Computational Fluid Dynamics in a Traditional Animation Environment

Patrick Witting*

DreamWorks Feature Animation and Squeaky Cat

Abstract

This paper presents a system that uses computational fluid dynamics to produce smoke, water, and other effects for traditionally-animated films. The system was used in over twenty scenes in the animated feature film *The Prince of Egypt.* Animators use images and animation sequences to drive two-dimensional numerical simulations of the time-dependent compressible Navier-Stokes equations. For instance, images can be used to initialize temperature fields which cause dynamic buoyancy-driven vortices to evolve. In addition to being image-driven, the system is unique in allowing for compressibility of the fluid, and in its use of partial differential equations for texture mapping.

Keywords: animation, animation systems, applications, fluid simulations, natural phenomena, numerical analysis, physically based animation, physically based modeling, scientific visualization, texture mapping

1 Introduction

Computer graphics simulations of fluid behavior are in demand in filmmaking for depicting gases, liquids, smoke, dust, fire, and other natural phenomena. Methods for creating these simulations vary widely, depending on the requirements for realism, controllability, rendering style, and complexity. This paper describes a system, which utilizes numerical simulations of the full equations of fluid dynamics, that is being used at DreamWorks Feature Animation Studios in the production of traditionally-animated films. The system employs techniques from both the scientific and computer graphics communities in order to be both efficient and accessible to animators.

1.1 Motivation

Of the many ways of incorporating simulation into the creation of fluids animations, one end of the spectrum in a traditional animation environment is to use no simulation at all, and draw every frame of the animation. This approach gives a wide range of flexibility and control, but is a tedious process with realistic limits on the complexity that can be achieved. At the other extreme, there are many advantages to numerically solving the full equations of motion for fluids, usually referred to as the Navier-Stokes equations, to create animations of fluid behavior. With simple user set-ups, the physically accurate equations take over, generating lots of high quality animation, rich in complexity and guaranteed realistic motion.

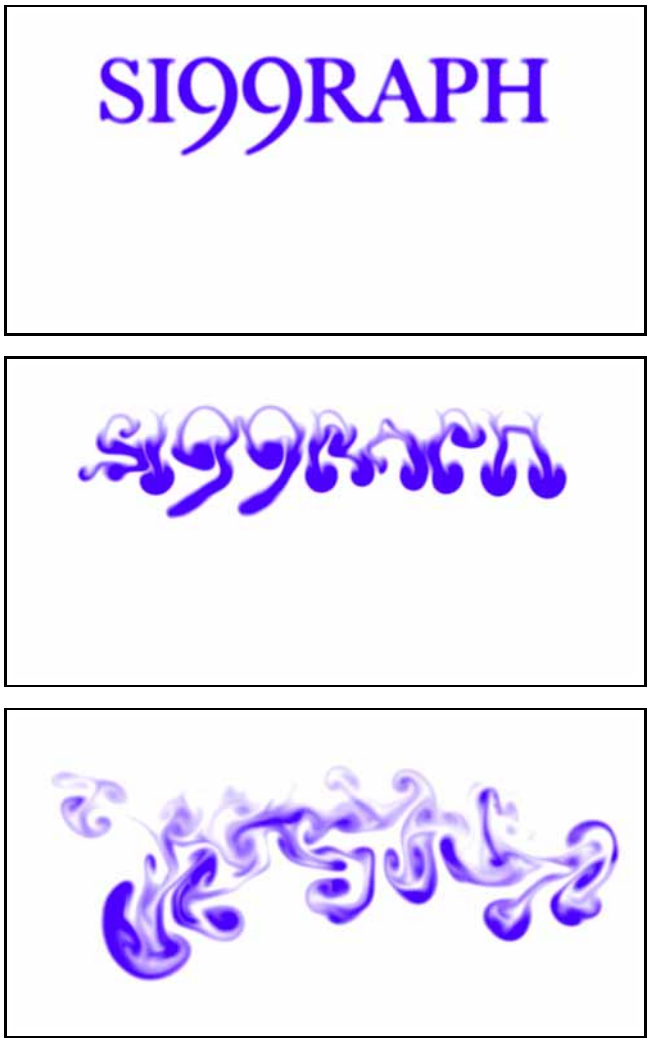

Figure 1: Temperature distribution - after 0, 100 and 400 time steps.

