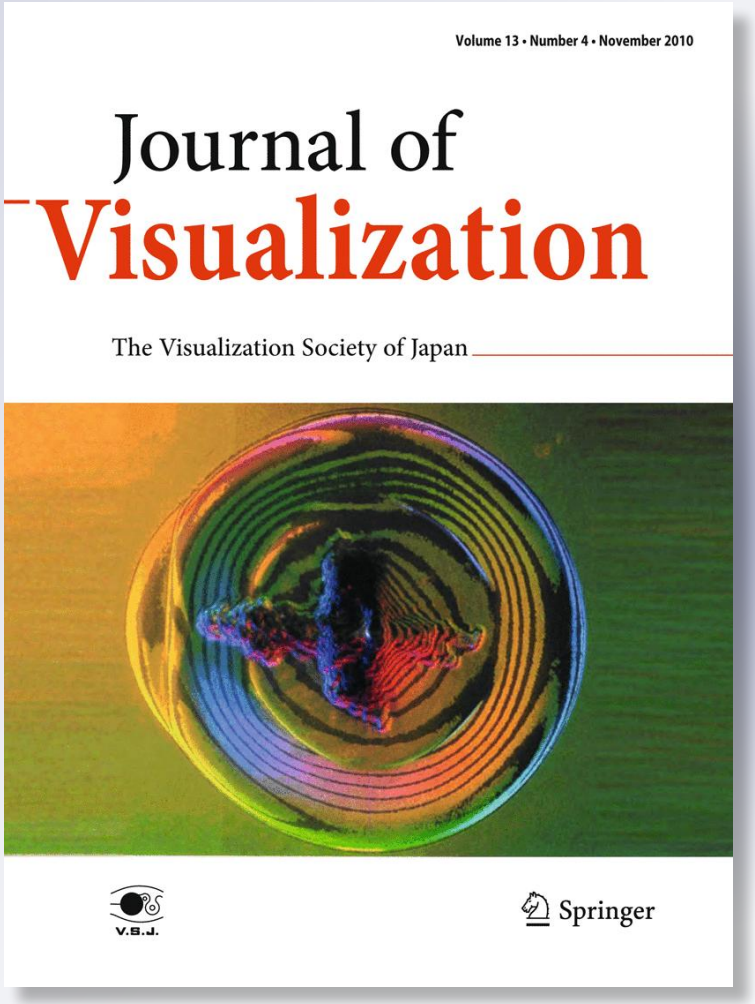
Particle-based simulation and visualization of fluid flows through porous media

Journal of Visualization

ISSN 1343-8875
Volume 13
Number 4

J Vis (2010) 13:327-336
DOI 10.1007/
s12650-010-0041-2



Springer

J Vis (2010) 13:327–336
DOI 10.1007/s12650-010-0041-2

REGULAR PAPER

Serkan Bayraktar · Uğur Güdükbay · Bülent Özgüç

Particle-based simulation and visualization of fluid flows through porous media

Received: 4 January 2010 / Revised: 19 April 2010 / Accepted: 12 May 2010 / Published online: 11 June 2010

Abstract We propose a method of fluid simulation where boundary conditions are designed in such a way that fluid flow through porous media, pipes, and chokes can be realistically simulated. Such flows are known to be low Reynolds number incompressible flows and occur in many real life situations. To obtain a high quality fluid surface, we include a scalar value in isofunction. The scalar value indicates the relative position of each particle with respect to the fluid surface.

Keywords Physically based modeling · Fluid simulation · Particle-based modeling · Smoothed particle hydrodynamics

1 Introduction

Being one of the most common natural phenomena, fluid behavior has been one of the well-researched topics. In literature, several methods have been proposed to simulate fluid flow realistically and efficiently. These methods can be grouped into two main categories: Eulerian, grid-based methods, and Lagrangian, particle-based methods. Eulerian methods use a grid structure that subdivides the simulation domain into cells. The parameters that control the fluid behavior are computed for each cell. This is achieved by discretizing Computational Fluid Dynamics (CFD) equations with respect to the cell structure. Lagrangian methods represent fluid mass with a set of particles that carry fluid characteristics with them, and are advected by the fluid's velocity field.

Smoothed particle hydrodynamics (SPH) is one of the Lagrangian methods that has been used in astrophysics and CFD to simulate fluid behavior in many different situations. In SPH, fluid volume is represented by particles whose fluid characteristics affect a spherical volume which is called "smoothing kernel" (Monaghan 1994).

Electronic supplementary material The online version of this article (doi:10.1007/s12650-010-0041-2) contains supplementary material, which is available to authorized users.

