

Multi-phase Flow Dynamics with Volume Preservation*

ROMAN ĎURIKOVIČ** and KATSUHIRO NUMATA

Abstract

In this research we focus on a technique for animating incompressible viscous multi-phase flows, i.e. flow of fluid mixtures, with surface tension, where each fluid can have a different density and viscosity. The main techniques used to solve Navier-Stokes equations in our approach are the projection method on Marker-and-Cell (MAC) grid, Volume-of-Fluid (VOF) method, the continuum surface force method and the donor-acceptor method. The main idea of computing a surface force from estimated curvature of a fluid surface is extended for multiple fluids in this work. A modification of donor-acceptor method for multiple fluids is a novel idea of the paper. The proposed donor-acceptor method prevents the mass loss in comparison with alternate methods such as the level set method.

Mathematics Subject Classification 2000: I.3.6, I.3.7

Additional Key Words and Phrases: Multi-phase flow, fluid animation, VOF, MAC, dynamics

1. INTRODUCTION

We see fluid motions on a daily basis and the phenomena shows very complex behaviors. One of the goals of CG animation is to produce highly realistic motion of natural phenomena. The problem is that many of the methods can treat only a homogeneous fluid i.e. one fluid. We often see rising bubbles or flow of muddy water, such flows involve several fluids. Such animations remain as challenging tasks in computer graphics area. In this research, we adopted multi-phase CFD (Computational Fluid Dynamics) technique to solve the problem.

The main techniques used to solve Navier-Stokes equations in our approach are the projection method on Marker-and-Cell (MAC) grid, Volume-of-Fluid (VOF) method, the continuum surface force method and the donor-acceptor method. The VOF method is very efficient and accurate scheme to track interfaces of fluids. Our method takes special care in computing fluid fluxes to preserve sharp interface between fluids. The VOF scheme also enables fast interface construction. We directly construct fluid interfaces from VOF values using the marching cubes algorithm. The interfaces are composed of polygon meshes and rendered.

In this work we explore the main idea of computing a surface force from estimated curvature of a fluid surface and extend it for multiple fluids. Another novel idea is

* This research was sponsored by grants from the EU-FP6-MC-040681-APCOCOS.

** e-mail: roman.durikovic@fmph.uniba.sk Also with Faculty of Mathematics, Physics and Informatics, Comenius University, Mlynska dolina, 842 48 Bratislava, Slovakia

a modification of donor-acceptor method for multiple fluids. The proposed donor-acceptor method prevents the mass loss in comparison with alternate methods such as the level set method.

The outline of the paper is as follows. The next section describes the previous work in the field of CFD for purpose of animation and simulation. Section 3 gives an overview of the general algorithm for fluid dynamics. The novel approaches of our method are described in Section 4. The most interesting aspects of the paper the estimation of surface tension and the proposed donor-acceptor method are described in Sections 6 and 7. Section 9 contains test examples and presents some results. We end with concluding remarks and ideas for future work.

2. RELATED WORK

Foster and Metaxas [6] introduced 3D physically based animation of fluids to computer graphics area by solving the 3D Navier- Stokes equations using a relaxation technique with Marker-And-Cell method to create water animation. The method uses marker particles to track fluid position allowing fluids to change into any shapes. In a series of their research, they extended the method to control the behavior of fluids and interact with moving objects [7]. Stam [13] improved the method using semi-Lagrangian method for fluid convection and implicit integration. A major strength of his method is that the simulation is unconditionally stable for any size of time step. He also introduced much efficient Chorin pressure projection instead of relaxation method to enforce incompressibility of fluids.

Foster and Fedkiw re-visited the MAC method with semi-Lagrangian method. They used both particles and implicit surfaces to represent liquid surfaces [5]. Their method called *particle level set* can track smooth and temporally coherent liquid surfaces with better fluid volume preservations.

In CFD area, VOF method was first proposed by Hirt and Nichols [8]. In the method, a donor-acceptor procedure can completely conserve a fluid volume when the fluid passes through a donor cell to an acceptor cell. The VOF method was improved by Fluid-attenuated Inversion Recovery (FLAIR) and Cubic Interpolated Propagation (CIP), and other methods developed to estimate fluid flux in more accurate ways.

There are a few researches to handle interaction of two fluids, or fluid and solid objects in computer graphics. Tanaka et al. [14] used VOF and CIP techniques to create fluid animations. They also integrated rigid body simulation into their fluid simulator. Hong and Kim [9] used the VOF method to animate bubbles in a liquid. Later they extended the method and described in detail a technique for capturing the boundary conditions at contact discontinuities between multiple fluids [10]. Carlson et al. [2] achieved to model high viscous, almost solid materials like a wax. The method avoids stability problems, which arises at high viscosities, and it can simulate melting effects by changing the viscosities. They also proposed rigid-fluid method [3]. The technique treats rigid bodies, as they were made of a fluid. Velocities inside solids are changed so that rigid motion is enforced in the region.

3. OVERVIEW OF NAVIER-STOKES SIMULATION

Motion of an incompressible viscous fluid is described by Navier-Stokes equations:

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

$$\nabla \cdot \mathbf{u} = 0, \quad (2)$$

where $\mathbf{u} = (u, v, w)$ is velocity vector, ρ is constant density, ν is constant viscosity, p is pressure and $\mathbf{f} = (f, g, h)$ is vector representing external force.

