EUROGRAPHICS 2003 / P. Brunet and D. Fellner
(Guest Editors)

Volume 22 (2003), Number 3

Particle-Based Simulation of Fluids

Simon Premože[1], Tolga Tasdizen[2], James Bigler[2], Aaron Lefohn[2] and Ross T. Whitaker[2]

[1] Computer Science Department, University of Utah
[2] Scientific Computing and Imaging Institute, University of Utah

Abstract

Due to our familiarity with how fluids move and interact, as well as their complexity, plausible animation of fluids remains a challenging problem. We present a particle interaction method for simulating fluids. The underlying equations of fluid motion are discretized using moving particles and their interactions. The method allows simulation and modeling of mixing fluids with different physical properties, fluid interactions with stationary objects, and fluids that exhibit significant interface breakup and fragmentation. The gridless computational method is suited for medium scale problems since computational elements exist only where needed. The method fits well into the current user interaction paradigm and allows easy user control over the desired fluid motion.

1. Introduction

Fluids and our interactions with them are part of our everyday lives. Due to our familiarity with fluid movement, plausible simulation of fluids remains a challenging problem despite enormous improvements[7, 6]. Advances in *computational fluid dynamics* (CFD) often cannot be directly applied to computer graphics, because they have vastly different goals. Visual effects for feature films call for very high resolution simulations with very realistic appearance and motion. In addition, control over motion and appearance is necessary for artistic purposes such that physically impossible things become possible and that fluid motion is scriptable for user's specific needs. Foster and Fedkiw[7] and Enright *et al.*[6] showed that very realistic animation of water is possible. Unfortunately, existing methods are computationally very expensive and very slow to use. While great strides have been made to make fluid simulation more controllable[7], these grid-based methods do not fit well with current user-interaction paradigm used in modelling and animation tools. Furthermore, large scale problems such as stormy seas require large grids and are currently impractical to simulate. Also, multiphase flows, multiple fluids mixing, and sedimentary flows are not easy to model. Foster and Fedkiw[7] and Enright *et al.*[6] addressed some of the difficult problems with the grid-based methods using a hybrid representation: fragmentation and merging of fluids, numerical diffusion in convection computation, etc. There are alternative approaches to grid-based methods for simulating fluid flows: Large Eddy Simulation, vorticity confinement, vortex vethods, and particle methods. *Large-Eddy Simulation* (LES) adds an extra term to the Navier-Stokes equations to model the transfer of energy from the resolved scales to the unresolved scales. In *vortex methods*, large time steps are allowed and computational elements exist only where interesting flow occurs.

To address some of the deficiencies of the grid-based methods, we describe a particle interaction method for simulation of fluids. The underlying equations of fluid motion (the Navier-Stokes equations) are discretized using moving particles and their interactions. The particle method described is very simple and easy to implement. It fits better into current user-interaction paradigm and setting the simulation and controlling it is easy and intuitive. The method allows setting up the simulation (inflow and outflow boundaries, obstacles, forces) at coarse resolution (small number of large particles) to quickly preview the motion. Once the user is satisfied with the overall motion of the fluid flow, the simulation is run at high resolution to produce the final fluid motion. The computational elements (fluid particles) are only used in parts of the scene where they are required and number of computational elements can change during the course of the simulation. Therefore, if more detail is required in part of the scene, more particles can be put there to get finer detail. The method can handle mixing fluids seamlessly without special treatment, and multiphase flows where multiple fluids exist in liquid and gaseous form can also be simulated with minimal modifications. While particle-based methods have been presented in computer graphics literature

before, none of those methods dealt with incompressible fluids and water-like liquids.

Foster and Fedkiw[7] pointed out that it is difficult to create a smooth surface out of particles. Many particles are necessary to obtain a smooth surface. While many different methods can be used to create a surface from particles, we use a *level set* PDE method to reconstruct the surface. The level set surface is reconstructed on a grid whose resolution and computation are completely independent of the fluid simulation. On the other hand, for preview purposes, blending of potential fields around each particle can be used to give fast feedback on how the surface might look.

While the grid-based methods described by Foster and Fedkiw[7] and Enright *et al.*[6] provide impressive results, the particle-based method could provide an alternative for simulation and animation of variety of fluids with different physical properties while allowing user control and fast feedback at coarse resolutions.

2. Background And Previous Work

Simulation of fluids and their motion in computer graphics has been attempted with a variety of methods. We briefly overview methods relevant to the work presented in this paper. Early methods were geared towards simplifying the computation by using Fourier synthesis[18] or providing specialized solution for a specific problem[9, 25]. Kass and Miller[13] used height field to represent the fluid surface and used shallow water partial differential equations to describe the fluid motion. O'Brien and Hodgins[22] also used a height field representation coupled with a particle system to represent fluid motion with splashing that was missing in previous methods. Foster and Metaxas[8] realized the limitation of the height field representation and used a Marker-And-Cell (MAC) method[11] to solve the Navier-Stokes equations. Massless marker particles are put into the computational grid and advected according to the velocity field to track the surface. Their method was a true 3D method and was therefore able to simulate fluid pouring and splashing. Stam[28] simulated fluids using a semi-Lagrangian convection that allowed much larger time-steps while being unconditionally stable. Foster and Fedkiw[7] greatly improved the MAC method by using the level set approach for tracking the fluid interface. Enright *et al.*[6] further improved this method by introducing an improved method of tracking the water interface using *p*article level set and a new velocity extrapolation method. Enright *et al.*[5] describe this *thickened front tracking* method in more detail. Carlson *et al.* also utilized the MAC algorithm for animation of melting and solidifying objects[1].

Alternative methods of simulation fluid motion were described by using particle-based simulations. Miller and Pearce[19] simulate deformable objects with particle interactions based on the Lennard-Jones potential force. This force is strongly repellent when particles are close together and weakly attractive when particles are some distance apart. Terzopoulos *et al.*[33] pairs particles together to better simulate deformable objects. Tonnesen[34] improves particle motion by adding additional particle interactions based on heat transfer among particles.

