Toward Interactive-Rate Simulation of Fluids with Moving Obstacles Using Navier–Stokes Equations

JIM X. CHEN*† and NIELS DA VITORIA LOBO†

*Visual Systems Lab, Institute for Simulation and Training, Orlando, Florida 32826; †Computer Science Department, University of Central Florida, Orlando, Florida 32816

Received April 18, 1994; revised November 17, 1994; accepted December 21, 1994

We present a new method for physically based modeling and interactive-rate simulation of 3D fluids in computer graphics. By solving the 2D Navier–Stokes equations using a computational fluid dynamics method, we map the surface into 3D using the corresponding pressures in the fluid flow field. The method achieves realistic interactive-rate fluid simulation by solving the physical governing laws of fluids but avoiding the extensive 3D fluid dynamics computation. Unlike previous computer graphics fluid models, our approach can simulate many different fluid behaviors by changing the internal or external boundary conditions. It can model different kinds of fluids by varying the Reynolds number. It can also simulate objects moving or floating in fluids. In addition, we can visualize the animation of the fluid flow field, the streamline of a flow field, and the blending of fluids of different colors. Our model can serve as a testbed to simulate many other fluid phenomena which have never been successfully modeled previously in computer graphics.

1. INTRODUCTION

There are increasing needs for the computer graphics capability of simulating physically realistic complex fluid behaviors. Such behaviors include boats driving through water, liquids being poured from containers, liquids being stirred, ocean currents, the blending of differently colored fluids, the mixing of insolutes such as oil and water, and motions of liquid metals. Such capabilities would be useful in simulation and training systems, computer art, advertising, and networked virtual environments.

A physically based approach to achieve this capability would provide a common framework and uniform treatment of such simulations, in addition to the benefits that follow from making the choice of input parameters physically relevant, physically inspired, and physically intuitive. An additional need that persists is for the capability of providing at an interactive-rate at least some, if not all, of the effects that are visually relevant to the simulation. By interactive-rate, we mean that the physical process can be simulated at a rate which is close to frame rate on an easily available workstation, such as a Silicon Graphics Indigo. Success in this quest has eluded the computer graphics research community thus far.

The modeling and animation of fluids have captured the attention of many graphics researchers [1, 3–6, 9, 12–14]. However, general fluid models that are both physically realistic and computationally efficient for interactive-rate animation have not been developed. Fournier and Reeves [1] presented a simple model based on wave equations. Peachey [6] presented an alternative implementation of this approach. Both approaches avoided solving the fluid differential equations, and their approaches are limited to generating simple ocean waves. Miller and Pearce [5] used interacting particles for animating viscous fluids. Their approach represents particles throughout the volume of the fluid, incurring significant computational costs. Terzopoulos et al. [12] used molecular dynamics to model the process of solids transforming into liquids, and their approach too is computationally expensive for obtaining the behaviors we desire. Kass and Miller [4] presented a real-time method for animating fluid using a simplification of shallow water equations. This simplification amounts to solving the classical linear wave equation. Though they solved it in its differential form, all of its solutions are superpositions of waves with constant propagation speed [10] and do not allow for the modeling of complex phenomena that entail variations in the Reynolds number (which controls whether a fluid is laminar or turbulent) or which might include moving (self-propelled) objects. Goss [3] used a particle system to model ship wakes in real-time, but his approach was not based on physics. Weichert and Haumann [14] presented a model for inviscid irrotational flow which is only applicable to some situations of objects in a wind field. Stam and Fiume presented a method...
which models turbulent wind fields [9] but lacks physical basis.

To provide a physical foundation for general fluid animation, one must employ the Navier–Stokes equations, which are the embodiment of Newton's second law in fluids, and the governing equations of general fluid flow. Several researchers in computer graphics have acknowledged this [4, 9, 14]. However, none of the previous methods solved these equations due to the effort involved in deriving a solution method and the time needed to obtain a solution by a particular method on a commonly available computer.