Figure 1 shows the results of a buoyancy-driven simulation created by simply interpreting the luminance of an image supplied by the user as the initial condition for the temperature field. The fluid inside the letters is colder and more dense than the surrounding fluid, causing it to sink. This is typical of the type of simulation that was used to generate smoke for *The Prince of Egypt*, where contours of temperature were rendered from a simulation driven by buoyant instabilities. This and other examples are discussed in more detail in section 5.

*3763 Lockerbie Lane Glendale, CA 91208
patrick@squeakycat.com

1.2 Modeling Fluid Behavior for the Sciences

Compared to computer graphics, the equations of fluid motion and solution methods for them have a long history. Equations expressing conservation of mass, momentum, and energy, often referred to as the Navier-Stokes equations, have been around since the early 1800's. Sir Horace Lamb's *Hydrodynamics* [11], from 1932, is still regarded as one of the best sources for fundamental theorems, equations, and solutions in fluid mechanics. The equations of motion cannot be solved analytically, except in simplified situations, and therefore need to be solved numerically. Numerical integration methods for systems of equations predate the modern computer as well, and John von Neumann envisioned using the computer to solve the equations of motion for weather prediction in the 1940's.

Today, the use of computers to solve the Navier-Stokes equations is widespread, with descriptions of particular models and their solutions filling the pages of journals such as *Journal of Fluid Mechanics* and *Journal of the Atmospheric Sciences*. Although computational fluid dynamics is a fairly mature subject, the emphasis so far has been on accurately simulating physical situations for scientific purposes, rather than creating images and animations as the end goal, which has different concerns and motivations. One simple example of this is the use of artificial compressibility, employed in the equation set presented in section 3, as a means of speeding up the calculations. For scientific work, the non-physical compressibility effects introduced need to be rigorously justified, whereas for the creation of imagery and animation, the guiding standard is how the images look.

When the emphasis is on the look of the final images, there are new sets of concerns about how to control and modify the simulation dynamics, and what and how to render. These concerns move us into the territory of computer graphics, with the highly practical production environment driving the process forward.

1.3 Modeling Fluid Behavior for Computer Graphics

Previous work in the graphics literature [2, 4, 6, 7, 9, 12, 13, 14, 16, 19] has modeled various aspects of fluid behavior with an emphasis on efficiency and controllability issues. Some of this work makes use of existing velocity fields or allows users to create their own in a variety of ways, rather than have a simulation determine the velocity field. The emphasis in this paper is on the use of the full Navier-Stokes equations to solve for the dynamic velocity and temperature fields numerically. Kass and Miller [9] solve the shallow water equations, which reduce the Navier-Stokes equations down to solving for an evolving height field for the surface of a shallow body of liquid. Yaeger, Upson, and Myers [19], used two-dimensional time-dependent vorticity equations to model the atmosphere of Jupiter. The strongest advocacy for use of the full Navier-Stokes equations so far in the graphics literature is from Foster and Metaxas [7], who solve the three-dimensional equations of motion to model smoke.

There may be no right or wrong answer as to what level of physical modeling is appropriate, in general, but there is usually a decision making process based on the imagery needed to guide this choice. Creative control and the level of realism desired are two of the main concerns. The decision making process is well illustrated in [19], where the end goal, creating animations of Jupiter's atmosphere for the film *2010*, guided aspects from the equations being solved to their final rendering method. This paper is of that same style, describing a system built at DreamWorks to support the use of fluid dynamics simulations in the creation of special effects for the animated feature film *The Prince of Egypt.*

1.4 Contributions of this Paper

Some of the unique features of the system described in this paper include: a compressible version of the equations of motion; the use of images and animations for controlling the dynamics; fast accurate texture mapping features; and finally, a complete production system.