Equation 1 describes conservation of momentum. Left-hand side of the equation is rate of change of velocity at given position. Its right-hand side is composed of four terms representing effects of advection, diffusion, pressure and gravity, respectively. Equation 2 corresponds to conservation of mass. Here we assume that a fluid's flux into a small region is equal to the flux out of the region. As a result, this equation enforces fluid to be incompressible.

There are several approaches to solve the equations. The methods can be classified into three types, Eulerian, Lagrangian, and particle methods. In the Eulerian method, the mesh remains fixed, and it is necessary to compute the flow through the mesh. In Lagrangian method, computational mesh is moving with fluid velocities. Using this method, it is easy to track surface of the fluid, but handling topological changes can be difficult when the surface become complex. Last method uses particles to simulate the flows.

Here we summarize Marker-and-Cell (MAC) based method to simulate incompressible fluids. MAC method divides the computational domain into rectangular cells. In our implementation, the cells are uniformly sized cubes, with sides of length Δx , and indexed by i, j and k in x, y and z directions respectively. Figure 1 depicts a cell (i, j, k) . The lower right back corner of the cell is placed at $(i\Delta x, j\Delta x, k\Delta x)$. Velocity components, u, v and w , are separately defined on each face of the cell. Other quantities, like pressure and density, are located at the cell's centers. If a value is needed at a point on which no value is defined, a linear-interpolated value is used. Each cell has an attribute which describes the cell's content. Common attributes for MAC based simulators are *Full*, *Empty*, *Surface* and *Obstacle*. *Full* means the cell is filled with fluid. An *empty* cell contains nothing, or is vacuum. A *surface* cell is a cell containing some fluid and faces an empty cell. An *obstacle* cell represents solid obstacle into which fluid cannot flow. In our implementation, there is no vacant space in the simulation, but a cell can contain different types of fluids. Therefore, there are only two cell types, *full* and *obstacle* in the proposed method, see bellow.

3.1 Simulation Steps

The simulation steps for the MAC method are summarized as:

- (1) Choose an appropriate time step size Δt . Refer to Sec. 4.4.
- (2) Move the VOF values according to the current velocity field. Refer to Sec. 7.
- (3) Set boundary velocities on faces of obstacle cells and similarly set Neumann boundary conditions for pressure on obstacle cells. Refer to Sec. 5.
- (4) Calculate intermediate velocities using the Navier-Stokes equations without taking account the pressure term. Refer to Sec. 4.3.

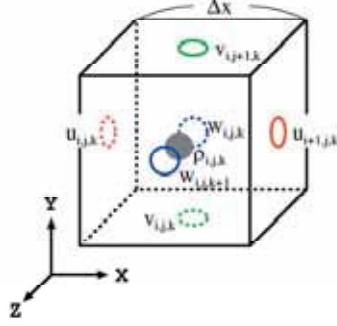


Fig. 1. MAC cell

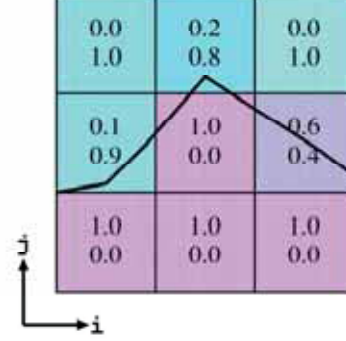


Fig. 2. VOF representation of two fluids.

- (5) Calculate pressure field from the intermediate velocity field, see Equation 7.
- (6) Make the intermediate velocity field mass-conserving to get correct velocities. Refer to Sec. 4.3.
- (7) To generate the animation frame construct fluid surface from the VOF values. Refer to Sec. 8.

4. PROPOSED METHOD FOR MULTIPLE FLUIDS

The main contribution towards the goal of this research is to simulate mixtures of fluids, for example a gas and liquid. We represent each fluid by a VOF function. Assuming the fluids are incompressible, then the spatial distributions of fluids has the following relation:

$$\langle F \rangle = \sum F_n = 1.0,$$

where F_n is the VOF function of n th fluid, while fluid can have different densities and viscosities, ρ_n and ν_n . Figure 2 shows a 2D example of our representation for two fluids. Each cell has two VOF values, the top row represents fraction of the first fluid and the bottom row corresponds to the second fluid. The interface between two fluids lies in cells which contain both fluids. We treat a flow of a mixture of fluids as a flow of a single fluid which have variable density and viscosity. Then, the motion of the flow is calculated using density and viscosity weighted by VOF values $\langle \rho \rangle = \sum (F_n \rho_n)$, $\langle \nu \rangle = \sum (F_n \nu_n)$.

4.1 Navier-Stokes Equations for Multiple Fluids

In Equation 2, the fluid density and viscosity were assumed to be constant all over the fluid. In this case, there are two or more fluids, and they are treated as a fluid with variable density and viscosity. The equation describing the multiple flow dynamics then will be

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla \cdot (\langle \nu \rangle \nabla \mathbf{u}) - \frac{1}{\langle \rho \rangle} \nabla p + \frac{\mathbf{f}}{\langle \rho \rangle} \quad (3)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (4)$$

Here the viscosity $\langle \nu \rangle$ is placed inside the derivative. The conservation of mass can be written in the other way as

$$\nabla \cdot (\langle \rho \rangle \mathbf{u}) = -\frac{\partial \langle \rho \rangle}{\partial t}.$$

We can interpret the second equation as that total of mass flow out of a small region is negative of mass change of the region.