S. Bayraktar · U. Güdükbay (✉) · B. Özgüç
Department of Computer Engineering, Bilkent University, Bilkent, 06800 Ankara, Turkey
E-mail: gudukbay@cs.bilkent.edu.tr
Tel.: +90-312-2901386
Fax: +90-312-2664047

S. Bayraktar
e-mail: serkan@cs.bilkent.edu.tr

B. Özgüç
e-mail: ozguc@bilkent.edu.tr

We use SPH to simulate behavior of fluid flow through pipes and porous medium. Such flows have a common feature of having low Reynolds numbers. Flow characteristics of such flows are dominated by boundary interactions and viscous forces. Thus, we adopt computational methods used in CFD to simulate behavior of fluid flow through pipes, filters, and porous materials.

Fluid surface construction is a challenging task for particle-based fluid simulation systems. One of the commonly used methods is to construct an isosurface by the well known Marching Cubes algorithm (Lorensen and Cline 1987). We propose an improvement to the isofunction computation such that the relative position of each particle with respect to the fluid surface influences the value of the isofunction. The result is a more detailed and smooth fluid surface. Locating the neighbors usually dominates the run time of a particle-based simulation system. We use a neighbor search algorithm where particle positions are hashed and sorted at each time step.

2 Related work

Particle-based modeling and simulation methods is a well-studied area. Reeves (1983) uses particles to model fuzzy objects (e.g., fire works). Miller and Pearce (1989) utilize pairwise particle interactions to model viscous fluids. Terzopoulos et al. (1991) model melting objects with interacting particles connected by springs whose constants are modified as the object changes its phase. Tonnesen (1991) incorporates a discrete form of heat transfer equation into inter-particle force equations to simulate change in particle positions due to thermal energy. Desbrun and Cani (1996) use SPH to model highly deformable objects. Stora et al. (1999) use SPH and heat transfer equations to simulate lava flows. Müller et al. (2004) utilize SPH version of Navier–Stokes equations to model incompressible fluid and handle fluid–deformable body interactions. Premoze et al. (2003) use particles to simulate incompressible fluids where they ensure incompressibility by using the moving particle semi-implicit method.

Hadap and Magnenat-Thalmann (2001) couple SPH and strand dynamics to simulate hair–hair, and hair–air interactions. Kruger et al. (2005) use graphics processing unit (GPU) to achieve interactive rates for large particle sets. Kipfer and Westermann (2006) use GPU-based data structure and SPH fluid simulation method to model and render interactive simulation of rivers. Solenthaler et al. (2007) simulate fluid, deformable bodies, and melting and solidification by using SPH and elastic–plastic model.

Becker and Teschner (2007) use the Tait equation so that their free flow fluid model is weakly compressible. Cleary et al. (2007) utilize SPH to model bubble and froth generation and their coupling with the fluid body. Losasso et al. (2008) propose a two-way coupled simulation system where dense liquid volumes are simulated using the particle level set and diffuse regions such as mixture of air and sprays are simulated by SPH.

As an alternative to particle-based Lagrangian methods, Eulerian methods are used where the equations governing the fluid behavior are solved in a (usually) regular grid. Foster and Metaxas (1996) are first to introduce 3D Eulerian form of the Navier Stokes equations in computer graphics. Stam (1999) introduces the so-called "semi-Lagrangian" numerical methods, which are unconditionally stable, thus allowing use of large time steps. Foster and Fedkiw (2001) extend the semi-Lagrangian method and use conjugate gradient method to enforce incompressibility. Enright et al. (2002) improve the level-set-based surface generation method to ensure mass preservation and photo-realistic fluid effects. Carlson et al. (2004) use Eulerian grid-based methods to model two way rigid–fluid interaction. Guendelman et al. (2005) use a complex surface traction method implemented in an octree grid so that fluid interaction with thin rigid objects and deformable bodies, such as cloth, is possible. Song et al. (2007) propose the derivative particle method where they implement the non-advection part of the simulation in a conventional Eulerian grid and use a Lagrangian scheme for the advection part. A notable work on the simulation of fluid flow through porous materials is described by Lenaerts et al. (2008). They use the Law of Darcy to model the physical principles of porous flow and SPH to simulate fluids and deformable objects.

3 Flows through porous medium

Fluid flow through porous medium has been an important research topic in engineering and computational physics (Bear 1988; Morris et al. 1997; Zhu and Fox 2002; Zhu et al. 1999). For fluid flow through porous materials, pipes, channels, and chokes, Reynolds number is low and the flow characteristics of such flows are dominated by viscous (frictional) effects rather than inertial forces. Reynolds number in fluid mechanics is defined as the ratio of inertial forces to viscous forces and is one of the indicators of the flow characteristics.

3.1 Smoothed particle hydrodynamics

To simulate fluid flow through time, we need to relate the total forces acting on a particle p_i to the rate of change of the particle's momentum. This relation is called the momentum equation and for the SPH-based particle simulation it can be written as:

$$\frac{\mathbf{v}_i}{\mathrm{d}t} = f_i^{\text{pressure}} + f_i^{\text{viscous}} + F_i, \tag{1}$$

where f_i^{pressure} is the pressure-based and f_i^{viscous} is the viscosity-based forces acting on the particle p_i, respectively. F_i denotes the body forces such as gravity.