Lucy[17] introduced a flexible gridless particle method called *smoothed particle hydrodynamics* (SPH) to simulate astrophysical problems including galaxial collisions and the dynamics of star formation. The fluid is modeled as a collection of particles, which move under the influence of hydrodynamic and gravitational force. SPH has recently been adapted to many relevant engineering problems, including heat and mass transfer, molecular dynamics, and fluid and solid mechanics. SPH is a flexible Lagrangian method[20, 21] that can easily capture large interface deformation, breaking and merging, and splashing. Desbrun and Cani-Gascuel[3] described a model of deformable surfaces and metamorphosis with active implicit surfaces. SPH method was used to compute particles motion and particles were coated with a potential field. Desbrun and Cani improved the particle model to simulate a variety of substances using a *state equation* to compute the dynamics of the substance[2]. Adaptive sampling of particles improved the computational efficiency of the method by subdividing particles where substance undergoes large deformation, and merging particles in less active areas. Stora *et al.*[29] also used smoothed particles to solve the governing state equation for animated lava flows by coupling viscosity and temperature field.

While the SPH method is flexible, it can only solve compressible fluid flow. Several extensions have been proposed to allow simulation of incompressible fluids with SPH. Recently, another gridless particle method called the *Moving-Particle Semi-Implicit* (MPS) was developed that solves governing Navier-Stokes equations for incompressible fluids[14]. The MPS method is capable of simulating a wide variety of fluid flow problems including phase transitions, multiphase flow, sediment-laden flows and elastic structures[37, 15, 12]. The computational algorithm in this paper is based on the MPS method.

3. Gridless Particle Method

The Moving Particle Semi-implicit method is a Lagrangian method of computing fluid motion. Contrary to the grid-based Eulerian methods where physical quantities are computed on fixed points in space, the computational elements in the MPS method are discrete number of *particles of fluid* followed in time. The MPS method is meshless. Given an arbitrary distribution of *interpolation points*, all problem variables are obtained from values at these points through an interpolation function (kernel). Interpolation points and fluid particles coincide.

GOVERNING EQUATIONS FOR INCOMPRESSIBLE FLOW. The motion of a fluid can be described by the Navier-

$\mathbf{u}$	Velocity
$\mathbf{r}$	Position
r	Distance between particles
r_e	Interaction radius
d	Number of space dimensions
t	Time
dt	Time step
w	Weight function
μ	Viscosity
σ	Surface tension
κ	Surface curvature
ρ	Density
n^0	Fluid particle density

Table 1: *Notation and important terms used in the paper.*

Stokes equations. If $\mathbf{u}$ is a velocity field of the fluid, the *continuity equation* states that the mass m is constant:

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

and thus enforces the incompressibility of the fluid. The *conservation of momentum* relates velocity and pressure:

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

where ρ is density of the fluid, p is the pressure, ν is the viscosity and $\mathbf{f}$ are forces. Other equations that describe molecular diffusion, surface tension, conservation of energy and many other relationships could also be written for a given fluid. Table 1 summarizes important terms and notation used in this paper.

3.1. Particle Interactions

In particle methods, mass, momentum and energy conservation equations are transformed to particle interaction equations. All interactions between particles are limited to a finite distance. The weight of interaction between two particles that are distance r apart is

$$w(r) = \begin{cases} \frac{r_e}{r} & \text{if } 0 \le r \le r_e \\ 0 & \text{if } r_e \le r \end{cases} \tag{3}$$

where r is the distance between two particles i and j,

$$r = |\mathbf{r}_j - \mathbf{r}_i|. \tag{4}$$

If all particles are allowed to interact, the complexity is $O(N^2)$. In contrast, if interaction radius is restricted, the complexity is only $O(NM)$, where M is the number of interacting particles[24] within the radius of interaction r_e. The particle number density n can be computed as

$$\langle n \rangle_i = \sum_{j \neq i} w(|\mathbf{r}_j - \mathbf{r}_i|).$$

The number of particles in a unit volume is approximated by the particle number density

$$\langle \rho_n \rangle_i = \frac{\langle n \rangle_i}{\int w(r) dv}.$$

For an incompressible fluid, the fluid density must be constant: n^0.

To solve the Navier-Stokes equations, *differential operators* on particles must be defined. Let ϕ and $\mathbf{u}$ be arbitrary scalar and vector quantities. The gradient $\nabla\phi$ is the average of scalar gradient between particle i and neighboring particle j:

$$\langle \nabla \phi \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \frac{\phi_j - \phi_i}{|\mathbf{r}_j - \mathbf{r}_i|^2} (\mathbf{r}_j - \mathbf{r}_i) w(|\mathbf{r}_j - \mathbf{r}_i|). \tag{5}$$

Similarly, the vector gradient $\nabla\mathbf{u}$ is the average of vector gradient between particle i and neighboring particle j:

$$\langle \nabla \cdot \mathbf{u} \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} w(|\mathbf{r}_j - \mathbf{r}_i|). \tag{6}$$

The Laplacian operator ∇^2 is derived from the concept of diffusion. It can be seen as if a fraction of a quantity at particle i is distributed to neighboring particle j:

$$\langle \nabla^2 \phi \rangle_i = \frac{2d}{\lambda n^0} \sum_{j \neq i} (\phi_j - \phi_i) w(|\mathbf{r}_j - \mathbf{r}_i|), \tag{7}$$

where λ:

$$\lambda = \frac{\int_V w(r) r^2 dv}{\int_V w(r) dv}. \tag{8}$$

Note that this model does not require any spatial connectivity information. When particles move around no special care or reconfiguration is needed. Grid-based method suffer from numerical breakdown when computational mesh gets tangled due to large interface deformations. Arbitrary materials and surfaces can all be represented with particles. Any complex boundaries and objects (stationary or moving) can be described with particle arrangements. This allows for simulation of complex problems in a simple unified manner without special cases.