4.2 Intermediate Velocities - Projection Method

Following the Chorin's projection method [4], we first derive equation for intermediate velocities that do not take into account the pressure term used in simulation step no. 4:

$$\tilde{\mathbf{u}} = \mathbf{u} + \Delta t \{ -(\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla \cdot (\langle \nu \rangle \nabla \mathbf{u}) + \frac{\mathbf{f}}{\langle \rho \rangle} \}. \quad (5)$$

Next step, is to make the velocity field mass-conserving by subtracting a missing term, $\frac{\Delta t}{\langle \rho \rangle} \nabla p$:

$$\nabla \cdot \mathbf{u}^{new} = \nabla \cdot \{ \tilde{\mathbf{u}} - \left(\frac{\Delta t}{\langle \rho \rangle} \nabla p \right) \}. \quad (6)$$

Here, we cannot place density $\langle \rho \rangle$ outside of the divergence operator, as is common for single fluid mass-conservation equation, because the density now changes spatially. But the equation can be rearranged as

$$\nabla \cdot \left(\frac{1}{\langle \rho \rangle} \nabla p \right) = \frac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}.$$

Solving the above equation for unknown pressure field p refers to simulation step no. 5. The current mass-conserving velocities are obtained by

$$\mathbf{u}^{new} = \tilde{\mathbf{u}} - \frac{\Delta t}{\langle \rho \rangle} \nabla p, \quad (7)$$

referring to simulation step no. 6.

4.3 Discretization

Intermediate Velocity. After discretization of gradient operator, Laplace operator, and divergence of the velocity in Equation 5 the x , component of equation becomes:

$$\begin{aligned} \tilde{u}_{i,j,k} = & u_{i,j,k} + \frac{\Delta t}{4\Delta x} \{ \\ & (u_{i-1,j,k} + u_{i,j,k})(u_{i-1,j,k} + u_{i,j,k}) - (u_{i,j,k} + u_{i+1,j,k})(u_{i,j,k} + u_{i+1,j,k}) + \\ & (u_{i,j-1,k} + u_{i,j,k})(v_{i-1,j,k} + v_{i,j,k}) - (u_{i,j,k} + u_{i,j+1,k})(v_{i-1,j+1,k} + v_{i,j+1,k}) + \\ & (u_{i,j,k-1} + u_{i,j,k})(w_{i-1,j,k} + w_{i,j,k}) - \\ & - (u_{i,j,k} + u_{i,j,k+1})(w_{i-1,j,k+1} + w_{i,j,k+1}) \} + \\ & + \frac{\Delta t}{\Delta x^2} \{ \nu_{i,j,k}(u_{i+1,j,k} - u_{i,j,k}) - \nu_{i-1,j,k}(u_{i,j,k} - u_{i-1,j,k}) + \\ & \nu_{i-1/2,j+1/2,k}(u_{i,j+1,k} - u_{i,j,k}) - \nu_{i-1/2,j-1/2,k}(u_{i,j,k} - u_{i,j-1,k}) + \\ & \nu_{i-1/2,j,k+1/2}(u_{i,j,k+1} - u_{i,j,k}) - \nu_{i-1/2,j,k-1/2}(u_{i,j,k} - u_{i,j,k-1}) \} + \end{aligned}$$

$$+ \frac{1}{\rho_{i-1/2,j,k}} f$$

Here, we used half index notation, $\nu_{i-1/2,j,k}$, that refers to viscosity at center of the left face of a cell (i, j, k) . Such values are calculated by linear-interpolation from neighboring cells.

Pressure Projection Matrix. For a cell (i, j, k) , Equation 6 is discretized as

$$\begin{aligned} & \frac{1}{\Delta x^2} \left\{ \frac{1}{\langle \rho \rangle_{i+\frac{1}{2},j,k}} p_{i+1,j,k} + \frac{1}{\langle \rho \rangle_{i-\frac{1}{2},j,k}} p_{i-1,j,k} + \right. \\ & \frac{1}{\langle \rho \rangle_{i,j+\frac{1}{2},k}} p_{i,j+1,k} + \frac{1}{\langle \rho \rangle_{i,j-\frac{1}{2},k}} p_{i,j-1,k} + \frac{1}{\langle \rho \rangle_{i,j,k+\frac{1}{2}}} p_{i,j,k+1} + \frac{1}{\langle \rho \rangle_{i,j,k-\frac{1}{2}}} p_{i,j,k-1} - \\ & \left. \left(\frac{1}{\langle \rho \rangle_{i+\frac{1}{2},j,k}} + \frac{1}{\langle \rho \rangle_{i-\frac{1}{2},j,k}} + \frac{1}{\langle \rho \rangle_{i,j+\frac{1}{2},k}} + \frac{1}{\langle \rho \rangle_{i,j-\frac{1}{2},k}} + \frac{1}{\langle \rho \rangle_{i,j,k+\frac{1}{2}}} + \frac{1}{\langle \rho \rangle_{i,j,k-\frac{1}{2}}} \right) p_{i,j} \right\} \\ & = \frac{1}{\Delta t \Delta x} (\tilde{u}_{i+1,j,k} - \tilde{u}_{i,j,k} + \tilde{v}_{i,j+1,k} - \tilde{v}_{i,j,k} + \tilde{w}_{i,j,k+1} - \tilde{w}_{i,j,k}). \end{aligned}$$