While researchers in the discipline of computational fluid dynamics (CFD) extensively study computational models of physical fluids, their goals are constrained to obtain highly accurate and completely descriptive simulations of fluid behaviors. In contrast, the goals for many fluid applications in graphics simulation are producing realistic-looking fluid surfaces and fast calculations for interactive-rate animation. It suffices to say that CFD researchers have not undertaken studies to develop fluid modeling tools, based on the Navier–Stokes equations, that might sacrifice completeness of description in exchange for interactive-rate performance, satisfactory for computer graphics. Here, we introduce a computational fluid dynamics method using the Navier–Stokes equations. This method reduces the computation of these behaviors from the cube of the resolution to the square of the resolution, yielding results at an interactive-rate and achieving physically realistic looking 3D fluid simulation. However, we have to say that our method lacks the information of the depth of the fluid and therefore does not correspond to the exact CFD method. The solutions obtained from our method cannot be applied to satisfy accurate engineering needs.

Instead of calculating the fluid behaviors through a volume, that is, calculating the 3D Navier–Stokes equations, we compute the full incompressible 2D Navier–Stokes equations [7, 15]. Then we raise the surface of the fluid according to the corresponding pressures in the flow field, thus obtaining the third dimension of surface points. Using the pressures to simulate the height proportions of the fluid is justified by the fact that the higher pressures at the base of a fluid cause taller columns of surface above. This is due to the incompressibility of the liquid fluid. As shown in Fig. 1 above, when fluid from neighboring points flows into a point, the pressure as well as the height of the fluid surface at that point will rise, while when the fluid at one point flows to its neighboring points, the pressure as well as the height of the fluid surface at that point will drop. Empirically, the demonstrations of our interactive-rate simulations modeled in this paper are further justifications. This reduces the expense from the cube of the resolution to the square of the resolution without losing the three-dimensional effects and power of the Navier–Stokes equations. We can demonstrate at an interactive-rate on commonly available Silicon Graphics workstations, as powerful as or better than an Indigo, the simulation of different kinds of fluid flows and their vector fields. We can also simulate floating objects in fluids, moving (self-propelled) objects in fluids, streaklines of fluids, and blending of fluids of different colors at an interactive-rate by employing the fluid flow velocity and pressure field. Some of these phenomena have never been modeled before in computer graphics. The main idea behind our contribution is that pressures can be used to simulate the heights of the fluid surface points, and thus the computation is significantly reduced so that the general Navier–Stokes equations can be applied to model and simulate fluids at an interactive-rate for computer graphics. The applications are wide-ranging due to the generality of the Navier–Stokes equations.

The rest of the paper is organized as follows. In Section 2 we provide some necessary background on Navier–Stokes equations of fluid dynamics. In Section 3 we present details of our implementation, including the discretization method of computational fluid dynamics and the use of pressure to obtain the third dimension. In Section 4 we present applications and examples. This includes fixed and movable objects such as posts and boats, blending of fluids of different colors, and objects that float with the fluid current. In Section 5 we discuss stability issues. In Section 6 we discuss some of the limitations and problems of our models. Finally, in Section 7, we summarize our findings and contributions and describe several avenues of future research.

2. NAVIER–STOKES EQUATIONS

Navier–Stokes equations are derived from Newton’s second law [2, 8]. Here we present an outline of the derivation for 3D fluid flow.

There are nine stress components that act at a particular point in a fluid flow as in Fig. 2. The first subscript on a stress component denotes the face upon which the component acts, and the second subscript denotes the direc-
Many fluids exhibit a linear relationship between the stress components and the velocity gradients. Such fluids are called Newtonian fluids and include common fluids such as water, oil, and air. The stress–velocity–gradient relations, often referred to as the constitutive equations, are

\[
\begin{align*}
\tau_{xx} &= -p + 2\mu \frac{\partial u}{\partial x} - \frac{2}{3} \mu \nabla \cdot \mathbf{v} \\
\tau_{yy} &= -p + 2\mu \frac{\partial v}{\partial y} - \frac{2}{3} \mu \nabla \cdot \mathbf{v} \\
\tau_{zz} &= -p + 2\mu \frac{\partial w}{\partial z} - \frac{2}{3} \mu \nabla \cdot \mathbf{v} \\
\tau_{xy} &= \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\
\tau_{xz} &= \mu \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\
\tau_{yz} &= \mu \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right),
\end{align*}
\]