3.2. MPS Method

The Navier-Stokes equations are solved by the semi-implicit MPS method. For every time step dt, the forces and viscosity in the momentum conservation equation are computed explicitly. Temporary particle locations $\mathbf{r}^*$ and velocities $\mathbf{u}^*$ are computed from the positions $\mathbf{r}^n$ and velocities $\mathbf{r}^n$ from the previous time step n as follows:

$$\mathbf{u}^* = \mathbf{u}^n + \frac{dt}{\rho}\left(\mu \nabla^2 \mathbf{u}^n + \sigma(\kappa \cdot \mathbf{n})^n + \rho \mathbf{g}\right), \tag{9}$$

and

$$\mathbf{r}^* = \mathbf{r}^n + \mathbf{u}^* dt. \tag{10}$$

The surface tension model and computation is described in Appendix A. After this step, the incompressibility of the fluid is temporarily violated. The temporary particle density n^* is not equal to n^0. The particle density n^* needs to be modified by n' such that the continuity equation is satisfied. The particle density n' is related to modification of the velocity $\mathbf{u}'$:

$$\frac{1}{dt}\frac{n'}{n^0} = \nabla \cdot \mathbf{u}'. \tag{11}$$

The modification velocity $\mathbf{u}'$ is obtained from the implicit pressure term in the momentum conservation equation:

$$\mathbf{u}' = -\frac{dt}{\rho}\nabla p^{n+1}. \tag{12}$$

Note that this is the same as in the grid-based methods such as MAC. By substituting equations 11 and 12 into

$$n^0 = n^* + n', \tag{13}$$

a Poisson equation for pressure is obtained:

$$\langle \nabla^2 p^{n+1} \rangle_i = -\frac{\rho}{dt}\frac{\langle n^* \rangle_i - n^0}{n^0}. \tag{14}$$

The right hand side of equation 14 is analogous to the divergence of the velocity vector. Equation 14 is solved by using the Laplacian differential operator and discretizing it into a system of linear equations. The matrix representing these linear equations is sparse and symmetric; therefore it can be solved using the conjugate gradient method. Once the pressure p^{n+1} is computed, the correction velocity $\mathbf{u}'$ also becomes known:

$$\mathbf{u}' = -\frac{dt}{\rho}\langle \nabla p^{n+1} \rangle. \tag{15}$$

New particle velocities and positions that satisfy both conservation of mass and momentum can then be updated:

$$\mathbf{u}^{n+1} = \mathbf{u}^* + \mathbf{u}' \tag{16}$$

$$\mathbf{r}^{n+1} = \mathbf{r}^n + \mathbf{u}^{n+1} dt. \tag{17}$$

BOUNDARY CONDITIONS. The particle density number decreases for particles on the free surface. A particle which satisfies a simple condition

$$\langle n^* \rangle_i < \beta n^0, \tag{18}$$

is considered on the free surface. In this paper, we use $\beta = 0.97$. Intuitively, this makes sense because the weighting kernel will span the fluid interface. Pressure $p = 0$ (or atmospheric pressure, if applicable) is applied to these particles on the free surface in the pressure calculation. Solid boundaries such as walls or other fixed objects are represented by fixed particles. Velocities are always zero at these particles. Three layers of particles are used to represent fixed objects to ensure that particle density number is computed accurately. Note that there is no explicit collision detection between particles. The pressure that is computed at fixed particles essentially repels the fluid particles from the fixed objects. Therefore, no special cases are therefore needed and arbitrary object-fluid configurations can be handled in the same computation seamlessly.

Algorithm 1 The Moving Particle Semi-Implicit (MPS) algorithm.

Initialize fluid: $\mathbf{u}^0$, $\mathbf{r}^0$
for each time step dt
  Compute forces and apply them to particles. Find temporary particle positions and velocities $\mathbf{u}^*$, $\mathbf{r}^*$
  Compute particle number density n^* using new particle locations r^*
  Set up and solve Poisson pressure equation using Conjugate Gradient method
  Compute velocity correction $\mathbf{u}'$ from the pressure equation
  Compute new particle positions and velocities:
  $\mathbf{u}^{n+1} = \mathbf{u}^* + \mathbf{u}'$
  $\mathbf{r}^{n+1} = \mathbf{r}^* + \mathbf{u}' dt$
end for

The basic MPS algorithm is summarized in Algorithm 3.2.

DISADVANTAGES. It is worth noticing that the gridless MPS method has several disadvantages. Since it is a Lagrangian method, inflow and outflow of fluid is not allowed. However, it can be easily combined with the Eulerian approach to handle inflow and outflow. We will describe a simple hybrid method in subsection 3.3. Conservation of scalars (energy, etc.) is not guaranteed. If one truly cares about conservation, the finite volume methods based on integrals in the cells are good for conservation. Also, large aspect ratio is impossible at the moment. Note however that this is also a problem for the most-advanced finite volume methods. The biggest disadvantage of using particle method is the question of surface representation. We have particles that accurately represent the fluid motion and other interesting quantities, but for rendering the fluid a surface representation is needed. We describe the surface representation and extraction in section 4.

3.3. MPS-MAFL Method

As discussed in the previous subsection, one of the main problems with a purely Lagrangian approach is that inflow and outflow of fluid cannot be handled. Furthermore, local resolution enhancement is hard, because fluid particles move around. If more particles are introduced to improve resolution, they will soon move to different locations. In order to alleviate problems of purely Lagrangian method in the MPS method, a gridless hybrid method has been developed[37]. The method consists of three steps:

1. Lagrangian Phase: the MPS method
2. Reconfiguration Phase: particle positions are reconfigured

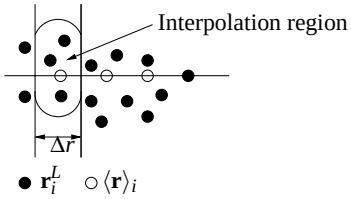

Figure 1: *Directional local grid. A one-dimensional grid is created in the particle's streamline direction. The quantities are only interpolated in the cut disk area. (After Heo* et al.[12]*)*

3. Eulerian Phase: particle convection is computed on a one-dimensional grid

The Lagrangian phase is exactly the same as described in the MPS method. We denote the particle positions and velocities obtained after this phase as $\mathbf{u}^L$ and $\mathbf{r}^L$. The reconfiguration phase and convection (Eulerian) phase are described next.

Reconfiguration Phase

In order to correctly handle inflow and outflow boundaries and deal with irregular distribution of particles, the particle positions have to be reconfigured. The computation points are redistributed considering the boundaries. Points that belong to a fixed boundary (fixed objects, walls, etc.) or inlet or outlet boundary, should go back to their original positions $\mathbf{r}^n$. The moving boundary can be traced through Lagrangian motion of points describing the free surface without computing the convection term: $\mathbf{r}_s^{n+1} = \mathbf{r}_s^L$ where subscript s denotes that the point is on the surface. In practice, it is likely that the points on the surface will cluster together. To fix this problem, we make sure that the particles on the moving boundary are an equal distance apart. The reconfiguration phase is equivalent to remeshing in grid-based methods. However, it is much easier in the particle-based methods because only the particle locations need to be adjusted. Note that the number of fluid particles can vary. Therefore, higher particle concentration can be used in areas that require higher accuracy.