In the SPH method, the fluid is represented by a set of particles that carry various fluid properties such as mass, velocity, and density. These properties are distributed around the particle according to an interpolating kernel function W whose finite support is h. For each point x in simulation space, the value of a fluid property can be computed by interpolating the contributions of fluid particles residing within a spherical region with radius h and centered at x. The density ρ_i at the position of particle i can be interpolated by Eq. 2:

$$\rho_i = \sum_j m_j W_{ij}, \tag{2}$$

where $W_{ij} = W(\mathbf{r}_{ij}, h), \mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j, \mathbf{r}_i$ is particle i's position, and h is the kernel support radius. For the density computation, we use the following kernel, which is originally proposed by Müller et al. (2004):

$$W(\mathbf{r}_{ij}, h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - |\mathbf{r}_{ij}|^2\right)^3 & \text{if } 0 \le |\mathbf{r}_{ij}| \le h \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

In SPH, pressure is an explicit function of local density (Monaghan 1994). Using a quasi-incompressible equation of state, the pressure p_i for particle i is computed as

$$p_i = k(\rho_i - \rho_0), \tag{4}$$

where the usual choice for coefficient k is c_s^2, c_s being the speed of sound. ρ_0 is the reference density and its inclusion results in more accurate simulations (Morris et al. 1997). The force due to pressure acting on particle i can be computed by (Monaghan 1994):

$$f_i^{\text{pressure}} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}. \tag{5}$$

Viscous forces for low Reynolds number flows can be computed as described in (Morris et al. 1997):

$$f_i^{\text{viscous}} = \sum_j \frac{m_j(\mu_i + \mu_j)\mathbf{v}_{ij}}{\rho_i \rho_j} \left(\frac{1}{r_{ij}} \frac{\partial W_{ij}}{\partial r_i}\right) \tag{6}$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ is the relative velocity of particle i with respect to particle j and μ_i is the viscosity of particle i. We use the following kernel for the viscosity and pressure computations:

$$W(\mathbf{r}_{ij}, h) = \frac{45}{\pi h^6} \begin{cases} \left(h - |\mathbf{r}_{ij}|\right) & \text{if } 0 \le |\mathbf{r}_{ij}| \le h \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

We take the values of the kernel radius, viscosity, the reference density as 1.0, 4.3, and 9.8, respectively. By computing internal forces (e.g., pressure- and viscosity-based forces) and external forces (e.g., gravity, boundary, and the user-defined forces), we obtain the rate of change in momentum. This momentum change is, then, integrated numerically to resolve the change in position.

3.2 Boundary conditions

To realistically simulate fluid flow behavior through porous media, non-penetration and no-slip boundary conditions must be satisfied. Non-penetration condition requires that fluid particles do not penetrate into the boundary region. Typically, non-penetration boundary condition in SPH applications is enforced by inserting stationary boundary particles to construct a layer of width $2h$, which exerts repulsive forces on the

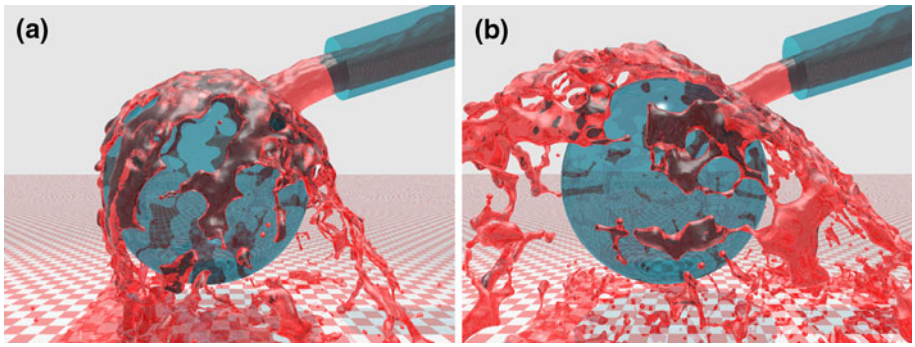


Fig. 1 Fluid pours down on a sphere with (**a**) adhesion effect and (**b**) no-adhesion effect (Animation 6)

fluid particles (Monaghan 1994). A method to ensure non-penetration is to compute pressure-based SPH force between fluid and boundary particles and exert that force on the fluid particles. To achieve this, pressure and density values of boundary particles are also computed and evolved. Another choice for repulsive boundary forces is known as the Lennard–Jones potential (Allen and Tildesley 1987), which is widely used in molecular dynamics. The Lennard–Jones potential is defined as:

$$f(r) = \begin{cases} \frac{D}{r}\left(\left(\frac{r_0}{r}\right)^{p_1} - \left(\frac{r_0}{r}\right)^{p_2}\right) & \text{if } 0 \le r \le r_0 \\ 0 & \text{if } r > r_0 \end{cases}, \tag{8}$$

where D is a dimensionless constant and r is the distance. We choose D as 1 and r_0 as the kernel radius h. The usual choices of p_0 and p_1 are 12 and 6, respectively (Monaghan 1994). This formulation of the Lennard–Jones potential is solely repulsive. We observed that the Lennard–Jones potential results in more stable repulsion forces than pressure-based SPH forces.