where \(p\) is the pressure, \(\mathbf{v}\) is the velocity vector \(ui + vj + wk\), and \(\mu\) is the viscosity of the fluid. Details of the development of the constitutive equations can be found in books on the subject of continuum mechanics [2, 8]. Using the stress–velocity–gradient relations, the above equations can be reduced to the following equations for an incompressible flow

\[
\begin{align*}
\rho \frac{Du}{Dt} &= -\frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) + \rho g_x \\
\rho \frac{Dv}{Dt} &= -\frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) + \rho g_y \\
\rho \frac{Dw}{Dt} &= -\frac{\partial p}{\partial z} + \mu \left( \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) + \rho g_z,
\end{align*}
\]

which together constitute the Navier–Stokes equations. The differential continuity equation for incompressible flow

\[
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \text{ i.e., } \nabla \cdot \mathbf{v} = 0,
\]

is derived from the law of conservation of mass. That is, the net flux of mass entering an infinitesimal control volume is equal to the rate of change of the mass of the element. It is used together with Navier–Stokes equations to decide the relationships between velocities and pressures in a fluid.

We can put the Navier–Stokes equations into the compact vector form
\[
\rho \frac{Dv}{Dt} = -\nabla p + \mu \nabla^2 v + \rho g, \quad (2)
\]

which is the governing equation of fluid dynamics and quite familiar to the fluid dynamicist. The solution for the velocities and pressures is complicated by the nonlinear terms \(DV/Dt\) on the left-hand side representing one of the major computational challenges in fluid mechanics.

The Navier–Stokes equations without external force can be written in dimensionless form

\[
\frac{\partial v}{\partial t} + \frac{u}{\partial x} + \frac{v}{\partial y} + \frac{w}{\partial z} + \nabla p = \frac{1}{Re} \nabla^2 v, \quad (3)
\]

where \(Re\) is the Reynolds number. It is a parameter that indicates the flow regime of the fluid and is defined as \(Re = \rho U L / \mu\), where \(L\) and \(v\) are a characteristic length and velocity, respectively. For example, given fluid flow inside a pipe, \(L\) can be the diameter of the pipe and \(v\) the velocity of the fluid flow inside the pipe. \(\rho, L,\) and \(v\) are considered constant in fluid applications. Therefore, if the Reynolds number is relatively small, the fluid is viscous and the flow regime is laminar; if it is large, the fluid tends to be inviscid and the flow regime is turbulent. Therefore the results of the Navier–Stokes equations with different Reynolds numbers correspond to the behaviors of different kinds of fluids.

### 3. IMPLEMENTATION

In our approach, we use the corresponding pressures in a 2D fluid flow field to simulate the third dimension (height) of the surface of fluid flow. Therefore, we only need to compute the 2D Navier–Stokes equations. We present the 2D discretization and computation method below.

#### 3.1. Discretization

There are several approaches in computational fluid dynamics to solve the Navier–Stokes equations [7]. Here we employ a finite-difference solution that uses penalization [7]. It introduces a penalty parameter \(\varepsilon\), to give

\[
\varepsilon p + \nabla \cdot v = 0, \quad \varepsilon > 0, \quad \varepsilon \rightarrow 0 \quad (4)
\]

instead of the divergence Eq. (1). It has been proven [11] that the joint solution of Eqs. (3) and (4) tends toward the solution of the Navier–Stokes equations (1) and (2), when \(\varepsilon \rightarrow 0\).

The spatial discretization of Eqs. (3) and (4) in 2D makes use of the staggered marker-and-cell mesh (Fig. 3) and we consider an explicit discretization [7],

\[
u_{i,j+1/2} = u_{i+1/2,j} + \left(-a_{n+1/2,j} - \Delta_p p_{n+1/2,j} + \frac{1}{Re} \nabla^2 u_{n+1/2,j} \right) \Delta t
\]

\[
u_{i,j+1/2} = v_{i+1/2,j} + \left(-b_{n+1/2,j} - \Delta_p p_{n+1/2,j} + \frac{1}{Re} \nabla^2 v_{n+1/2,j} \right) \Delta t
\]

\[
p_{i,j} = -\frac{\Delta_p u_{n+1} + \Delta_p v_{n+1}}{\Delta t}
\]

where \(i,j\) are flow fluid field mesh coordinates, \(n\) represents the current state, and \(n + 1\) represents the next state after a time-step of \(\Delta t\).