The computation point $\mathbf{r}^{n+1}$ at the new time step is determined arbitrarily and the velocity of the computing point $\mathbf{u}^c$ is

$$\mathbf{u}^c = \frac{\mathbf{r}^{n+1} - \mathbf{r}^n}{dt}. \tag{19}$$

The convection velocity is then given by

$$\mathbf{u}^a = \frac{\mathbf{r}^L - \mathbf{r}^n}{dt} - \frac{\mathbf{r}^{n+1} - \mathbf{r}^n}{dt} = -\frac{\mathbf{r}^{n+1} - \mathbf{r}^L}{dt}. \tag{20}$$

Eulerian Phase

After arbitrary convection velocity $\mathbf{u}^a$ is computed, properties at $\mathbf{r}^{n+1}$ are computed by interpolation of physical property f (flow velocity, temperature, etc.):

$$f(t+dt, \mathbf{r}_i^{n+1}) = f(t, \mathbf{r}_i^L - \mathbf{u}_i^a dt). \tag{21}$$

If the number of computation points changes during the reconfiguration phase, the physical quantity f is computed by

$$f(t+dt, \mathbf{r}_i^{n+1}) = f^L(t, \mathbf{r}_o^L - \mathbf{u}_i^a dt), \tag{22}$$

where $\mathbf{r}_o^L$ is the closest point to $\mathbf{r}_i^{n+1}$.

We follow a simple meshless advection method, MAFL, proposed by Yoon *et al.*[37]. Other convection methods[32] can easily be substituted if so desired. There are four stages in the convection phase computation:

1. Generate a one-dimensional directional grid,
2. Locally interpolate physical quantities,
3. Apply any high-order convection scheme,
4. Filter the result to prevent oscillatory solutions.

DIRECTIONAL GRID GENERATION. Because the fluid properties are changing along the streamline (direction of the velocity vector), the convection problem is a one-dimensional problem if a grid is generated in the flow direction. Figure 1 shows the directional grid. The number of grid points used in computation depends on the convection difference scheme used.

LOCAL INTERPOLATION. At a local grid point, the physical properties $\langle f \rangle_k$ are interpolated from the neighboring computing points f_j^L using the weight:

$$\langle f \rangle_k = \frac{\sum_j f_j^L w(|\mathbf{r}_j^L - \langle \mathbf{r} \rangle_k|, r_{e,k})}{\sum_j w(|\mathbf{r}_j^L - \langle \mathbf{r} \rangle_k|, r_{e,k})} \text{for k = -2,-1,1.} \tag{23}$$

CONVECTION SCHEME. Any difference scheme can be used in the convection phase. If the first order upwind scheme is applied to local grid points, only two points are considered:

$$\tilde{f}_i^{n+1} = f_i^L - q(f_i^L - \langle f \rangle_{-1}), q = \frac{|\mathbf{u}^a| dt}{dr}. \tag{24}$$

The second order QUICK[16] scheme uses four points (two upstream and one downstream) and yields

$$\tilde{f}_i^{n+1} = f_i^n - q(\frac{1}{8} f_{i-2}^n - \frac{7}{8} f_{i-1}^n + \frac{3}{8} f_i^n + \frac{3}{8} f_{i+1}^n), q = \frac{|\mathbf{u}^a| dt}{dr}. \tag{25}$$

FILTERING. Higher order schemes often result in oscillatory solutions. A filtering scheme can be applied to prevent overshooting and undershooting. Minimum and maximum limits are computed at each time step and the interpolants are bounded by them:

$$f_i^{n+1} == \begin{cases} \tilde{f}_i^{n+1} & \min(f_i^n) \le \tilde{f}_i^{n+1} \le \max(f_i^n) \\ \min(f_i^n) & \tilde{f}_i^{n+1} < \min(f_i^n) \\ \max(f_i^n) & \tilde{f}_i^{n+1} > \max(f_i^n) \end{cases} \tag{26}$$

Higher order schemes can exhibit numerical instability if there is a large change in the number of particles and their locations.

The hybrid MPS-MAFL algorithm that allows arbitrary inflow and outflow of fluid, and arbitrary addition and

Algorithm 2 The hybrid MPS-MAFL algorithm.

Initialize fluid
for each time step *dt*
  Lagrangian phase: the same as MPS algorithm (Algorithm 1)
  Reconfiguration phase: determine the positions of computing points and convection velocities
  Create a one-dimensional local grid in the streamline direction
  Interpolate physical properties within the particle neighborhood
end for

removal of computation points is summarized in Algorithm 3.3. The interested reader is referred to papers by Heo *et al.*[12] and Koshizuka *et al.*[15] to learn more about MPS-MAFL methods.

3.4. Multifluid Flow

It is straightforward to extend the MPS and MPS-MAFL models described in sections 3.2 and 3.3 for multifluid and multiphase flows[27]. Let $\mathbf{u}_{\xi,i}$ denote the velocity of a fluid particle i of type ξ, and $\mathbf{r}_{\xi,i}$ be the position of the fluid particle. The temporary velocity $\mathbf{u}^*_\xi$ that includes the diffusion, gravity, and surface tension is similar to equation 9:

$$\mathbf{u}^*_\xi = \mathbf{u}^n_\xi + \frac{dt}{\rho_\xi}\{\mu_\xi \nabla^2 \mathbf{u}^n_\xi + \sigma_\xi(\kappa_\xi \cdot \mathbf{n}_\xi)^n + \rho_\xi \mathbf{g}\}. \qquad (27)$$

Other forces acting on the fluid can be included. The implicit pressure computation is similar to the single fluid MPS method. If the density of mixing fluids is on the same order of magnitude, the pressure computation is done in a single step as before. In order to avoid numerical instabilities when multiple fluids have drastically different densities (e.g. water and air), the pressure computation for each fluid is done separately. First, the pressure computation is performed as if all particles are gas particles. In the second step, the computed pressure for gas particles is applied to the interface of the liquid particles. This is an iterative process: if the particle velocity and position converge we proceed to next step, otherwise we repeat the pressure computation again until convergence.