We can set up Equation 8 for all simulation cells. Note that boundary conditions have to be considered for some of the cells. Finally, the equations for all cells have to be combined into a global matrix equation, $\langle \mathbf{A} \rangle \mathbf{x} = \langle \mathbf{b} \rangle$. First, the unknown pressure values of all cells are collected into a vector \mathbf{x} , defined by $\mathbf{x} = (\dots, p_{i,j,k-1}, \dots, p_{i,j-1,k}, \dots, p_{i-1,j,k}, \dots, p_{i,j,k}, \dots, p_{i+1,j,k}, \dots, p_{i,j+1,k}, \dots, p_{i,j,k+1}, \dots)$. Next, we define a matrix that represents an operator:

$$\langle \mathbf{A} \rangle (\cdot) \equiv -\Delta x^2 \nabla \cdot \left(\frac{1}{\langle \rho \rangle} \nabla (\cdot) \right). \quad (9)$$

Therefore, each row of the matrix \mathbf{A} is formed as

$$\begin{aligned} & \left(\dots, -\frac{1}{\langle \rho \rangle_{i,j,k-\frac{1}{2}}}, \dots, -\frac{1}{\langle \rho \rangle_{i,j-\frac{1}{2},k}}, \dots, -\frac{1}{\langle \rho \rangle_{i-\frac{1}{2},j,k}}, \right. \\ & \dots, \frac{1}{\langle \rho \rangle_{i+\frac{1}{2},j,k}} + \frac{1}{\langle \rho \rangle_{i-\frac{1}{2},j,k}} + \frac{1}{\langle \rho \rangle_{i,j+\frac{1}{2},k}} + \frac{1}{\langle \rho \rangle_{i,j-\frac{1}{2},k}} + \frac{1}{\langle \rho \rangle_{i,j,k+\frac{1}{2}}} + \frac{1}{\langle \rho \rangle_{i,j,k-\frac{1}{2}}}, \\ & \left. \dots, -\frac{1}{\langle \rho \rangle_{i+\frac{1}{2},j,k}}, \dots, -\frac{1}{\langle \rho \rangle_{i,j+\frac{1}{2},k}}, \dots, -\frac{1}{\langle \rho \rangle_{i,j,k+\frac{1}{2}}}, \dots \right). \quad (10) \end{aligned}$$

Finally, a right-hand side vector is defined as $\langle \mathbf{b} \rangle \equiv -\frac{\Delta x^2}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}$. That is, each element of the vector $\langle \mathbf{b} \rangle$ is given by

$$-\frac{\Delta x}{\Delta t} (\tilde{u}_{i+1,j,k} - \tilde{u}_{i,j,k} + \tilde{v}_{i,j+1,k} - \tilde{v}_{i,j,k} + \tilde{w}_{i,j,k+1} - \tilde{w}_{i,j,k}). \quad (11)$$

The matrix $\langle \mathbf{A} \rangle$ is still sparse and symmetric. General sparse matrix solver can be used to solve the system for pressure unknowns.

4.4 Choosing Time Step

The Equation 8 is a forward Euler formulation, that requires small time step to keep the simulation stable. It is known that Δt must satisfy the following conditions [3]:

$$\Delta t < \frac{\Delta x}{\max(|u|, |v|, |w|)}, \text{ and } \Delta t < \frac{\Delta x^2}{6\nu}.$$

On every simulation step (refer to simulation step no. 1), we have to check if the current time step is small enough to satisfy above conditions. If not, the size is decreased to satisfy both conditions.

5. BOUNDARY CONDITIONS

At simulation step no. 3 we set the boundary velocities and Neumann boundary conditions for pressure. The cell can be occupied by a fluid in such case it is marked as *full* cell or it can be occupied by an obstacle, marked as *obstacle*. On the interface between two different types of cells we define the boundary conditions.

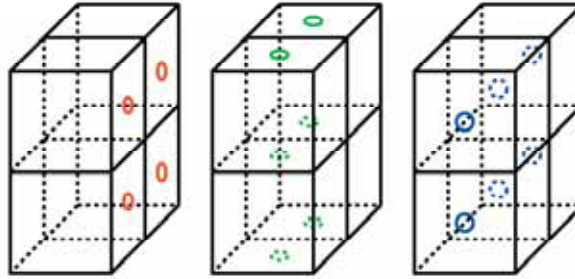


Fig. 3. Four *obstacle* cells and three velocity components, u , v and w .

5.1 Velocities on Obstacles/Full Interface

For faces between two *full* cells, their intermediate velocities are calculated with Equation 5. For other faces, the velocity is set according to specific conditions. We assume there is a large wall of *obstacle* cells with *fluid* cells to the right of the wall. Figure 3 illustrates the situation with four *obstacle* cells of the wall, and its three highlighted u , v and w components of the velocity \mathbf{u} .

The faces highlighted in the left image have u velocity components, on the interface between *obstacle* and *full* cells. Their velocities are set to zero to prevent fluid from passing through the faces. The middle and the right image show v and w components, respectively. This components are defined on the interface between two *obstacle* cells. We have three possibilities to set the velocity for these faces. The first, is simply to set the velocities to zero. The second, is to set the velocities to the averaged value of adjacent faces which have *full* cell on both of its sides. It is known as free-slip condition, allowing the fluid near the wall to slide freely. The third, is to set the velocities to the negative of that value, called no-slip condition. This condition makes tangential velocity at the surface of the wall to be zero. Given

that a cell's size is very small, the no-slip condition is the most accurate one. On the other hand, free-slip condition makes plausible fluid motion for coarse grids.