The difference operators \(\Delta_{x}, \Delta_{y}, \) and \(\nabla_k\) are defined by

\[
\Delta_{x} f_{i,m} = \frac{f_{i+1/2,m} - f_{i-1/2,m}}{\Delta x}
\]

\[
\Delta_{y} f_{i,m} = \frac{f_{i,m+1/2} - f_{i,m-1/2}}{\Delta y}
\]

\[
\nabla_{x} f_{i,m} = \Delta_{x} f_{i,m} + \Delta_{x} f_{i,m}
\]

\[
\Delta_{x} f_{i,m} = \frac{f_{i,m+1} - 2 f_{i,m} + f_{i,m-1}}{\Delta x^2}
\]

\[
\Delta_{y} f_{i,m} = \frac{f_{i,m+1} - 2 f_{i,m} + f_{i,m-1}}{\Delta y^2}
\]

The terms \(a_{n+1/2,j}\) and \(b_{n+1/2,j}\) are the approximations of \(\vec{u} \cdot \nabla \vec{u} + \vec{v} \cdot \nabla \vec{v} + \vec{w} \cdot \nabla \vec{w}\) and \(\vec{u} \cdot \nabla \vec{u} + \vec{v} \cdot \nabla \vec{v} + \vec{w} \cdot \nabla \vec{w}\) as defined in (3).

\[
a_{n+1/2,j} = u_{n+1/2,j}^{r+1} u_{n+1/2,j} + v_{n+1/2,j}^{r+1} u_{n+1/2,j} + w_{n+1/2,j}^{r+1} u_{n+1/2,j}
\]

\[
b_{n+1/2,j} = v_{n+1/2,j}^{r+1} v_{n+1/2,j} + u_{n+1/2,j}^{r+1} v_{n+1/2,j} + w_{n+1/2,j}^{r+1} v_{n+1/2,j}
\]

FIG. 3. The staggered marker-and-cell mesh.
where

\[ U_{i,j+1/2} = \frac{1}{4} (u_{i+1/2,j} + u_{i+1/2,j+1} + u_{i-1/2,j+1} + u_{i-1/2,j}) \]  
(15)

\[ V_{i+1/2,j} = \frac{1}{4} (v_{i+1/2,j+1} + v_{i+1/2,j} + v_{i-1/2,j} + v_{i+1/2,j-1}) \]  
(16)

\[ \Delta^x_f_{i,m} = \frac{1}{2\Delta x} (f_{i+1,m} - f_{i-1,m}) \]  
(17)

\[ \Delta^y_f_{i,m} = \frac{1}{2\Delta y} (f_{i,m+1} - f_{i,m-1}) \]  
(18)

The staggered maker-and-cell discretization results in all these approximations being of second-order accuracy. The iterative algorithm to compute the solution to the Navier–Stokes equations then is as follows. From the known current state of the velocity vectors and pressures of the fluid flow field \( \{u_{i,j+1/2}, v_{i,j+1/2}, \text{ and } p_{i,j}\} \), the next state \( \{u_{i+1/2,j}, v_{i+1/2,j}, \text{ and } p_{i+1,j}\} \) after \( \Delta t \) time is calculated by Eqs. (5)–(7).

3.2. Obtaining the Third Dimension

After we calculate each state of the velocity vectors and pressures of the 2D fluid flow field, we can render the current frame of the velocity field. For a given grid \((i,j)\) in the flow field, after computing

\[ u_{i,j} = \frac{u_{i+1/2,j} + u_{i-1/2,j}}{2} \]

\[ v_{i,j} = \frac{v_{i,j+1/2} + v_{i,j-1/2}}{2} \],

we can draw a velocity vector from \((i, j)\) to \((i + u_{i,j}, j + v_{i,j})\). By elevating the grid \((i,j)\) in the third dimension to some scaled value of \( p_{i,j} \), we can draw the fluid velocity field in 3D. Therefore as the interactive-rate calculations and rendering process, we can animate the velocity vectors of the fluid flow field. For example, given a channel flow with a boundary condition as specified at the beginning of Section 4, we obtain a frame of the animation of the velocity field as in Fig. 4, where yellow indicates levels above zero, white indicates those equal, and blue indicates those below.

