley Publishing Co., 121–128.

Stam, J., 2001. A simple fluid solver based on the FFT. *Journal of Graphic Tools*, 6 (2), 43–52.

Stam, J., 2003. Real-time fluid dynamics for games. *In: Proceedings of the game developer conference*, GDC 2003. Mt. Royal, NJ: International Game Developers Association.

Terzopoulos, D., Platt, J., and Fleischer, K., 1989. Heating and melting deformable models (From goop to glop). *In: Graphics interface 1989*, 19–23 June, London, Ontario, Canada. Toronto, Canada: Canadian Information Processing Society, 219–226.

Tessendorf, J., 2001. Simulating ocean water. *In:* Simulating Nature: Realistic and Interactive Techniques. *SIGGRAPH 2001 course notes (Course 47)*. ACM.

Thürey, N., *et al.*, 2006. Detail-preserving fluid control. *In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '06, 2–4 September, Vienna, Austria. Aire-la-Ville, Switzerland: Eurographics Association, 7–12.

Thürey, N. and Rüde, U., 2004. Free surface Lattice-Boltzmann fluid simulations with and without level sets. *In: Proceedings of VMV 2004*, 16–18 November, Stanford, CA: IOS Press, 199–207.

Treuille, A., Lewis, A., and Popović, Z., 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics*, 25 (3), 826–834.

Treuille, A., *et al.*, 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics*, 22 (3), 716–723.

Vanzine, Y. and Vrajitoru, D., 2008. Pseudorandom noise for real-time volumetric rendering of fire in a production system. *In: IEEE/EG symposium on volume and point-based graphics*. Aire-la-Ville, Switzerland: Eurographics, 129–136.

Vines, M., Mora, J., and Lee, W.S., 2009. Haptic display of 3D liquids for interactive applications. *In: Games innovations conference*, ICE-GIC 2009. Washington, DC: International IEEE Consumer Electronics Society IEEE, 140–148.

Wang, C., *et al.*, 2003. Real-time simulation of ocean wave based on cellular automata. *In:* E.Wu, H.Sun and D.Qi, eds. *Proceedings of CAD/Graphics'2003*. Macao, China, 8th International Conference on CAD/Graphics, 2003, 26–31.

Wei, X., Li, W., and Kaufman, A., 2003. Melting and flowing of viscous volumes. *In: Proceedings of the 16th international conference on computer animation and social agents (CASA 2003)*, CASA '03. Washington, DC: IEEE Computer Society, 54–59.

Wei, X., *et al.*, 2002. Simulating fire with texture splats. *In: Proceedings of the conference on visualization '02*, VIS '02, October 27–November 1, Boston, Massachusetts. Washington, DC: IEEE Computer Society, 227–235.

Wei, X., *et al.*, 2004a. The Lattice-Boltzmann method for simulating gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics*, 10 (2), 164–176.

Wei, X., *et al.*, 2004b. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10 (6), 719–729.

Weißmann, S. and Pinkall, U., 2010. Filament-based smoke with vortex shedding and variational reconnection. *ACM Transactions on Graphics*, 29 (4), 115:1–115:12.

Wendt, J.D., *et al.*, 2007. Finite volume flow simulations on arbitrary domains. *Graphical Models*, 69 (1), 19–32.

Wicke, M., Stanton, M., and Treuille, A., 2009. Modular bases for fluid dynamics. *ACM Transactions on Graphics*, 28 (3), 39:1–39:8.

Wojtan, C., *et al.*, 2009. Deforming meshes that split and merge. *ACM Transactions on Graphics*, 28 (3), 76:1–76:10.

Wojtan, C., *et al.*, 2010. Physics-inspired topology changes for thin fluid features. *ACM Transactions on Graphics*, 29 (4), 50:1–50:8.

Wu, E., Liu, Y., and Liu, X., 2004. An improved study of real-time fluid simulation on GPU: Research articles. *Computer Animation & Virtual Worlds*, 15 (3–4), 139–146.

Yang, M., *et al.*, 2009. GPU methods for real-time haptic interaction with 3D fluids. *In: IEEE international workshop on haptic audio visual environments and games, 2009. HAVE 2009*, 7–8 November. New York: IEEE, 24–29.

Yngve, G.D., O'Brien, J.F., and Hodgins, J.K., 2000. Animating explosions. *In: Proceedings of the 27th annual conference on computer graphics and interactive techniques*, SIGGRAPH '00. New York: ACM Press/Addison-Wesley Publishing Co., 29–36.

Yoon, J.C., *et al.*, 2009. Procedural synthesis using vortex particle method for fluid simulation. *Computer Graphics Forum*, 28 (7), 1853–1859.

Yu, J. and Turk, G., 2010. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '10, 2–4 July, Madrid, Spain. Aire-la-Ville, Switzerland: Eurographics Association, 217–225.

Yu, J., *et al.*, 2012. Explicit mesh surfaces for particle based fluids. *In: Proceedings of Eurographics 2012*. Aire-la-Ville, Switzerland: Eurographics Association.

Yuksel, C., House, D.H., and Keyser, J., 2007. Wave particles. *ACM Transactions on Graphics*, 26 (3), 99:1–99:8.

Zhang, F., *et al.*, 2011. A SPH-based method for interactive fluids simulation on the multi-GPU. *In: Proceedings of the 10th international conference on virtual reality continuum and its applications in industry*, VRCAI '11, 11–12 December, Hong Kong, China. New York, NY: ACM, 423–426.

Zhao, Y., Yuan, Z., and Chen, F., 2010. Enhancing fluid animation with adaptive, controllable and intermittent turbulence. *In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*, SCA '10, 2–4 July, Madrid, Spain. Aire-la-Ville, Switzerland: Eurographics Association, 75–84.

Zhao, Y., *et al.*, 2003. Voxels on fire. *In: Proceedings of the 14th IEEE visualization 2003 (VIS'03)*, VIS '03. Washington, DC: IEEE Computer Society, 271–278.

Zhao, Y., *et al.*, 2006. Melting and flowing in multiphase environment. *Computers & Graphics*, 30 (4), 519–528.

Zhu, H., *et al.*, 2006. Simulation of miscible binary mixtures based on Lattice Boltzmann method: Research articles. *Computer Animation & Virtual Worlds*, 17 (3–4), 403–410.

Zhu, Y. and Bridson, R., 2005. Animating sand as a fluid. *ACM Transactions on Graphics*, 24 (3), 965–972.