5.2 Neumann Boundary Condition for Pressure

To prevent fluid from entering into or flowing out of the *obstacle* cell, there should be no pressure changes between the *full* and the *obstacle* cells. In other words, gradient of the pressure on the face between the cells should be zero. This sort of condition is called Neumann boundary condition. For example, lower left back cell in Figure 4

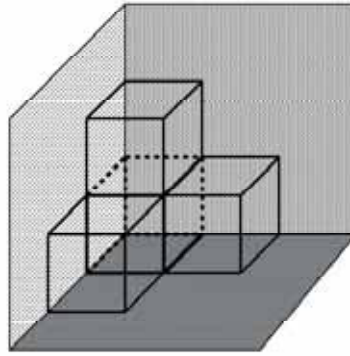


Fig. 4. Cells facing obstacles.

is faced with 3 *obstacle* cells in back. The Neumann boundary condition should be set on these faces. That is, pressure difference on a face interface in direction of the face normal should be set to zero. In this case a corresponding row of the left hand side of matrix Equation 8 can be written as

$$-p_{left} - p_{below} - p_{back} + 6p_{center} - p_{right} - p_{above} - p_{front} = (p_{left} - p_{center}) + (p_{below} - p_{center}) + (p_{back} - p_{center}) - 3p_{center} + p_{right} + p_{above} + p_{front}. \quad (12)$$

The first three terms on the right side of Equation 12 are differences between *full* and *obstacle* cells. Since, the boundary condition states that pressure differences should be zero, the corresponding row of the matrix looks like:

$$(\dots 0 \dots 0 \dots 0 \dots 3 \dots -1 \dots -1 \dots -1 \dots). \quad (13)$$

6. ESTIMATION OF SURFACE TENSION

Sometimes, the fluid interfaces behave like elastic films. The effect is caused by surface tension. Surface tension results in a surface force, which acts on the interfaces. Surface tension plays an important role in many natural phenomena. Accurate calculation of the effect needs well-defined fluid interfaces. Although our simulator doesn't consider actual shape of the interface, a model called continuum surface force (CSF) can be used to approximate the surface force without that information [1]. Method is based on volume force that converges to surface force as the grid size becomes smaller. We propose the extension of this idea to multiple fluids as well.

Volume force exerted by fluid n based on the VOF value, F_n , is written as

$$\mathbf{f}_n^{sv} = \sigma_n \kappa_n \nabla F_n,$$

where σ_n is surface tension coefficient, and κ_n is curvature of the fluid's surface. Normal and curvature of the surface are calculated from VOF values as

$$\begin{aligned} \mathbf{n}_n &= \nabla F_n, \\ \kappa_n &= -(\nabla \cdot \hat{\mathbf{n}}_n), \end{aligned}$$

where $\hat{\mathbf{n}}_n$ represents unit normal vector, $\mathbf{n}_n/|\mathbf{n}_n|$. Curvature κ_n is defined at center of a cell. However, as normal is a vector quantity, components of $\hat{\mathbf{n}}_n$ are separately defined on faces of a cell, and they must be interpolated to obtain values at other positions, see Figure 5. For example, x -component of a normal vector defined on center of right face of a cell (i, j, k) will be

$$n_{i+1/2,j,k}^{nx} = \frac{F_{n\ i+1,j,k} - F_{n\ i,j,k}}{dx},$$

and its y -component is obtained by interpolation:

$$n_{i+1/2,j,k}^{ny} = \frac{n_{i,j-1/2,k}^{ny} + n_{i+1,j-1/2,k}^{ny} + n_{i,j+1/2,k}^{ny} + n_{i+1,j+1/2,k}^{ny}}{4}.$$

Now, we are almost ready to calculate the surface force \mathbf{f}_n^{sv} on each cell face. To get the curvature defined on a face between two cells we need to interpolate two cell-centered curvatures. Finally, volume forces for all fluids are accumulated and added to the external forces in Equation 5:

$$\mathbf{f} = \sum \mathbf{f}_n^{sv} + \mathbf{f}^{ext}.$$

7. TRACKING FLUID SURFACE

As a fluid moves, its VOF value changes continuously along the fluid's velocity field as calculated at the simulation step no. 2. We can write down the equation that moves the VOF value F with fluid's velocities as

$$\frac{\partial F}{\partial t} = -(\mathbf{u} \cdot \nabla)F = -u \frac{\partial F}{\partial x} - v \frac{\partial F}{\partial y} - w \frac{\partial F}{\partial z}.$$

For an incompressible fluid the above equation can be combined with Equation 4 to yield the equation:

$$\frac{\partial F}{\partial t} = -\frac{\partial Fu}{\partial x} - \frac{\partial Fv}{\partial y} - \frac{\partial Fw}{\partial z}.$$

Above equation is in the divergence form that might be easier to understand if we think of a small cell with faces. It governs the change of F in a small cell as a total flux of F across the cell faces. Consequently, the total of the F within the grid is conserved. Straightforward finite difference discretization of the equation is not sufficient to track the fluid surface. Since each cell has only an averaged value of F , it can cause severe diffusion for the grid. The main reason of the diffusion is an assumption that the value of F is constant within a cell. If a cell contains fluid interface, there should be sharp changes of F in the cell, but the averaged value of

F smears the sharp profile of the F and introduces a blur like effect. Eventually, surfaces of fluids are quickly smeared.

For that reason, a special care must be taken to preserve sharpness of fluid surfaces. A flux approximation technique called donor-acceptor method was used in the case of a single fluid [8].