Slip conditions consider force exerted on the fluid particles in tangential direction of the boundary surface. In free-slip condition, no force is exerted on fluid particles. In no-slip condition, a tangential force is applied so that tangential velocity is zero. Slip conditions can be simulated by modeling viscous drag applied by boundary particles. A viscosity-based force is dependent on the velocity of fluid particles, the distance between the fluid particles and boundary particles, and the viscosity coefficients of fluid particles. To compute the viscous force acting on fluid particles, velocity and density values of boundary particles are computed by interpolating the velocities of neighboring fluid particles. The velocity $\mathbf{v}_j$ of boundary particle j is computed as:

$$\mathbf{v}_j = \sum_i m_i \mathbf{v}_i W_{ij}, \tag{9}$$

where W_{ij} is the kernel defined by Eq. 3, and m_i and $\mathbf{v}_i$ are the mass and velocity of neighboring fluid particle i, respectively. The density value of a boundary particle is found by Eq. 2. The viscous drag exerted on fluid particles by boundary particles is computed by Eq. 6. The boundary particles are stationary and their evolved density and velocity values are used only in viscous drag computation.

In addition to ensuring non-penetration and no-slip conditions, boundary particles contribute to fluid particles' pressure-based force computation. This creates an adhesive force that prevents fluid particles from leaving the solid boundary freely. We evaluate the pressure value of each solid boundary particle and apply the force computed by the following equation on each fluid particle p_i for creating a realistic and easily controllable adhesion-like effect.

$$f_i^{\text{adhesive}} = \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}, \tag{10}$$

where j is the neighboring boundary particle of particle i, m_j is the mass, p_j is the pressure, ρ_j is the density of particle j, and W_{ij} is as defined by Eq. 7. Figure 1 shows the still images from two simulations of the same scene (see Animation 6). In the left frame, proposed pressure-based adhesion force is active and in the right there is no adhesion force acting on the fluid particles.

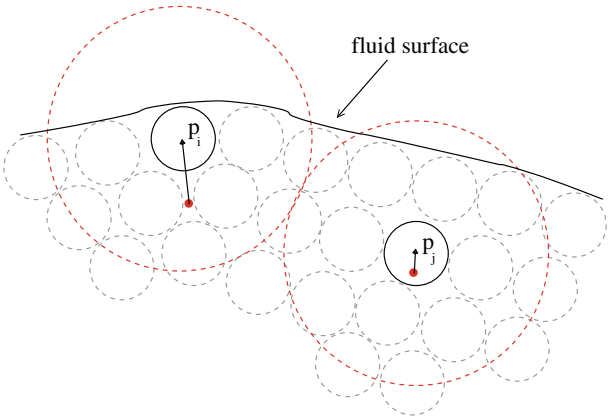


Fig. 2 Computing particle normals. The particle closer to the surface p_i has a longer normal vector than p_j since the center of masses of the neighbor particles (the *red dot*) is farther in case of p_i

4 Implementation

4.1 Surface generation

For Lagrangian-based fluid simulation methods, it is a challenging task to extract a fluid surface since particles do not carry any explicit information about their spatial arrangement and connectivity. Typically, an isosurface is computed from particle positions and polygonized for rendering. One of the frequently used algorithms for polygonizing an isosurface is the Marching Cubes (Lorensen and Cline 1987).

The Marching Cubes algorithm traces a uniform grid iteratively and achieve a tessellation that is based on the scalar values computed at each grid location. Within the context of particle-based fluid simulation, this scalar value is computed by considering the neighboring fluid particles. For a grid location x, the function $\phi(x)$ computes the scalar value:

$$\phi(x) = \sqrt{\left(\sum_i \left(1 - \left(\frac{r_i}{h}\right)^2\right)^3\right)}, \tag{11}$$

where i iterates over the fluid particle neighbors of x, and $r_i = |\mathbf{x} - \mathbf{p}_i|$, where p_i is the fluid particle with $r_i \leq h$.