By shading and lighting the surface of the flow field, we obtain a frame of the simulation of the channel flow as shown in Fig. 5.

Thus, for a fluid flow, we have obtained not only the simulation of the fluid surface, but also the velocity field on the surface, thus providing the velocities of all the visible points in the fluid. This information is very important when one wishes to simulate objects floating in the fluid with the fluid’s current. This also makes it possible to obtain the streamlines and streaklines of the fluid flow [8].

4. APPLICATION EXAMPLES

Here we present several application examples. In all the examples given here, the initial values of Reynolds number \( Re = 300 \), grid size in \( x \) and \( y \) direction is 1 (meter), time slice \( \Delta t = 0.001 \) (s), penalty parameter \( \varepsilon = 0.005 \), and the fluid field size is \( X \times Y \), where \( X = 140 \) and \( Y = 64 \).

4.1. Fixed Boundary Conditions

A flow is called internal flow if the flow field is inside a channel, pipe, or any other internal boundaries. A flow is termed external flow if the flow field is outside a post, bridge, or any other external boundaries. A fluid flow can be both internal and external if it has both internal and external boundaries. The flow field and fluid behavior change with the boundary conditions. With different boundary conditions, the fluid behaves quite differently. Therefore, the boundary conditions are extremely important in achieving the required fluid flow simulation. The boundary conditions can be fixed or moving, and these are inputs to the computation.

For example, consider the initial conditions of the fluid as follows:

\[ u_{i+1/2,j} = 5, \]

\[ v_{i,j+1/2} = 0, \]

and

\[ p_{i,j} = 0 \quad \text{for all } i = 0, 1, \ldots, X - 1 \text{ and } j = 0, 1, \ldots, Y - 1. \]

If we add a fixed external boundary condition presenting the edges and boundaries of a channel,

\[ u_{i+1/2,0} = 5, \quad u_{i+1/2,3/2} = 0, \]

\[ v_{i,3/2} = (v_{i,1/2} + v_{i,3/2})/2, \]

\[ u_{i+1/2,Y-1} = 5, \quad v_{i,Y-3/2} = 0, \]

\[ u_{1/2,j} = 5 + (Y - j)*j/Y, \]

\[ v_{0,j+1/2} = 0, \]

\[ u_{X-3/2,j} = (u_{X-1/2,j} + u_{X-3/2,j})/2, \]

and

\[ v_{X-1/2,j+1/2} = (v_{X-1,j+1/2} + v_{X-2,j+1/2})/2 \]

for all \( i = 0, 1, \ldots, X - 1 \text{ and } j = 0, 1, \ldots, Y - 1, \)

we get the simulation shown in Fig. 5. If we add to that simulation a fixed internal boundary condition which represents a post in the middle of the fluid field,

\[ u_{i+1/2,j} = 0 \text{ and } v_{i,j+1/2} = 0 \quad \text{for all } i \text{ and } j \text{ such that } (i - X/4)^2 + (j - Y/4)^2 < (\text{PostRad}/2)^2, \]
where \( \text{PostRad} \) is the radius of the post, we get the simulation shown in Fig. 6.

The boundary conditions can be modified separately to change the fluid behaviors. Therefore, depending on the applications, the boundary conditions can be fixed, movable, or modified at each step. This is described next.

### 4.2. Moving Boundary Conditions

With the exception of [3], the fluid behaviors due to moving (self-propelled) objects in the fluid have never been modeled before in computer graphics. In [3], the technique, though real-time, was not based on physics. However, with our fluid model, it appears very easy to model these behaviors in a physically based manner.