The compressible formulation, unlike any in the graphics literature, allows for the modeling of compressible effects, such as shock waves, and also provides a mechanism for speeding up flow calculations by an order of magnitude or more. Another unique feature of the system is the use of images and animations as input devices, which allows animators to control initial conditions, source terms, and movable internal boundaries in an easy and flexible way. The inclusion of texture mapping differential equations, another new concept developed here, makes it possible to precalculate particle paths on a fixed grid which can be used in a straight-forward manner at render time.

The system also provides fast turn around time. Fourth order accuracy allows animators to use coarser grids, thus saving time. The use of two-dimensional simulations, the compressible formulation, and coarser grids, results in fast, useful simulations. Simulations performed on a 100 by 100 grid are detailed enough for film work and can be calculated at a rate of one frame per second. Additional production components make the overall process efficient for the animator.

2 Design Goals

Desirable characteristics of a useful production system which incorporates fluid dynamics simulations include the following:

- **Simulating a Variety of Flow Situations:** The equations being solved and the solution method should be capable of modeling a wide variety of flow situations, i.e. shear flow instabilities (Kelvin-Helmholtz), vortex motions, buoyant instabilities (Rayleigh-Taylor), Coriolis effects, gravity waves, compressible effects, etc. In addition, arbitrary forcing functions, or source terms, would be desirable to make many more situations possible, even those without any physical justification. Users should have easy access to setting up the various flow situations.

- **Control Throughout the Process:** The biggest difference between simulation systems for scientific purposes and simulation systems for production purposes is the level of control required in production work. Ideally, animators would control many aspects of the simulation dynamics and be able to incorporate the results into the final scene in a variety of ways.

- **Speed:** Speed is always a consideration in production work, because it usually translates into more iterations of the creative design cycle and a better final result.

- **Pipeline:** The overall process must make sense within the context of the production environment. The system should be able to make use of other scene elements, produce scene elements in the most convenient formats, and should be part of an efficient work flow.

- **A Variety of Rendering Styles:** The rendering style plays an important role in the overall process. A wide variety of rendering styles increases the expressive power of scene elements and their interpretation.

3 The Model

The equation set used was derived for a meteorology application, the study of clouds [10, 18]. The equation set presented in section 3.2 is a simplification of that system which meets the needs discussed in section 2 in a variety of ways discussed throughout.

3.1 Important Aspects of the Equation Set

Because this formulation of the equations of motion will be unfamiliar to many readers, this section has been included to characterize the equation set in a qualitative manner.

- **Conservation of Mass, Momentum, Energy:** The system of five equations and five unknowns is used to express conservation of mass, conservation of the 3 components of momentum, and conservation of energy. Along with the equation of state, which is an equation for one thermodynamic quantity as a function of two others, this forms a complete description of the fluid, i.e. the velocity and thermodynamic state of the fluid at any point. Given appropriate initial conditions and boundary conditions, the equations can be used to advance the solution forward in time. At the boundaries, a well-posed problem can be formed by specifying information for all the variables except the pressure, where the solution needs to be calculated [8].

- **Compressibility:** One of the most important aspects of the equation set is that there is no assumption of incompressibility. Not only does this mean that compressibility effects can be modeled, but the equations can be solved much faster. When an incompressible formulation is used, there is an elliptical partial differential equation involved, corresponding to an "infinite" speed of propagation of pressure waves. This typically translates into solving a large matrix equation, usually by iterative techniques, to ensure the pressure field is consistent with the velocity field. This is usually a time consuming part of the solution method and does not scale well as grid resolution is increased. Using the compressible formulation means that calculation times for each time step are essentially linear in the number of grid points.

- **Pressure Equation:** Because of the lengthy derivation, the pressure equation is presented as is. In summary, conservation of mass is expressed in the compressible equations by the mathematical statement that changes in density for a parcel of fluid are the result of divergence in the velocity field.

- **Buoyancy:** Some systems of equations make an assumption that the fluid has the same density everywhere, which simplifies the equation set at the expense of not modeling buoyancy effects. The equations being used here do not make that assumption and buoyancy effects dominate the dynamics in most of the examples presented.