4. Surface Reconstruction

In this section, we introduce the notation of level set methods and discuss our approach to surface reconstruction from particles. Let $\mathbf{x}(t)$ be the set of points on a deforming surface at time t. We represent this deforming surface implicitly as

$$S = \{\mathbf{x}(t) \mid \phi(\mathbf{x}(t), t) = 0\}, \qquad (28)$$

where $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ is the embedding function. Surfaces defined in this way divide a volume into two parts: inside ($\phi > 0$) and outside ($\phi < 0$). It is common to choose ϕ to be the signed distance transform of S, or an approximation thereof. The motion of S is controlled indirectly by modifying ϕ with a PDE. This family of PDEs and the upwind scheme for solving them on a discrete grid is the methods of *level sets* proposed by Osher and Sethian[23]. In this paper, we consider level set PDEs of the form

$$\partial\phi(\mathbf{x})/\partial t = -||\nabla\phi||\,(F(\mathbf{x}) + \mu H(\mathbf{x})), \qquad (29)$$

where F is a force term , H is the mean curvature of the level set interface and μ defines the relative weights of the two terms. The mean curvature term guarantees the smoothness of the interface by favoring surfaces of smaller area over surfaces of larger area[26]. The force term F is designed to make the level set interface track the moving particles. We fix $\mu = 1$ for all of our experiments. Note that the setting of the parameter μ is a trade-off between overall surface smoothness and the capturing of surface features defined by particle positions.

For a given frame, the particle simulation provides a set of particle locations, radii and velocities $\{\mathbf{r}_i, r_i, \mathbf{u}_i\}_{i=1}^N$. Let F be a sum over all the particles of a kernel function f, i.e.

$$F(\mathbf{x}) = T + \sum_{i=1}^{N} f(\mathbf{x}, \mathbf{r}_i, r_i, \mathbf{u}_i), \qquad (30)$$

where T is a constant threshold. Let $d_i(\mathbf{x})$ denote the Euclidean distance from the center of particle i to the point $\mathbf{x}$ in space. Since the particles have a finite size, if we define $f_i(\mathbf{x})$ to be 1 for $d_i(\mathbf{x}) \leq r_i$ and 0 outside, then F represents the number of particles at any point in space. If we also choose $T = -0.5$, the points in space that satisfy $F = 0$ will approximately represent the surface defined by the particles. However, this binary choice for f leads to a very rough looking surface, and the individual particles are easily recognizable in the reconstruction. The curvature smoothing term in (equation 29) can not compensate for this large-scale roughness. What is needed is a smoother choice for f that provides an interpolation between particles. Consider the following

$$f_i(\mathbf{x}) = \frac{1}{1 + |d_i(\mathbf{x})/r_i|^k}, \qquad (31)$$

where $k = 2$. This function falls off smoothly as the distance to the particle increases, and is well-behaved as $d \rightarrow 0$. Because, $f > 0$ for $d_i(\mathbf{x}) > r_i$, F accumulates to larger quantities than with the previous case. This necessatiates choosing a lower T value; we find that $T = -2$ is suitable. We also experimented with $k = 1$ and $k > 2$, but these choices resulted in oversmoothing and not enough interpolation, respectively.

The interpolation obtained from (equation 31) is isotropic; therefore, it has a thickening effect on the surface due to interpolation in the direction perpendicular to the surface. A further modification can be made to solve this problem by allowing more interpolation in the physical surface tangent plane, and less interpolation perpendicular to this plane. We use the assumption that the velocities of the particles will be approximately in the tangent plane. Let $d_i^{\nu}(\mathbf{x})$ and $d_i^{\perp}(\mathbf{x})$ be

defined as :

$$d_i^v(\mathbf{x}) = \left| (\mathbf{x}-\mathbf{r}_i) \cdot \frac{\mathbf{u}_i}{\| \mathbf{u}_i \|} \right| , \; d_i^\perp(\mathbf{x}) =$$
$$= \| (\mathbf{x}-\mathbf{r}_i) - d_i^v \frac{\mathbf{u}_i}{\| \mathbf{u}_i \|} \| .$$

Let s_{max} be the largest particle speed for the given frame. Then, we define the modified distance to a particle as

$$d_i(\mathbf{x}) = \left(1 - \frac{\| \mathbf{u}_i \|}{2 s_{max}}\right) d_i^v(\mathbf{x}) + \left(1 + \frac{\| \mathbf{u}_i \|}{2 s_{max}}\right) d_i^\perp(\mathbf{x}). \qquad (32)$$

Using this definition of distance in (equation 31) has the effect of elongating the influence of a particle along its velocity vector and shrinking it in all other directions.

The surface reconstruction algorithm starts with the particle information for the first frame. Before starting to evaluate (equation 29), we need an initialization for ϕ. After computing F for the first frame, the initialization is obtained from $F = 0$. Then, we iterate (equation 29) using an upwind, sparse-field implementation[35] until convergence. The computation only occurs in grid cells that are on or near the surface. Convergence is reached when the change to ϕ per time step falls below a pre-determined threshold. At this point, we save the state of the surface for generating the animation. Then, F is constructed for the particle information contained in the next frame, and we continue iterating (equation 29) without reinitialization. In other words, the surface result from the previous frame acts as the initialization. This approaach guarantees the continuity of the surface models produced for consecutive frames. Note that the level set surface reconstruction step is solved on a grid whose resolution is completely independent from the fluid simulation. Once the fluid simulation is computed, the surface reconstruction is done at arbitrary resolution.

5. Results and Discussion

The MPS and MPS-MAFL methods described in this paper methods are straighforward to implement. For efficient computation, a spatial data structure that quickly finds particles in a neighborhood is desired although not necessary. The Poisson pressure equation can be efficiently solved with a Conjugate Gradient (CG) method. In our implementation, we use the CG method with an Incomplete Cholesky preconditioner. We use the *SparseLib++* library[4] for computing the Poisson pressure equation.