For example, by specifying a small internal boundary area inside the fluid flow field and moving the area and its boundary conditions in the fluid field, we can achieve the behaviors of a boat moving inside the fluid. Given an initial condition and fixed external boundary condition for a channel flow,

\[
\begin{align*}
    u_{i+1/2;j} &= 30 + (Y - j)*j/Y, \quad v_{i,j+1/2} = 0, \quad p_{i,j} = 0, \\
    u_{i,j} &= (u_{i+1/2;j} + u_{i-1/2;j})/2, \quad v_{0,j+1/2} = 0, \\
    u_{x-3/2;j} &= (u_{x-3/2;j} + u_{x-1/2;j})/2, \\
    v_{x-1/2;j} &= (v_{x-1/2;j} + v_{x-3/2;j})/2 \\
    \text{for all } i = 0, 1, ..., X - 1 \text{ and } j = 0, 1, ..., Y - 1,
\end{align*}
\]

and a movable internal boundary condition to represent a moving boat,

\[
\begin{align*}
    u_{i+1/2;j} &= \text{WakeSpeed}, \\
    u_{i,j+1/2} &= \text{WakeSpeed}, \quad j = \text{colorEle} \rightarrow y, \\
    u_{i-1/2;j} &= \text{WakeSpeed},
\end{align*}
\]

where \((i; j)\) is the current location of the moving boat, which in our example is repeatedly changing from \((0; Y/2); (1; Y/2), ..., (X - 1; Y/2)\), and WakeSpeed is the speed of the current boat, where in our example, WakeSpeed = 50 for the sailboat and WakeSpeed = 70 for the speed boat, we get the animation frame shown in Fig. 7. These conditions amount to setting the fluid velocities in the internal boundary area in correspondence with the movement of the boat.

If the internal boundary area moves slowly, the wake behind the area is very small and we have the effect of a sailboat in the fluid as shown in Fig. 8 (WakeSpeed = 50). Compare this with the result in Fig. 7, in which the area moves fast, causing the wake behind the area to be large and turbulent, simulating the effect of a speed boat (WakeSpeed = 70). We can put several internal movable boundary conditions together into the fluid to have the animation of different boats in the same fluid. For example, we can have two boats driving in a channel flow with brown balls drifting in the fluids of blended colors as shown in Fig. 9. The blending of fluids of different colors and the drifting of objects are discussed later.

One can also have movable external boundary conditions as the fluid interacts with the boundaries. It appears that we are able to achieve a variety of natural fluid behaviors simply by changing the boundary conditions and the Reynolds number.

### 4.3. Blending of Fluids of Different Colors

When fluids of different colors mix together, it will result in a new fluid of a mixed color. The blending process of differently colored fluids is very complex. Physical as well as chemical reactions occur during the blending process. Here we present a simple method of mixing fluids of different colors, producing a realistic look of color blending.

We consider fluids of different colors to consist of elements of different colors which can mix their colors together when they flow to the same grid position in the flow field. Each element updates its color and the color of the grid position to which the element travels by averaging the grid color and all elements at the same grid position. That is, at each iteration, we update at each grid position,

\[
\begin{align*}
    i &= \text{colorEle} \rightarrow x; \\
    j &= \text{colorEle} \rightarrow y; \\
    \text{colorEle} \rightarrow x &= \text{colorEle} \rightarrow x + \frac{u_{i+1/2;j} + u_{i-1/2;j}}{2} \Delta t \\
    \text{colorEle} \rightarrow y &= \text{colorEle} \rightarrow y + \frac{u_{i,j+1/2} + u_{i,j-1/2}}{2} \Delta t \\
    \text{color}(i; j) &= \frac{\text{color}(i; j) + \text{colorEle} \rightarrow \text{color}}{2}
\end{align*}
\]

where \((i; j)\) is the color of grid position \((i; j)\).
For example, we can have a channel flow with two pipes dumping fluids of different colors that blend and flow around an obstacle as shown in Fig. 10. It is possible to mix the colors according to more complex update rules.

4.4. Streaklines and Floating Objects

A streakline in fluid dynamics is defined as an instantaneous line whose points are occupied by all particles originating from some specified point in the flow field. Streaklines tell us where the particles are "right now." Here we present a method to view the streaklines in a flowing fluid.