The resulting surface captures the main features of the fluid body but it has a thickening effect in detailed surface regions such as waves and water fronts and a bumpy look in flat regions. Adams et al. (2007) address this problem by using a weighted function for the isosurface computation. We propose a modification to Eq. 11 such that it differentiates particles according to their relative positions to the fluid surface by assigning a value to each of the particles according to their proximity to the fluid surface. We refer to this value as the *surface value*. We calculate the surface values for each particle using normals, as described in Steele et al. (2004). For each particle i, we find the centroid c_i of the sphere with radius h centered at the location of particle i. The vector $\mathbf{n}_i = \mathbf{r}_i - \mathbf{c}_i$ is the normal vector. The length of the normal vector indicates relative proximity of the particle to the fluid surface. The normal vectors of the particles that are closer to the surface are larger in magnitude than those of the particles located within the fluid body (Fig. 2). We compute the surface values for each particle p_i by Eq. 12:

$$s_i = 1 - \left(\frac{\sum_j \|\mathbf{n}_j\|}{kh} + \frac{\Re}{2}\right), \tag{12}$$

where p_j is a neighbor of p_i, and k is the number of such neighbors. $\Re$ is defined to be the maximum of $\Re_i$'s where

$$\Re_i = \frac{\sum_j \|\mathbf{n}_j\|}{kh}. \tag{13}$$

Equation 12 ensures that particles closer to the fluid surface have smaller surface values, thus contribute to the isosurface less than non-surface particles. After including s_i to the function $\phi(x)$, it becomes:

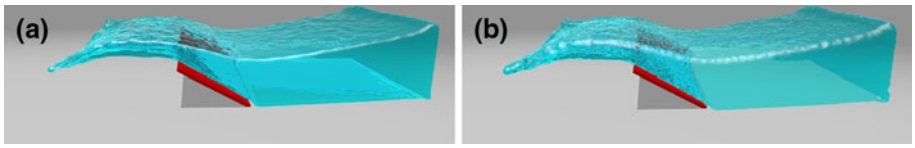


Fig. 3 Fluid dam breaks into a rigid object. **a** The surface is generated by incorporating the surface value; **b** the surface value is not incorporated (Animation 7)

$$\phi(x) = \sqrt{\left(\sum_i \left(1 - \left(\frac{r_i s_i}{h}\right)^2\right)^3\right)}. \tag{14}$$

By incorporating the surface value s_i, the generated fluid surface is more realistic on flat surfaces and captures finer details in splashes and filaments since the fluid particles close to the surface contribute less to the isovalue. Figure 3 illustrates the effectiveness of the modified surface generation algorithm. The figure includes two frames of the same scene. Figure 3a shows a frame from the simulation where the surface value is incorporated into the surface generation and Fig. 3b shows a frame from the simulation where the surface value is not incorporated (see Animation 7).

We experimented several values for the grid resolution of the Marching Cubes algorithm. Finer grids produce high quality surfaces at the expense of increasing the computational cost. We determined experimentally that the grid resolution of $\frac{1}{6}h$, where h is the kernel radius, produces good results. With this resolution, grid points surrounded by the fluid body have around 80 neighboring fluid particles.

4.2 Neighbor search

Locating particle neighbors dominates the run time of particle-based simulations. Space subdivision methods that employ a uniform grid have been proposed to achieve faster contact detection. These methods discretize the simulation space to improve the performance of contact detection. Bounding volumes and binary space partitioning (BSP) trees are among other methods of improving contact detection performance. Grid-based approaches used in SPH-based particle simulation systems employ a uniform grid with cell size of $2h$. By registering the particles to grid cells, the neighbor search time complexity can be reduced from $O(n^2)$ to linear time. Particles are registered to 27 cells including the host cell and its immediate neighbor cells and potential neighbor particles are searched only within these 27 cells. We use a grid-based method that hashes particle coordinates and sorts them according to their hash values. Potential neighbors are searched by scanning these sorted particles. The details of the neighbor search algorithm can be found in Bayraktar et al. (2009).

4.3 Numerical integration

Our choice for numerical integration is the velocity Verlet algorithm. It is a variation of the common Verlet algorithm. Its error in position and velocity is $O(\Delta t^3)$. It requires computation of particle accelerations once per time step and is much more stable than simple forward Euler technique. The velocity Verlet algorithm updates the velocities and positions according to the following equations:

$$\begin{aligned}
\mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \left(\frac{1}{2}\right)\mathbf{a}(t)\Delta t^2, \\
\mathbf{v}\left(t+\frac{\Delta t}{2}\right) &= \mathbf{v}(t) + \left(\frac{1}{2}\right)\mathbf{a}(t)\Delta t, \\
\mathbf{a}(t+\Delta t) &= \left(\frac{1}{m}\right)\mathbf{f}, \\
\mathbf{v}(t+\Delta t) &= \mathbf{v}\left(t+\frac{\Delta t}{2}\right) + \left(\frac{1}{2}\right)\mathbf{a}(t+\Delta t)\Delta t,
\end{aligned} \tag{15}$$

where m is the mass, $\mathbf{r}$ is the position, $\mathbf{v}$ is the velocity, $\mathbf{a}$ is the acceleration, and $\mathbf{f}$ is the total force acting on the particle. The time step Δt should satisfy the Courant–Friedrichs–Lewy (CFL) condition. The time step Δt should satisfy the following two constraints (Eq. 16) to meet the CFL condition (Monaghan 1994):

