Controlling Fluid Flow Simulation

William Gates and Alain Fournier
University of British Columbia

Abstract

Simulating fluid dynamics can be a powerful approach to animating liquids and gases, but it is often difficult to "direct" the simulation to "perform" as desired. We introduce a simple yet powerful technique of controlling incompressible flow simulation for computer animation purposes that works for any simulation method using a projection scheme for numerically solving the Navier-Stokes equations. In our technique, an abstract vector field representing the desired influence over the simulated flow is modelled using simple primitives. This technique allows an arbitrarily degree of control over the simulated flow at every point while still conserving mass, momentum, and energy.

Keywords: animation, control, fluids, gases, liquids, Navier-Stokes

1 Introduction

Animating liquids and gases presents some of the most difficult challenges in computer graphics. Fluid motion can be extremely complex, yet it has a characteristic appearance that is easily recognized. With traditional animation methods such as key-framing, just positioning all the necessary control points of a model of, for example, a breaking wave can be tedious: getting the motion to appear realistic can be outright painstaking. Thus, researchers in computer graphics have developed methods specifically for animating fluids.

Recent research has demonstrated the effectiveness of adapting techniques from computational fluid dynamics (CFD) for the needs of computer animation [Fedkiw et al. 2001; Foster and Metaxas 1996; Foster and Fedkiw 2001; Stam 1999]. In this research, the Navier-Stokes equations are numerically solved in three dimensions over a regular grid to update fluid velocity values. This allows realistic motion to be generated from appropriate initial and boundary conditions. A key challenge of this approach is getting the fluid flow simulation to behave as desired.

This challenge has received relatively little attention. Foster and Metaxas [Foster and Metaxas 1997] presented several techniques for animator control over their liquid simulation method. In addition to setting initial and boundary conditions and modelling external body forces, these techniques include modifying pressure values to manipulate the simulated flow for effects such as the shock wave from an explosion. Foster and Fedkiw [Foster and Fedkiw 2001] allowed velocity values to be directly set anywhere in the flow by adapting their approach to modelling objects moving in fluids. They note their approach "[does not give] perfect direct control over the liquid motion."

We present a new approach that gives more direct control. Our approach is based on using a projection scheme for simulating incompressible flow. Before discussing our approach to controlling fluid flow simulation, we review this scheme for simulating incompressible flow.

2 Fluid Flow Simulation

We assume the fluids under consideration are incompressible, i.e., the density does not change with applied pressure. Note that does not necessarily mean that the fluid density is constant. This is an excellent approximation for both liquids and gases (as long the speed does not approach the speed of sound). Fluid flow is described by the Navier-Stokes equations. Under the assumption of incompressible flow, i.e., the fluid density does not change with applied pressure, the Navier-Stokes equations which describe fluid motion are:

$$\nabla \cdot \mathbf{v} = 0 \tag{1}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = \frac{1}{\rho}\left(\nabla \cdot 2\mu \mathsf{D} - \nabla p\right) + \mathbf{b} \tag{2}$$

where D is the rate of strain (deformation) tensor :

$$\mathsf{D} = \frac{1}{2}\left[\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right].$$

which gives the momentum equation in explicit terms of the fluid acceleration.

Our simulation algorithm is based on a projection scheme [Chorin 1967], a standard approach to numerically solving the incompressible Navier-Stokes equations in computational fluid dynamics that has recently become the favored approach for the computer animation of fluids [Fedkiw et al. 2001; Foster and Fedkiw 2001; Stam 1999]. A readable description of a basic implementation of a numerical simulation algorithm for the incompressible Navier-Stokes equations in two dimensions based on a projection scheme can be found in the book by Griebel et al. [Griebel et al. 1998]. The computation of the velocity for the next time step can be summarized as:

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \delta t\left(\mathbf{f} - \frac{1}{\rho}\nabla p^{n+1}\right)$$

where $\mathbf{f}$ is all terms other than the pressure term in the momentum equation (2). Note that this is implicit in pressure. Computing $\mathbf{v}^{n+1}$ involves two steps. First, an intermediate velocity $\mathbf{v}^*$ that ignores the pressure term is computed:

$$\mathbf{v}^* = \mathbf{v}^n + \delta t \mathbf{f} \tag{3}$$

and second, the projection phase where pressure values are used to correct the velocity:

$$\mathbf{v}^{n+1} = \mathbf{v}^* - \delta t \frac{1}{\rho}\nabla p^{n+1} \tag{4}$$

To compute the pressure at the new time step, p^{n+1}, such that the new velocity field conserves mass, the continuity equation for incompressible flow (1) is applied to equation (4):

$$\nabla \cdot \mathbf{v}^{n+1} = \nabla \cdot \left(\mathbf{v}^* - \delta t \frac{1}{\rho} \nabla p^{n+1} \right) = 0$$

which (assuming density is constant in the given domain) gives a Poisson equation for the pressure:

$$\nabla^2 p^{n+1} = \frac{\rho}{\delta t} (\nabla \cdot \mathbf{v}^*). \tag{5}$$

Solution of this equation for the pressure allows equation (4) to be used to correct the intermediate velocity.