7.1 Proposed Donor-Acceptor Method

The essential idea is to use multiple values in upstream/downstream, leftstream/rightstream, fromstream/backstream cells to approximate crude shape of surface in the cells, and then compute the flux using that information. First we must find the *donor* and *acceptor* cells that are determined by normal velocity \mathbf{u} on cell faces. The amount of F_n fluxed across a cell face in a single time step then given by

$$\begin{aligned}\nabla F_n &= \min(F_n^{AD}|V| + CF_n, F_n^D), \\ CF_n &= \max((1 - F_n^{AD})|V| - (\sum_i F_i^D - F_n^D), 0), \\ V &= \frac{u\Delta t}{\Delta x},\end{aligned}$$

where F_n^D is F_n value of the donor cell, and the value of F_n^{AD} is determined by conditions described below. The above calculation is iterated over all faces between cells marked in MAC grid as *full*.

DEFINITION 1. *Isolated cell.* The donor cell is called isolated if there is fluid n that satisfies $F_n^D > F_n^{neighbor}$ for all neighboring cells.

The value of F_n^{AD} is initially set to F_n^D if the donor cell is isolated or otherwise F_n^A . Our approach prevents a fluid in a donor cell to flow into the next cell until the donor cell is filled with the fluid, unless the fluid is isolated in the cell. Figure 6

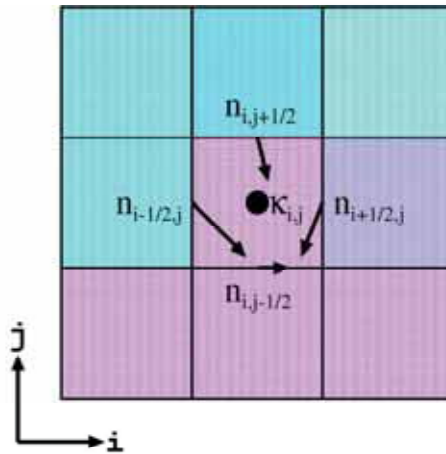


Fig. 5. Estimation of the normal and curvature at fluid surface.

visually shows some examples of ΔF_n and CF_n . Since velocity on the middle face is directed to the right cell, the left cell becomes *donor* and the right cell becomes *acceptor*. There are two fluids, namely *fluid1* and *fluid2*. Left cell contains more *fluid2* than *fluid1*, while right cell has more *fluid1* than the left cell. We assume that some of neighbor cells of the left cell are filled with *fluid2* therefore, the left cell is not isolated in this case, and we will use the values of acceptor cell for F_n^{AD} . Vertical dashed line represents $|V|$ and minimum in the above equation works for the *fluid1*. It limits the flow of *fluid1* to the amount of the fluid in the left cell. The flux of the *fluid1*, ΔF_1 , is highlighted in the right bottom part of the left cell. For the *fluid2* the above equation makes value CF_2 larger than 0. It accounts for the additional flux of the *fluid2* shown under the horizontal dashed line. It compensates the lack of the *fluid1*. At the end the fluids will flow through the face, and the hole process is iterated for all simulation cells.

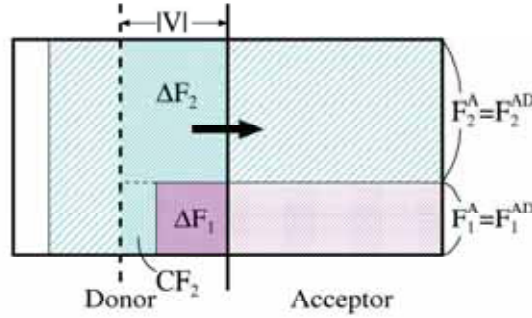


Fig. 6. Illustration of ΔF_n and CF_n .

8. VISUALIZATION

After we obtained VOF values fluid surfaces can be constructed from those values by employing the marching cubes algorithm [11]. The algorithm divides space into small logical cubes and creates triangles inside each cube according to values of its eight vertices. If a value at a vertex is larger than a user-specified value, that vertex is inside the surface. Vertices with values below the specified value are outside of the surface. Normal to the surface is obtainable from gradient of values at grid vertices. We can calculate gradients at the cube vertices by taking central differences. Normal vectors at the intersection points are estimated by linear interpolation and assigned to the triangle vertices.

This approach enables direct polygonization of VOF values. Since VOF values are placed at centers of simulation cells, the only thing to do is to create cubes by shifting our simulation grid. Setting values of cube vertices to the VOF values of the fluid makes polygonized surface of the fluid, see Figure 7.

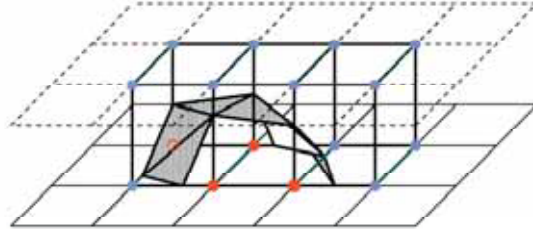


Fig. 7. Logical cubes and triangulated surfaces.