Fig. 4 Fluid is poured onto a hanged cloth and leaks through the pores of cloth (Animation 1)

Fig. 5 A river flowing through a stack of rocks (Animation 2)

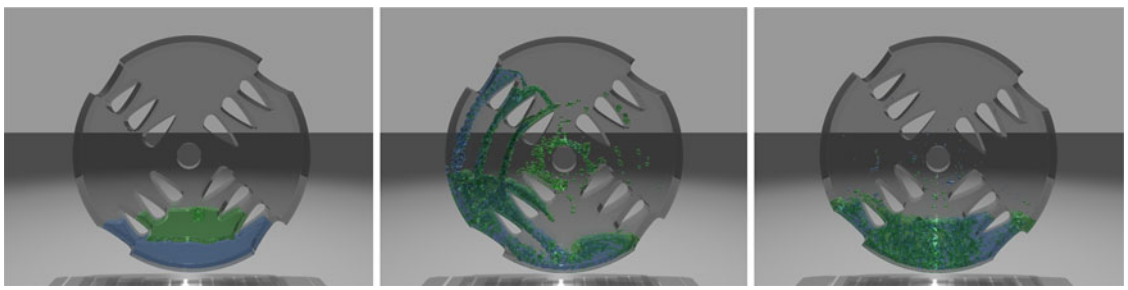


Fig. 6 Two types of fluids with different densities are mixed by a rotating mixer (Animation 3)

$$\Delta t \leq \mathrm{CFL} \times \min_i \sqrt{\frac{h}{\mathbf{f}_i}}, \quad \Delta t \leq \mathrm{CFL} \times \min_i \frac{h}{\mathbf{v}_i}, \quad 0 \leq \mathrm{CFL} \leq 1.0, \tag{16}$$

where f_i is the net force on each particle and $\mathbf{v}_i$ is the velocity of each particle.

5 Results

Figure 4 gives still images from an animation where fluid is poured onto a cloth mesh, which is hanged from its edges (see Animation 1). Fluid leaks through the pores of the cloth mesh. The cloth is modeled as a mass-spring mesh. Interactions between fluid and cloth are handled in particle-to-particle basis where governing forces are computed according to the proposed method. The simulation runs at 14 frames per second (fps). Figure 5 shows still images from an animation where a river flowing through a stack of rocks blocking a valley bed (see Animation 2). By employing the boundary conditions mentioned in Sect. 3.2, water flows through a complex structure realistically. The scene consists of 60 K fluid particles and 220 K solid boundary particles including the stone stack and the valley. The simulation runs at 12 fps. Figure 6 gives still images from an animation where two bodies of fluid are mixed by a rotating disk (see Animation 3). The example demonstrates the splashing and mixing effects of two fluid bodies with different densities. The scene has a total of 50 K fluid particles and 150 K boundary particles. The simulation runs at 14 fps.

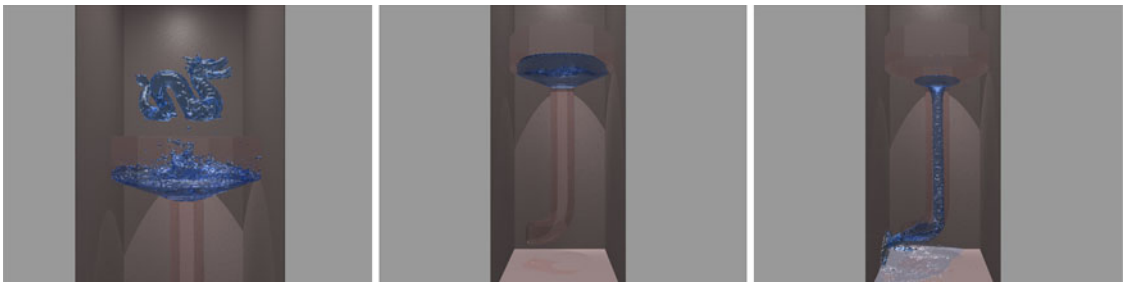


Fig. 7 Dragons dropped into a container and flow down through a pipe (Animation 4)