3 Influencing the Simulation

A convenient physical metaphor for the will of the animator over simulated fluid flow is the specification of external body forces. The momentum equation (2) in our dynamic model includes a term for body forces, and thus we can model arbitrary changes in momentum. As mentioned previously, however, it is generally not clear what body forces are required to realized specific effects on the flow.

Rather than explicitly modelling body forces, our approach to controlling flow simulation is to model the effects of implied external body forces. In our simulation algorithm, all momentum changes except for those due to the pressure gradient are computed to give an intermediate velocity field $\mathbf{v}^*$ not constrained to conserve mass. At this point in the algorithm, we can simply set $\mathbf{v}^*$ to whatever values we want by assuming there are corresponding external forces that effect these changes in momentum. The next stage of the simulation algorithm corrects the intermediate velocity so that the resulting flow conserves mass (has zero divergence).

Directly manipulating intermediate velocity values can be a powerful approach to controlling our flow simulation that is far more intuitive than modelling external forces. However, simply setting an intermediate velocity value at some grid point to an arbitrary value ignores the current momentum of the fluid as well as the effects of gravity and viscosity. For some scenarios, such a jarring change may be exactly what is desired, but in general a more subtle approach is desired: instead of clobbering the momentum we want to accelerate or decelerate the flow towards the desired behaviour, i.e., we want the implied body forces to be smoothly varying in space and time.

Another issue with manipulating the intermediate velocity $\mathbf{v}^*$ is that the effects of the subsequent mass conservation step of the simulation algorithm on the velocity may be hard to predict. This step globally modifies the velocity field so that it has zero divergence. The more divergence we create anywhere in $\mathbf{v}^*$, the greater the global velocity correction will be, and in general the harder it becomes to predict its effects. Thus, we should try to not add any divergence if possible.

- $\mathbf{v}_{\mathscr{C}}(x,y,z,t)$ is the *control velocity* and
- $\alpha_{\mathscr{C}}(x,y,z,t) \in [0,1]$ is the degree of control over simulated fluid velocity.

The control velocity $\mathbf{v}_{\mathscr{C}}(x,y,z,t)$ is constrained to have zero divergence, i.e., to satisfy the continuity equation (1) for incompressible flow:

$$\nabla \cdot \mathbf{v}_{\mathscr{C}} = 0 \tag{6}$$

We modify the computed intermediate velocity $\mathbf{v}^*$ at time t_n as follows:

$$\mathbf{v}^*(\mathbf{x}) = (1 - \alpha(\mathbf{x}, t_n))\mathbf{v}^*(\mathbf{x}) + \alpha(\mathbf{x}, t_n)\mathbf{v}_c(\mathbf{x}, t_n) \tag{7}$$

so that the effect on the intermediate velocity varies depending on $\alpha_{\mathscr{C}}$, from unconstrained simulation ($\alpha_{\mathscr{C}} = 0$) to explicit control ($\alpha_{\mathscr{C}} = 1$). Thus, at any point where $\alpha_{\mathscr{C}} > 0$ and $\mathbf{v}^* \neq \mathbf{v}_{\mathscr{C}}$ there is an implied force driving the velocity towards $\mathbf{v}_{\mathscr{C}}$. The magnitude of this implied force is proportional to $\alpha_{\mathscr{C}}$ and $|\mathbf{v}^* - \mathbf{v}_{\mathscr{C}}|$. This simple technique can be quite powerful—if there is an easy way to model the influence field.

4 Streamtube Flow Primitives

To make our approach to controlling flow simulation useful, we need an effective method of interactively modelling the influence field $\mathscr{I}(x,y,z,t)$. The key challenge here is satisfying the mass-conservation constraint (6) on the control velocity $\mathbf{v}_{\mathscr{C}}$ for all points where $\alpha_{\mathscr{C}} > 0$. In previous work [Gates 1994], we addressed the challenge of modelling divergence-free flow fields (with infinite domains) using the superposition of divergence-free *flow primitives* $\mathbf{v}_p$, i.e.,

$$\nabla \cdot \mathbf{v}_p = 0,$$

and thus by the principle of superposition:

$$\nabla \cdot (\sum_p \mathbf{v}_p) = 0. \tag{8}$$

In other words, flow primitives are building blocks for constructing divergence-free flow fields.