9. RESULTS

The input to the simulator is the simulation environment defined by the size of simulation grid, fluid's properties and placement of fluids and obstacles in grid cells. State of the simulation is calculated at every simulation time step and the polygonizer constructs the surfaces of fluids composed of triangles. We demonstrate the im-

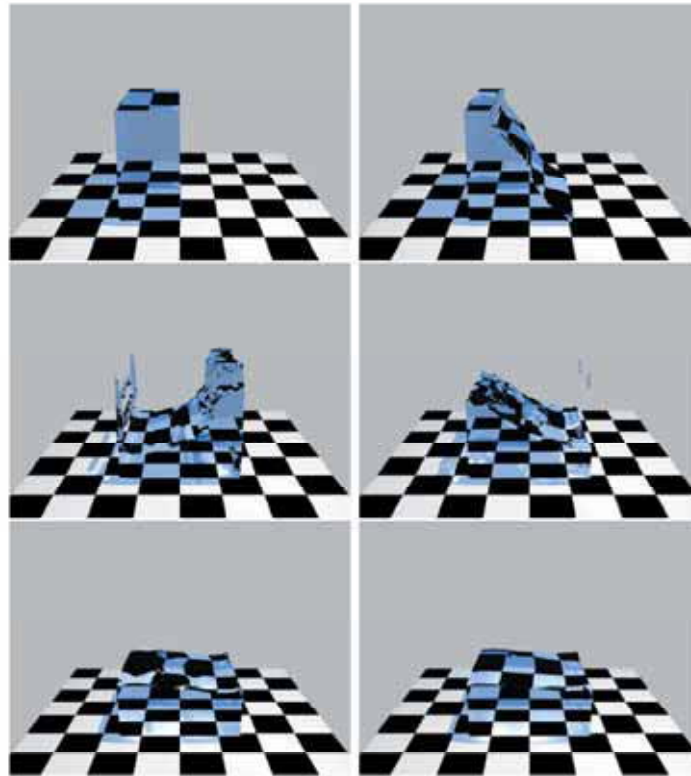


Fig. 8. Animation frames showing the interaction between two fluids. Initially the heavy fluid is on the left and light fluid is on the right side.

plementation of proposed algorithms in our custom system on an example including two fluids, heavy and light one. For heavy fluid we use the real physical parameters

of water at $20^{\circ}C$: density $\rho_{water} = 1000kg/m^3$, viscosity $\nu_{water} = 0,001002Nsm^{-2}$, and surface tension $0.0728N/m^2$. For light fluid we use the parameters of air at $20^{\circ}C$: density $\rho_{air} = 1.229kg/m^3$, viscosity $\nu_{air} = 1.71 \cdot 10^{-5}Nsm^{-2}$, and surface tension $0N/m^2$. There is a gravity vector acting downward, $(0.0, -9.8, 0.0)m/s^{-1}$. The size of the simulation grid cell is $0.5cm$.

The simulation results were finally rendered by POV-Ray (Persistence of Vision Raytracer) [12], which added the optical effects using ray-tracing algorithm. All of the simulation examples run on a PC with Athlon XP 1900+ and 1GB RAM.

In Figure 8 a column of a heavy fluid is placed at the left half of the simulation grid, the other half is filled with the light fluid. The simulation was performed on $24 \times 24 \times 24$ cells. The gravity causes pressure differences that moves down the heavy fluid and moves up the light fluid. The heavy fluid moves left and right several times. Gradually the movement slows down as the surface become horizontal.

In Figure 9 a block of the heavy fluid is placed at the top of simulation grid, and a layer of the fluid is also placed at the bottom. The other region is filled with the light fluid. Size of the simulation grid is $24 \times 24 \times 24$ cells. As the simulation progresses, the block begins to fall down, changing its shape to rounded one, then it drops to the bottom layer, making small splashes. Eventually, the fluid settles at the bottom of the grid. Figure 10 is a counterpart of the previous example. A

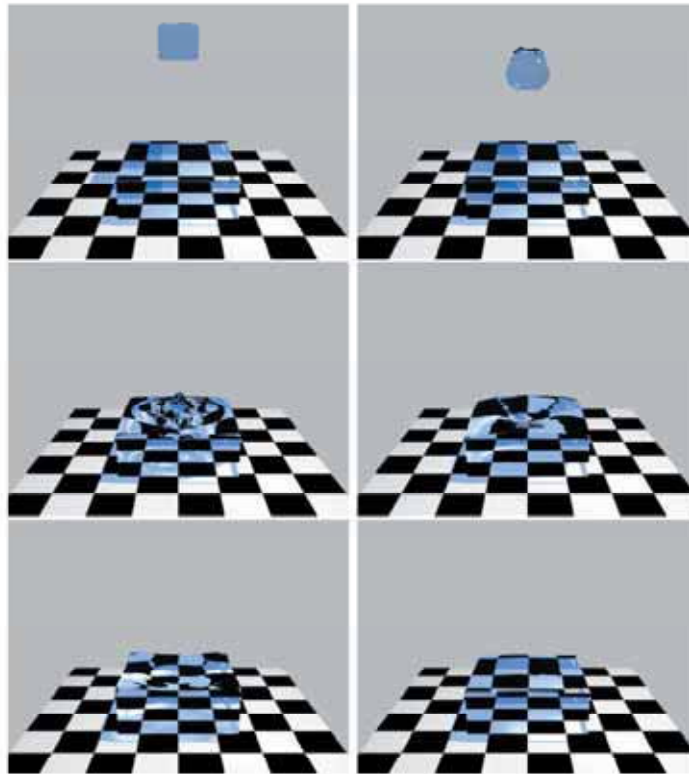


Fig. 9. Animation frames showing a mass of a heavy fluid falling in a light fluid.

block of the light fluid is at the bottom, and some of the upper cells are filled with the heavy fluid. The block behaves like a bubble in a liquid, changing its shape dynamically. First, it gets rounded, becomes ringed, and separates into small parts. When they reach to the surface, they disappear leaving small waves.