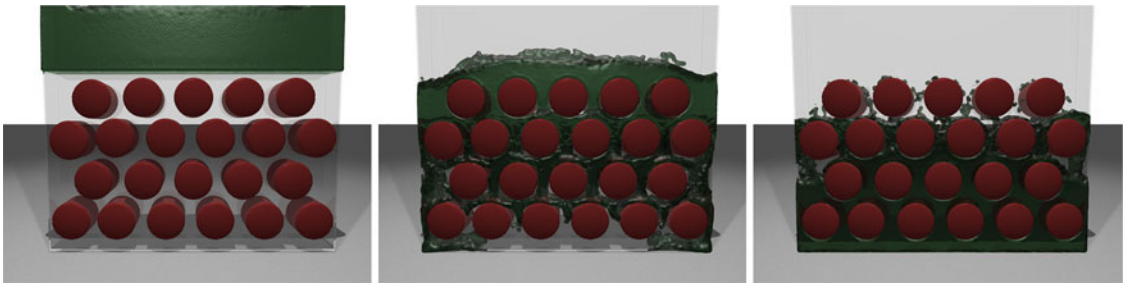


Fig. 8 Fluid flows through a set of cylinders (Animation 5)

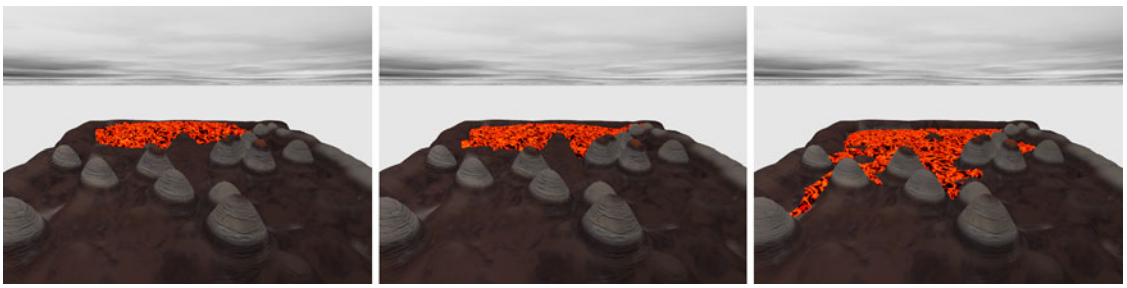


Fig. 9 Lava flows down on a terrain (Animation 9)

Figure 7 shows still images from an animation where fluid dragons are dropped into a container (see Animation 4). After the fluid stabilizes, it flows down through a pipe. The simulation runs at 12 fps. Figure 8 gives still images from an animation where fluid runs through a porous medium constructed by a set of cylinders (see Animation 5). The scene consists of 48 K fluid particles and 130 K solid boundary particles. The simulation runs at 15 fps.

Figure 9 shows still images from an animation where a viscous fluid, lava, runs downs on a terrain (see Animation 8). The value of the viscosity coefficient for the simulation is 12.0. The lava and terrain consist of 50 K and 60 K particles, respectively and the simulation runs at 25 fps.

The animations are rendered by the ray tracer POVRay (2009). The results are obtained using a PC equipped with Athlon^{TM} 64 X2 Dual Core processor and NVIDIA GeForce^{TM} 7900 GS graphics board with 256 MB of VRAM. To take advantage of dual core structure of the processor, we use the Open Multi-Processing (OpenMP) Application Programming Interface (API).

6 Conclusion

We present a fluid simulation system based on the SPH paradigm. Our method improves the widely used technique of employing stationary boundary particles to satisfy non-penetration and no-slip conditions.

Frictional forces between solid boundaries and fluid are modeled by using a SPH-based method for ensuring uniformity and realism of system. A pressure-based force is modeled between the boundary and fluid particles to prevent fluid particles from leaving the boundary freely, thereby simulating a realistic looking adhesion effect.

We propose an improved surface extraction method where we compute the surface value for each fluid particle indicating its relative position to the fluid surface. The surface value ensures that fluid particles closer to the fluid–air boundary have less influence on isosurface. This results in higher quality surfaces which are not bumpy in flat regions and generate more detail in splashes and waves. We use a neighbor search algorithm that hashes particle coordinates and sort particles with respect to the hash value and search for potential contacts by scanning the sorted particle list.

One of the disadvantages of the proposed method is that the computation of hydrodynamic properties of the rigid object particles and the computation of the surface values for the fluid particles introduce a computational overhead. One obvious method to alleviate this burden is to parallelize the whole system either by exploiting the power of multi-core CPUs or implement the system on the GPU. Another possible improvement of the system would be implementing a dynamically allocated grid for the Marching Cubes algorithm since the fine grid statically allocated for this purpose is not memory efficient, especially for large scenes.