We emit particles from several origins at regular intervals. Small intervals generate intense streaklines while large intervals generate sparse streaklines. Once a particle is generated, it flows with the fluid field. When the particle moves to a new position, its velocity is updated to the velocity at that position. Thus after generating a series of particles, it produces a streakline of the fluid flow.

For example, given the boundary conditions of a channel flow around a post, we have a frame of the animation of the streaklines as in Fig. 11; given the boundary conditions of a flow involving two dams, we have a frame of the animation of the streaklines and color blending as in Fig. 12. The vector field in Fig. 12 displays the natural turns and vortices generated automatically by the calculations of the Navier–Stokes equations. The only prior fluids work in computer graphics that employed vortices was by Wejchert and Haumann [14] in which the vortices are added by the user.

Using the same idea, we can animate objects drifting with the fluid by considering the objects as streakline particles. That is, the objects' positions can be updated as that of the streakline particles. For example, we can have a frame of the animation of brown balls drifting in the fluid as shown in Fig. 9. The interaction between objects and fluid is application dependent and could be included by modifying the velocity vectors and pressures around the objects.

In summary, the user of our system needs only to specify fixed or movable internal and external boundary conditions, fluid properties such as Reynolds number and colors, and streakline particles or floating objects.

5. STABILITY CONSIDERATIONS

The stability of the numerical computation depends on many factors. For a specific numerical discretization, the stability conditions can be mathematically analyzed. However, to play it safe, one can adjust some parameters to the safe zone or even dynamically update these parameters to achieve stability and other special effects. For example, fluid behavior varies with the Reynolds number. However, higher Reynolds numbers in the calculation may result in numerical divergence. We may first specify a very high Reynolds number to achieve some specific turbulent behavior but reduce the Reynolds number before the numerical calculation diverges. This way, we can use a simple numerical method to achieve behaviors not possible for a constant parameter. Even though this would not model an accurate physical fluid computation, it suffices for the purpose of realistic fluid animation for certain fluid phenomena.

Rather than present a lengthy mathematical analysis of stability conditions, we simply point out that the numerical calculations tend to be stable by choosing smaller $dt$ and Re and larger $dx$, $dy$, and $\epsilon$. The user can dynamically adjust these parameters to achieve the desired effects or find a set of values which is stable. The trade-off is that a smaller $dt$ results in a slower simulation; a smaller Re results in a quieter fluid behavior (tending to be laminar); a larger $dx$, $dy$, and $\epsilon$ will result in greater errors in the simulation. In all of our applications, we have a set of chosen convergence parameters, and the user is able to adjust all the parameters to change the simulation.

There are many other numerical discretization methods which allow larger Reynolds number and achieve better accuracy [7, 15], but the calculations of each iteration take longer because most of the methods require one to calculate the Poisson equations to obtain the distribution of the pressure. Thus, there is a trade-off between speed of iteration and stability/accuracy.

6. DISCUSSION

The new physically based method to model and animate fluid flows does have limitations. Thus far, experimental results with moving boundary conditions have been shown for objects that can be modeled by a single-layered cross-section. However, more complicated objects whose cross-sectional area varies with depth, for example, a stirring-spoon, cannot easily be modeled with our single-layered 2D approach. Such cases may require several layers that interact.

The external boundaries of the fluid body are fixed in our model. In order to simulate fluid phenomena such as water breaking a dam or water splashing when a vehicle drives over a water pond, we need to generate a new method to handle free boundaries.

While our model is stable for a period of simulation, and we can preserve the stability by adjusting the Reynolds number and other parameters, we are unable to provide a thorough analysis of the stability criteria yet. It is desirable to decide the exact parameters which is at the critical point where the stability is just preserved.
While avoiding extensive numerical calculations, we lost the depth information of the 3D fluid body. In contrast, using the full 3D Navier–Stokes equations to achieve the existing effects takes considerably longer and is difficult to implement. In order to include more accurate depth information while simultaneously avoiding extensive numerical calculations, further study of this approach is needed. The depth of the fluid is connected with the external boundaries of the fluid. This relationship is complicated by the fact that we need the environment information to calculate the depth and the boundary. The depth of the fluid will affect the area of the fluid to be simulated and the changing fluid boundaries.