We show several examples of fluid simulation computed with the MPS and MPS-MAFL methods. All simulations and rendering were performed on a Pentium IV 1.7 Ghz with 512 Mb of memory running Linux operating system. Videos of animations discussed in this sections accompany this paper.

The computational method is fairly efficient and allows simulating about 10,000 particles per timestep per second. This is fast enough to get an instantaneous feedback on whether the fluid simulation will run as desired. After we set up the scene (objects, obstacles, forces, fluid properties) and initialize fluid particles, we run the fluid simulation at low resolution to get feedback. After we set all simulation parameters (particle size, interaction radius, etc.) and forces (gravity, drag, surface tension, etc.), the final simulation is run at high resolution. The main bottleneck in the computation is the Poisson pressure equation computation and the computation time ultimately depends on the number of particles in the simulation. As a part of future work, it would be beneficial to parallelize the Conjugate Gradient algorithm to further speed up computation time on parallel architectures.

Corridor Flood

We simulated a simple flood in an underground corridor. We modeled the corridor with a small number of polygons and converted the polygonal scene into the particle representation. Each polygon was represented with three layers of fixed particles. The inflow of water was simulated by positioning a virtual square inflow boundary that was turned on for a short period of time. This could also be simulated as a water column collapse. The total number of fluid particles in the scene is about 100,000. The only force acting on particles was gravity. Surface tension was not included in this simulation. The computation time for the final simulation was about 3 minutes per frame. Surface reconstruction took 10 minutes per frame (volume size 459x74x374). Figure 2 shows several frames from the simulation. The final surface was rendered using a Monte Carlo path tracer with 10 area light sources. The rendering time per frame was about 20 minutes. Since the MPS method is fully Lagrangian, the computational elements (fluid particles) were only present in the part of the scene where they were needed. Note that because the corridor is L-shaped, this provides some computational savings over the grid-based methods. In grid-based approaches, about 70 percent of the space would be wasted. A larger grid would be needed to accommodate the computation, therefore yielding larger computation time and memory requirements.

Box Filling

In this simulation, we fill a box with a fluid that is being poured from a source with three nozzles. The fluid does not fall directly into the box. A polygonal obstacle set near the top of the box at an angle obstructs the flow. The fluids from the nozzles first hit the obstacle surface and then the side of the box. The fluid was simulated with about 150,000 fluid particles. The simulation time is about 4 minutes per frame. Gravity, surface tension and drag force were acting upon fluid particles during the simulation. The fluids from the three nozzles have the same physical properties. Surface reconstruction took 30 minutes per frame (volume size 197 x 283 x 256). Figure 3 shows several frames from the box filling simulation. The example animation was being rendered

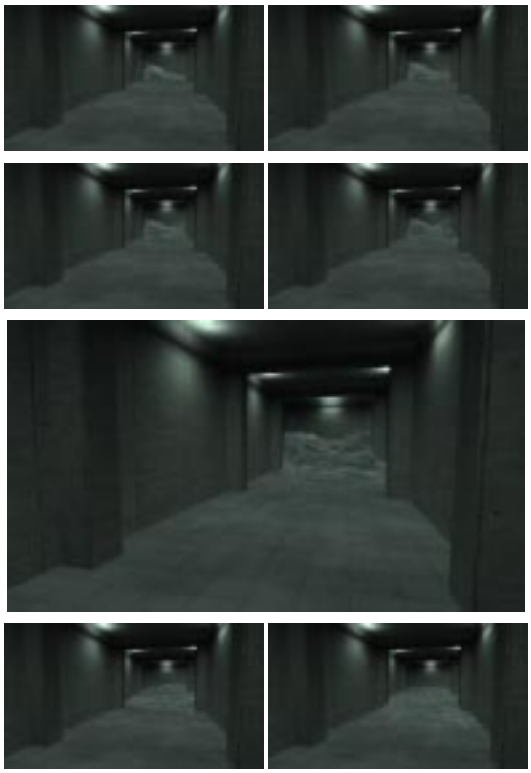

Figure 2: *Corridor flood simulation. The fluid motion is simulated by 100,000 fluid particles. The simulation time is about 3 minutes per frame.*

with a raytracer. Approximate rendering time was about 5 minutes per frame.

Mixing fluids

In this simulation, we fill a box with two fluids that have very different densities and viscosities. First, we start filling the box with a water-like fluid. After some time, we start filling the box with the second oil-like fluid. The fluids start interacting and mixing. The second fluid ends on top of the first fluid as expected from the physical properties of the fluids. Gravity and surface tension were applied to both fluids. About 80,000 fluid particles represent both fluids. The simulation time was 3 minutes per frame. Surface reconstruction took 10 minutes per frame per fluid (volume size 245x245x274). Figure 4 shows three frames from this simulation. Observe the mixing and interactions between the two fluids. The images were being rendered with a raytracer. Approximate rendering time was 5 minutes per frame. The second-fluid has oil-like physical properties. For visualization purposes we rendered this fluid opaquely to show the separation between the two fluids. Some artefacts (e.g. round