References

Adams B, Pauly M, Keiser R, Guibas LJ (2007) Adaptively sampled particle fluids. ACM Trans Graph (Proc of SIGGRAPH'07) 26(3):8; Article no. 48

Allen MP, Tildesley DJ (1987) Computer simulation of liquids. Clarendon Press, New York

Bayraktar S, Güdükbay U, Özgüç B (2009) GPU-based neighbor-search algorithm for particle simulations. J Graph GPU Game Tools 14(1):31–42

Bear J (1988) Dynamics of fluids in porous media. Courier Dover, New York

Becker M, Teschner M (2007) Weakly compressible SPH for free surface flows. In: Proceedings of ACM SIGGRAPH/ Eurographics symposium on computer animation, pp 209–217

Carlson M, Mucha PJ, Turk G (2004) Rigid fluid: animating the interplay between rigid bodies and fluid. ACM Trans Graph 23(3):377–384

Cleary PW, Pyo SH, Prakash M, Koo BK (2007) Bubbling and frothing liquids. ACM Trans Graph (Proc of SIGGRAPH'07) 26(3):6; Article no. 97

Desbrun M, Cani M (1996) Smoothed particles: a new paradigm for animating highly deformable bodies. In: Eurographics workshop on computer animation and simulation (EGCAS), pp 61–76

Enright D, Marschner S, Fedkiw R (2002) Animation and rendering of complex water surfaces. ACM Trans Graph (Proc of SIGGRAPH '02) 21(3):736–744

Foster N, Fedkiw R (2001) Practical animation of liquids. ACM Comp Graph (Proc. of SIGGRAPH '01), 23–30

Foster N, Metaxas D (1996) Realistic animation of liquids. Graph Model Image Process 58(5):471–483

Guendelman E, Selle A, Losasso F, Fedkiw R (2005) Coupling water and smoke to thin deformable and rigid shells. ACM Trans Graph 24(3):973–981

Hadap S, Magnenat-Thalmann N (2001) Modeling dynamic hair as a continuum. Comp Graph Forum 20(3):329–338

Kipfer P, Westermann R (2006) Realistic and interactive simulation of rivers. In: Proceedings of graphics interface, pp 41–48

Kruger J, Kipfer P, Kondratieva P, Westermann R (2005) A particle system for interactive visualization of 3D flows. IEEE Trans Vis Comp Graph 11(6):744–756

Lenaerts T, Adams B, Dutré P (2008) Porous flow in particle-based fluid simulations. ACM Trans Graph (Proc. of SIGGRAPH'08) 27(3):8; Article no. 49

Lorensen W, Cline H (1987) Marching cubes: a high resolution 3D surface construction algorithm. ACM Comp Graph (Proc of SIGGRAPH'87) 21(4):163–169

Losasso F, Talton JO, Kwatra N, Fedkiw R (2008) Two-way coupled SPH and particle level set fluid simulation. IEEE Trans Vis Comp Graph 14(4):797–804

Miller G, Pearce A (1989) Globular dynamics: a connected particle system for animating viscous fluids. Comp Graph 13(3):305–309

Monaghan J (1994) Simulating free surface flows with SPH. J Comp Phys 110(2):399–406

Morris JP, Fox PJ, Zhu Y (1997) Modeling low Reynolds number incompressible flows using SPH. J Comp Phys 136(1):214–226

Müller M, Schirm S, Teschner M, Heidelberger B, Gross M (2004) Interaction of fluids with deformable solids. J Comp Anim Virtual Worlds 15(3–4):159–171

POVRay (2009) The persistence of vision raytracer. http://www.povray.org/

Premoze S, Tasdizen T, Bigler J, Lefohn AE, Whitaker RT (2003) Particle-based simulation of fluids. Comp Graph Forum (Proc of Eurographics'03) 22(3):401–410

Reeves WT (1983) Particle systems: a technique for modeling a class of fuzzy objects. ACM Trans Graph 2(2):91–108

Solenthaler B, Schläfli J, Pajarola R (2007) A unified particle model for fluid–solid interactions. Comp Anim Virtual Worlds 18(1):69–82

Song OY, Kim D, Ko HS (2007) Derivative particles for simulating detailed movements of fluids. IEEE Trans Vis Comp Graph 13(4):711–719
Stam J (1999) Stable fluids. In: ACM Comp Graph (Proc. of SIGGRAPH '99), Addison Wesley, Los Angeles, pp 121–128
Steele K, Cline D, Egbert PK, Dinerstein J (2004) Modeling and rendering viscous liquids. Comp Anim Virtual Worlds 15(3–4):183–192
Stora D, Agliati PO, Cani MP, Neyret F, Gascuel JD (1999) Animating lava flows. In: Proceedings of graphical interface, pp 203–210
Terzopoulos D, Platt J, Fleischer K (1991) Heating and melting deformable models. J Vis Comp Anim 2(2):68–73
Tonnesen D (1991) Modeling liquids and solids using thermal particles. In: Proceedings of graphical interace, pp 255–262
Zhu Y, Fox PJ (2002) Simulation of pore-scale dispersion in periodic porous media using smoothed particle hydrodynamics. J Comp Phys 182(2):622–645
Zhu Y, Fox PJ, Morris JP (1999) A pore-scale numerical model for flow through porous media. Int J Numer Anal Methods Geomech 23:881–904