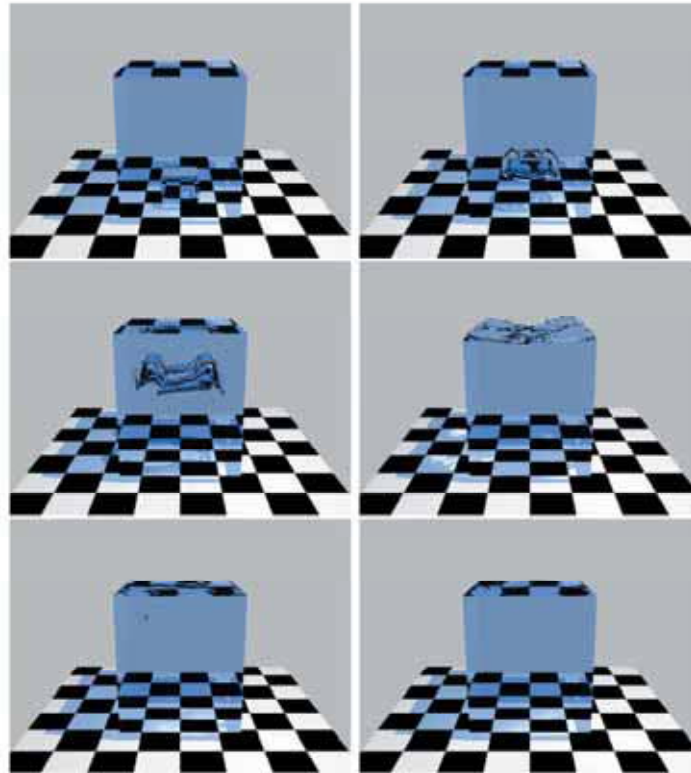


Fig. 10. Animation frames of a light fluid bubble rising in heavy fluid.

10. CONCLUSION

In this research, we proposed a physics based method for animating fluids. The major advantage of the method is that the method can deal with many fluids simultaneously. Whereas many simulation methods treat only single fluid in vacuum space, such treatment may not be appropriate in some cases. For instance, if we want to animate dropping water in oil, the mass and viscosity of the oil is not negligible. It was also difficult to make animation of air bubbles without considering air pressure. Our approach can handle those simulations because of novel estimation of surface tension for multiple fluids and extended VOF method.

Our simulation method is based on finite-difference equations of motion of fluids. Most of the process is straightforward. We adopted the proposed extension of VOF based method to track multiple fluids, because it prevents mass losses of the fluids completely. Since we don't have any extra space in our simulation region, change

Multi-phase Flow Dynamics with Volume Preservation

of fluid volume would be problematic. In addition, proposed methods are suitable for long time animations. Our simulation examples show many capabilities of our method. Animations of fluid interactions, like bubbles in liquid, are realistically simulated. The method is also applicable for fluid motions in which only single fluid plays a large part.

REFERENCES

- [1] BRACKBILL, J. U., KOTHE, D. B., AND ZEMACH, C. A continuum method for modeling surface tension. *J. Comput. Phys.* 100, 2 (1992), 335–354.
- [2] CARLSON, M., MUCHA, P. J., HORN, R. I. B. V., AND TURK, G. Melting and flowing. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, 2002), ACM, ACM Press / ACM SIGGRAPH, pp. 167–174.
- [3] CARLSON, M., MUCHA, P. J., AND TURK, G. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph.* 23, 3 (2004), 377–384.
- [4] CHORIN, A. J. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics* 2.
- [5] FOSTER, N., AND FEDKIW, R. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 23–30.
- [6] FOSTER, N., AND METAXAS, D. Realistic animation of liquids. *Graphical models and image processing: GMIP* 58, 5 (1996), 471–483.
- [7] FOSTER, N., AND METAXAS, D. Controlling fluid animation. In *CGI '97: Proceedings of the 1997 Conference on Computer Graphics International* (Washington, DC, USA, 1997), IEEE Computer Society, p. 178.
- [8] HIRT, C. W., AND NICHOLS, B. D. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics* 39 (1981), 201–225.
- [9] HONG, J.-M., AND KIM, C.-H. Animation of bubbles in liquid. *Computer Graphics Forum* 22, 3 (2003), 253–253.
- [10] HONG, J.-M., AND KIM, C.-H. Discontinuous fluids. *ACM Trans. Graph.* 24, 3 (2005), 915–920.
- [11] LORENSEN, W. E., AND CLINE, H. Marching cubes: a high resolution 3D surface construction algorithm. In *Proc. SIGGRAPH 1987* (1987), vol. 21, pp. 303–312.
- [12] PAGE, W. Pov-ray - the persistence of vision raytracer. In <http://www.povray.org/> (Jan. 2006).
- [13] STAM, J. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.
- [14] TANAKA, K., UEKI, H., AND KUNIMATSU, A. The cubic interpolated level set method for realistic fluid animation. In *SIGGRAPH '03: Proceedings of the SIGGRAPH 2003 conference on Sketches & applications* (New York, NY, USA, 2003), ACM Press, pp. 1–1.

Roman Ďurikovič,
Faculty of Natural Sciences,
The University of Saint Cyril and Metod,
Nám. J. Hedru 2, 917 01 Trnava, Slovak Republic
e-mail: roman.durikovic@fmph.uniba.sk

Numata Katsuhiko,
Software Department,
The University of Aizu, Japan

Received June 2006;