We adapt this approach for modelling the influence field $\mathscr{I}(x,y,z,t) = \alpha_{\mathscr{C}}\mathbf{v}_{\mathscr{C}}$ by using primitives $\mathscr{I}_p(x,y,z,t)$ where

$$\mathscr{I}_p(x,y,z,t) = \alpha_p(x,y,z,t)\mathbf{v}_p(x,y,z,t)$$

and

$$\alpha_{\mathscr{C}}(x,y,z,t) = \min\{\sum_p \alpha_p(x,y,z,t), 1\} \tag{9}$$

$$\mathbf{v}_{\mathscr{C}}(x,y,z,t) = \sum_p \mathbf{v}_p(x,y,z,t) \tag{10}$$

and thus

$$\mathscr{I} = \min\{\sum_p \alpha_p, 1\} \sum_p \mathbf{v}_p$$

Note that the mass-conservation constraint (6) on the control velocity is satisfied by equation (8).

An influence field primitive $\mathscr{I}_p$ then is the product of a flow primitive $\mathbf{v}_p$ and a scalar field primitive α_p. The influence field primitives are blended in a way such that resulting velocity control magnitude $\alpha_{\mathscr{C}}$ is always in the range $[0,1]$ and the resulting control velocity $\mathbf{v}_{\mathscr{C}}$ has zero divergence. We use α_p like a filter for flow primitives $\mathbf{v}_p$ where α_p is greatest in the region where the strongest influence is desired goes to zero moving away from that region. This facilitates blending as well as spatially smoothing out the influence of the primitive on the simulation.

The flow primitives $\mathbf{v}_p$ in our previous work [Gates 1994] had infinite domains; none of these primitives are suitable for our finite spatial domain Ω with its boundary constraint:

$$\int_\Gamma \mathbf{v} \cdot \mathbf{n} d\Gamma = 0$$

Thus, we have developed a new class of flow primitives based on closed streamtubes. A streamtube is a surface made up of streamlines passing through a closed curve. There is no flow through a streamtube surface. Our primitives are based on streamtubes that are topologically equivalent to tori. Thus, there is no flow through the boundary of these toroid volumes; the flow circulates entirely within the streamtube. As long as these streamtube flow primitives

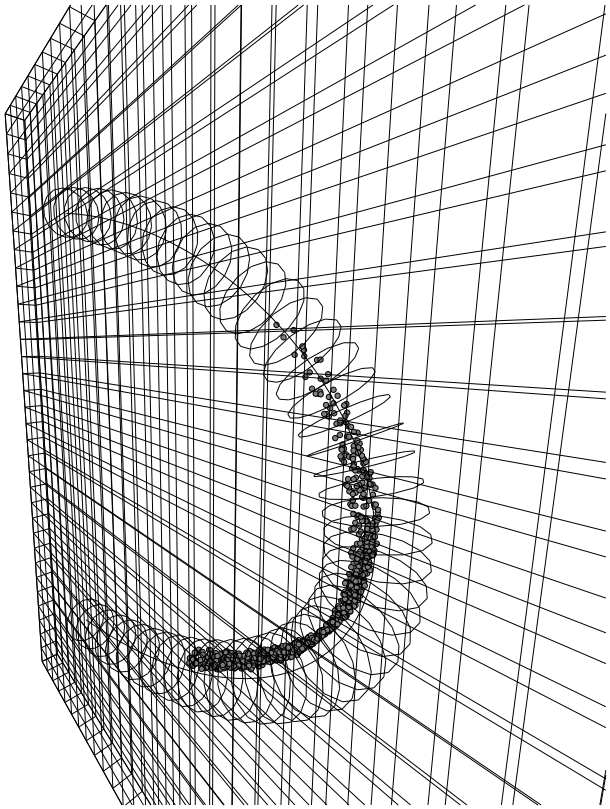

Figure 1: Particles following the desired flow specified by the streamtube primitives. Achieving this degree of control using forces would be quite difficult.

are completely within the fluid domain, they satisfy all the conditions of our dynamic problem.

Our streamtube flow primitives are naturally based on curves. We use B-splines, but any curve could be used. Figure 1 shows particles following the flow defined by a streamtube primitive. Streamtube primitives can have variable thickness, i.e., the radial dimension can be a function of the curve parameter. We approximate the incompressible flow in a streamtube by numerically integrating a circular disc source along a curve. We do not require an *exact* solution here as our simulation algorithm will correct for any divergence in this numerical approximation.

References

CHORIN, A. J. 1967. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics 2*, 12–26.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. *Proceedings of SIGGRAPH 2001* (August), (to appear).

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. *Proceedings of SIGGRAPH 2001* (August), (to appear).

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *Proceedings of Computer Graphics International '97*, 178–188.

GATES, W. F. 1994. *Interactive Flow Field Modeling for the Design and Control of Fluid Motion in Computer Animation*. Masters thesis, University of British Columbia.

GRIEBEL, M., DORNSEIFER, T., AND NEUNHOEFFER, T. 1998. *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. SIAM.

STAM, J. 1999. Stable fluids. *Proceedings of SIGGRAPH 99* (August), 121–128. ISBN 0-20148-560-5. Held in Los Angeles, California.