7. CONCLUSION AND FUTURE WORK

We have presented a new physically based method to model and animate fluid flows at an interactive-rate. The correctness of this model is a consequence of employing the Navier–Stokes equations, which are the governing laws of fluid dynamics. Using the pressures to simulate the height proportions of a fluid is justified by the fact that higher pressures in a layer of fluid result in a higher free boundary surface. Thus, 3D effects are achieved while avoiding extensive numerical calculations. In contrast, using the full 3D Navier–Stokes equations to achieve the effects we have takes considerably longer.

We have successfully modeled several different fluid flows, objects floating in fluids, fluid interaction with objects (such as boats) inside fluids, and blending of fluids of different colors. Our examples are simple and limited; however our model can serve as a testbed to simulate many more fluid phenomena by changing the boundary conditions, adding interactions between fluids and objects, mixing fluids of different properties, etc. The applications are wide-ranging. In addition, we can employ different numerical discretization approaches in computational fluid dynamics to broaden the range of applications and the accuracy of the simulation.

The introduction of this interactive-rate, physically realistic, general-purpose fluid modeling technique into computer graphics will allow graphics animators to tackle many previously unsolved problems. These include having deformable objects move in fluids, rendering turbulence, changing the physical characteristics of fluids, modeling mixtures of fluids such as oil and water, correctly modeling compressible fluids such as gaseous flows, and modeling behaviors of fluids under certain conditions such as objects dropping into them.

APPENDIX: NOMENCLATURE

\( x, y, z \)

Coordinates along the three axes of a right-handed coordinate system. The \( x \)-\( y \) axes form the ground plane and position \( z \) is directed upward.

\( \tau_{xx}, \tau_{xy}, \tau_{xz}, \tau_{yx}, \tau_{yy}, \tau_{yz}, \tau_{zx}, \tau_{zy}, \tau_{zz} \)

Stress components: the first subscript denotes the face upon which it acts, and the second subscript denotes the direction in which it acts.

\[
\frac{D}{dt} = \frac{\partial}{\partial t} + \frac{\partial}{\partial x} u + \frac{\partial}{\partial y} v + \frac{\partial}{\partial z} w,
\]

Partial differential operators.

\( u, v, w \)

\( \frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt} \), respectively.

\( \rho \)

The density of the fluid.

\( g_x, g_y, g_z \)

The component of the gravity vector \( \mathbf{g} \) acting in the \( x \), \( y \), or \( z \) direction.

\( \mu \)

The viscosity of the fluid.

\( \rho \)

The pressure of a point in the fluid.

\( \nabla \)

Gradient vector operator \( \frac{\partial}{\partial x} \mathbf{i} + \frac{\partial}{\partial y} \mathbf{j} + \frac{\partial}{\partial z} \mathbf{k} \).

\( \mathbf{v} \)

Velocity vector \( u \mathbf{i} + v \mathbf{j} + w \mathbf{k} \).

\( \mathbf{g} \)

Gravity vector \( g_x \mathbf{i} + g_y \mathbf{j} + g_z \mathbf{k} \).

\( \nabla^2 \)

Laplacian operator \( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \).

\( \text{Re} \)

Reynolds number \( = \frac{\rho \nu L}{\mu} \), where \( \rho \), \( L \), and \( \nu \) are constants. Thus Reynolds number indicates the viscosity of the fluid.
Characteristic length and velocity, respectively. Given fluid flow inside a pipe, then $L$ can be the diameter of the pipe and $v$ the velocity of the fluid.

Discretization of velocities and pressure where $i, j$ are fluid flow field mesh coordinates, $n$ represents the current state, and $n + 1$ represents the next state after a time-step of $\Delta t$.

**ACKNOWLEDGMENTS**

The authors are very grateful to Tim Barto, Robert Buckley, Mike Goslin, Dr. Alain Kassab, Dr. Xin Lin, Glenn Martin, Dr. Michael Moshell, Mike Smith, Trenton Tuggle, Sean Waldron, and Greg Wiatroiski for their support.

**REFERENCES**