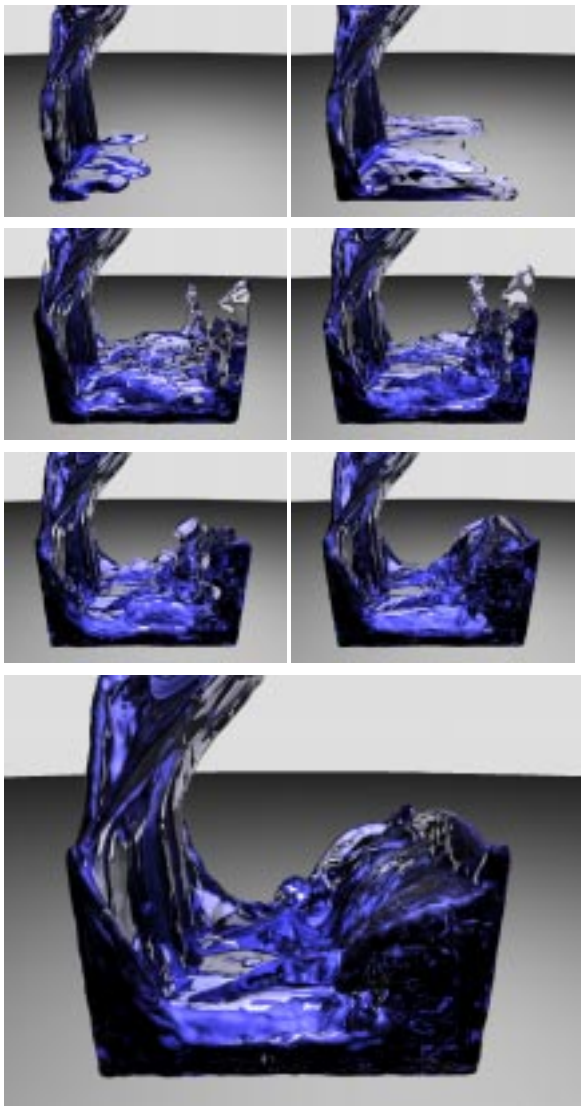

Figure 3: *A source with three nozzles filling a box. The fluid motion is simulated by 150,000 fluid particles. The fluid is being emitted from three nozzles that hit an obstacle surface set near the top of the box.*

boundary, non-perfectly smooth surface) resulting from the surface reconstruction are visible.

6. Conclusion

We described a particle-based method for simulation fluid motion. It is based on a particle discretization of the differential operators to solve the Navier-Stokes equations. It is suited for simulating a wide variety of fluid flows including multifluids, multiphase flows and medium scale problems such as the corridor flood. Because of the Lagrangian nature

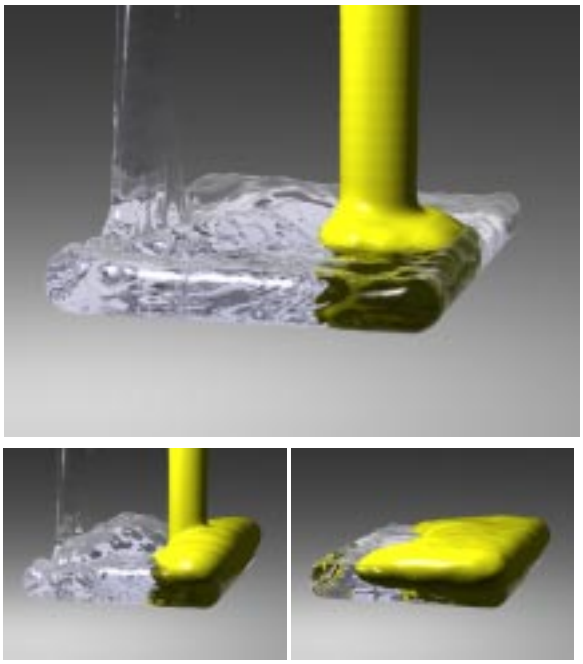

Figure 4: *Two fluids mixing in a box. The box is being filled with two fluids with drastically different physical properties (density and viscosity). After interaction and mixing, the second fluid ends up on top of the first fluid. About 80,000 particles were used to compute the fluid motion.*

of the method, no grids are needed. The method is adaptive as it allows arbitrary addition and removal of computation points during the simulation. The described method is also well suited for the current human-interaction paradigm used in commercially available modelling and animation software. The fluid simulation can be easily directed (e.g. fluid moves on a path) and scripted (fluid reaches specified destination), because the particle interactions are easily controlled while ensuring physical correctness and plausibility of motion. While particle-based fluid simulations (*ad hoc* and semi-physically-based) were described in the graphics literature before, the methods presented in this paper is governed by the Navier-Stokes equations and works with incompressible fluids.

There are numerous possibilities for future work and extensions of the described method. The main computational bottleneck is the Poisson pressure computation. By parallelizing the conjugate gradient computation the method could be much faster and could potentially become more interactive. More accurate interpolation schemes could be used if more accurate simulations are desired[32]. Solid-fluid interactions (e.g. fluid displacing a solid object) and sediment flows could easily be added to the existing method[10]. The MPS-MAFL method could be extended for adaptive simulations similar to Desbrun and Cani[2]. Particles radii would be small in areas where large deformations of the interface occur. The Eulerian step in the MPS-MAFL ensures that small particles remain in areas of interest and would not get advected to other regions. The MPS particle method could also be applied to simulate solid dynamics[36].

The main problem with the particle-based method remains surface generation. While the surface is easily being tracked with fluid particles, it is hard to create a surface from the fluid simulation. In this work we use a level set method to obtain surface for rendering purposes. The method produces good results for most cases, but it also has some inherent problems with creating sharp boundaries when the fluid is in contact with a solid object or another fluid. As part of the future work, we want to try the hybrid approach by Enright *et al.*[5] for tracking and evolving the surface. An alternative would be to still use the level set to represent the fluid interface, but use particles to directly evolve the level set[30, 31]. By removing the level set surface extraction as a separate step, all grid-based computations would be removed entirely.

Acknowledgements

We thank Marko Dabrović and John Moores for discussions and rendering help and suggestions. Part of this work was supported by the Office of Naval Research under grant N00014-01-10033 and the National Science Foundation under grant CCR0092065.

Appendix A: Surface Tension Model

The momentum equation containing the surface tension is:

$$\rho\frac{\partial \mathbf{u}}{\partial t} = -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{g} + \sigma\kappa\delta\mathbf{n} \tag{33}$$

where σ is the surface tension coefficient, κ is the curvature of the interface, δ is the delta function, and $\mathbf{n}$ is the surface normal. Surface tension is calculated for the particles that are regarded as on the interface. Another particle density n_i^{st1} is computed at these particles:

$$\langle n\rangle_i^{st1} = \sum_{j\neq i} w^{st1}(|\mathbf{r}_j - \mathbf{r}_i|) \tag{34}$$

$$w^{st1}(r) = \begin{cases} 1 & \text{if } 0 \le r \le r_e^{st} \\ 0 & \text{if } r_e^{st} \le r \end{cases} \tag{35}$$

where r_e^{st} is the interaction radius for surface tension model. The particles regarded as on the surface of the interface are found in thickness d^{st}. Within the thickness d, the interior particles have larger particle density n_i^{st1} than exterior particles. This leads to errors in curvature computation. A second particle density excluding outside particles is computed:

$$\langle n\rangle_i^{st2} = \sum_{j\neq i} w^{st2}(|\mathbf{r}_j - \mathbf{r}_i|) \tag{36}$$

$$w^{st2}(r) = \begin{cases} 1 & \text{if } 0 \le r \le r_e^{st} \text{ and } n_j^{st1} > n_i^{st1} \\ 0 & \text{otherwise} \end{cases} \quad (37)$$

The curvature of the interface is then computed as:

$$\kappa = \frac{1}{R} = \frac{2\cos\theta}{r_e^{st}} \quad (38)$$

$$2\theta = \frac{n_i^{st2}}{n_0^{st1}}\pi, \quad (39)$$

where n_0^{st1} is constant and is computed when the interface is flat (curvature is zero).

Appendix B: Drag Force Model

The drag force due to the permeable layer is:

$$\mathbf{f} = -\frac{3}{4}\frac{C_D}{d_0}|\bar{\mathbf{u}}_i|\bar{\mathbf{u}}_i \quad (40)$$

$$w_u(r) = \begin{cases} 1/\sum w_u(r) & \text{if } r \le \alpha d_0 \\ 0 & \text{if } r > \alpha d_0 \end{cases} \quad (41)$$

$$\bar{\mathbf{u}}_i = \sum \mathbf{u}_j w_u(|\mathbf{r}_{ij}|), \quad (42)$$

where C_D is drag coefficient, $\bar{\mathbf{u}}_i$ is spatially averaged velocity of neighborhood particles, α is model constant and d_0 is diameter of a fluid particle.

References

1. Mark Carlson, Peter J. Mucha, R. Brooks Van Horn III, and Greg Turk. Melting and flowing. In *ACM SIGGRAPH Symposium on Computer Animation*, 2002.
2. Mathieu Desbrun and Marie-Paule Cani. Space-time adaptive simulation of highly deformable substances. Technical report, INRIA, 19990.
3. Mathieu Desbrun and Marie-Paule Cani-Gascuel. Active implicit surface for animation. In *Graphics Interface*, 1998.
4. J. Dongarra, A. Lumsdaine, R. Pozo, and K. Remington. A Sparse Matrix Library in C++ for High Performance Architectures. In *Proceedings of the Second Object Oriented Numerics Conference*, pages 214–218, 1994.
5. D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, (183):83–116, 2002.
6. Douglas P. Enright, Stephen R. Marschner, and Ronald P. Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics*, 21(3):736–744, July 2002.
7. Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, pages 23–30, August 2001.
8. Nick Foster and Demitri Metaxas. Realistic animation of liquids. In *Graphics Interface '96*, pages 204–212, 1996.
9. Alain Fournier and William T. Reeves. A simple model of ocean waves. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 75–84, 1986.
10. Hitoshi Gotoh, Tomoki Shibahara, and Tetsuo Sakai. Sub-particle-scale turbulence model for the MPS method. *Computational Fluid Dynamics Journal*, 9(4):339–347, 2001.
11. F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids*, 8:2182–2189, 1965.
12. S. Heo, S. Koshizuka, and Y. Oka. Numerical analysis of boiling on heat-flux and high subcoling condition using MPS-MAFL. *Comput. Fluid Dynamics J.*, 45:2633–2642, 2002.
13. Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 49–57, August 1990.
14. S. Koshizuka, H. Tamako, and Y. Oka. A particle method for incompressible viscous flow with fluid fragmentation. *Comput. Fluid Dynamics J.*, 29(4), 1996.
15. S. Koshizuka, H. Y. Yoon, D. Yamashita, and Y. Oka. Numerical analysis of natural convection in a square cavity using MPS-MAFL. *Comput. Fluid Dynamics J.*, 30(8):485–494, 2000.
16. B. P. Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Comp. Methods Appl. Mech. Eng.*, 19:59–98, 1979.
17. L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82(12):1013–1024, Dec 1977.
18. G. A. Mastin, P. A. Watterberg, and J. F. Mareda. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications*, 7(3):16–23, March 1987.
19. Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.
20. J. J. Monaghan. Smoothed particle hydrodynamics. *Ann. Rev. Astron. Astrophys.*, 30(2):543–574, 1992.
21. J. J. Monaghan. Simulating free surface flows with SPH. *Journal of Computational Physics*, (110):399–406, 1994.
22. J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *Computer Animation '95*, pages 198–205, April 1995.
23. S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.
24. Allen M. P. and Tildesley D. J. *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.
25. Darwyn R. Peachey. Modeling waves and surf. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 65–74, 1986.
26. Guillermo Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, 2001.
27. N. Shirakawa, H. Horie, Y. Yamamoto, and S. Tsunoyama. Analysis of the void distribution in a circular tube with the two-fluid particle interaction model. *Journal of Nuclear Science and Technology*, 38(6):293–402, June 2001.
28. Jos Stam. Stable fluids. In *Proceedings of SIGGRAPH 99*, pages 121–128, August 1999.
29. D. Stora, P.O. Agliati, M.P. Cani, F. Neyret, and J.D. Gascuel. Animating lava flows. In *Graphics Interface*, Kingston, Canada, June 1999.
30. John Strain. Semi-Lagrangian Methods for Level Set Equations. *J. Comput. Phys.*, 151:498–533, 1999.
31. John Strain. A Fast Modular Semi-Lagrangian Method for Moving Interfaces. *J. Comput. Phys.*, 161:512–536, 2000.
32. Nobuatsu Tanaka. Hamiltonian particle dunamics, CIVA-particle method and symplectic upwind scheme. *Computational Fluid Dynamics Journal*, 9(4):384–393, 2001.
33. Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models (from goop to glop). In *Proceedings of Graphics Interface '89*, pages 219–226, June 1989.
34. David Tonnesen. Modeling liquids and solids using thermal particles. In *Proceedings of Graphics Interface '91*, pages 255–262, June 1991.
35. Ross T. Whitaker. Algorithms for implicit deformable models. In *Fifth International Conference on Computer Vision*. IEEE Computer Society Press, 1995.
36. S. Koshizuka Y. Chikazawa and Y. Oka. A particle method for elastic and visco-plastic structures and fluid-structure interactions. *Computational Mechanics*, 27:97–106, 2001.
37. H. Y. Yoon, S. Koshizuka, and Y. Oka. A particle gridless hybrid method for incompressible flows. *Int. J. Numer. Meth. Fluids*, 29(4), 1996.