- **Potential Temperature:** Potential temperature is used in meteorology as the appropriate measure of static stability, instead of density, temperature, or other variables which are not conserved in the atmosphere. For instance, a situation of having a colder fluid on top of a hotter fluid is not necessarily an unstable arrangement, due to the stratified hydrostatic pressure in the atmosphere. This concept is defined in most meteorology texts [3]. Throughout this paper "temperature" is often used in place of "potential temperature" for ease of reading.

- **Forcing:** The equations also allow for arbitrary forcing functions to each of the equations, except the pressure equation, corresponding to localized source terms for momentum and energy. These forcing functions can be analytical functions of the other variables, such as coriolis or buoyancy terms, or can come from other sources, such as images and animations.

- **Diffusion:** Each of the equations includes a diffusion term, which has the effect of damping out the high frequency waves. These terms have many interpretations, from molecular diffusion, to turbulence modeling, to numerical stability devices. Most ODE solvers (ordinary differential equation), including the fourth order Runge-Kutta scheme employed here, require some level of diffusion to avoid nonlinear instabilities.

- **Passive Scalar:** As discussed later, the system can also be augmented with additional equations, for things such as passive scalars which advect with the flow. Equations are derived for including texture mapping information, so that particle trajectories don't need to be computed via integration later.

3.2 Equation Set

The equations being solved are essentially those in [10]. The subgrid scale model is replaced by diffusion terms with constant diffusion coefficients, and the rain processes and coriolis terms are neglected. Also, the coefficient for the sound speed is multiplied by a constant, introducing artificial compressibility, so that the time step requirement is less severe. The primary variables being advanced forward in time are u, v, and w, which are the velocity components in the x, y, and z directions, respectively, the pressure perturbation variable, π, defined in equation 9, and the potential temperature, θ, defined in equation 8. The meteorology convention of using z as the *up* direction is used here.

$$\frac{Du}{Dt} = -c_p\overline{\theta}\frac{\partial \pi}{\partial x} + \nu\Delta u + f_u \tag{1}$$

$$\frac{Dv}{Dt} = -c_p\overline{\theta}\frac{\partial \pi}{\partial y} + \nu\Delta v + f_v \tag{2}$$

$$\frac{Dw}{Dt} = -c_p\overline{\theta}\frac{\partial \pi}{\partial z} + g\left(\frac{\theta-\overline{\theta}}{\overline{\theta}}\right) + \nu\Delta w + f_w \tag{3}$$

$$\frac{\partial \pi}{\partial t} = \frac{\overline{c}^2}{c_p\overline{\rho}\overline{\theta}^2}\left(\frac{\partial}{\partial x}(\overline{\rho}\overline{\theta}u) + \frac{\partial}{\partial y}(\overline{\rho}\overline{\theta}v) + \frac{\partial}{\partial z}(\overline{\rho}\overline{\theta}w)\right) \tag{4}$$

$$\frac{D\theta}{Dt} = \nu_\theta\Delta\theta + f_\theta \tag{5}$$

$\frac{D}{Dt}$ is the material derivative operator $\frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}$, and Δ is the Laplacian operator $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$. g is the acceleration of gravity, $\nu$and $\nu_\theta$are diffusion coefficients, c_p is the specific heat at constant pressure, c is the speed of sound, and f_u, f_v, f_w, and f_θ are forcing functions, or source terms for their respective equations. Base state variables, denoted by overbars, are time-invariant functions of z, the vertical coordinate.

The equation of state is the perfect gas law,

$$p = \rho RT, \tag{6}$$

where p is the pressure, ρ is the density of the fluid, R is the gas constant, and T is the temperature. Using p_0 as a reference pressure, a non-dimensional pressure, Π, is defined by

$$\Pi = \left(\frac{p}{p_0}\right)^{\frac{R}{c_p}}, \tag{7}$$

and a potential temperature, θ, by

$$\theta = T\left(\frac{p}{p_0}\right)^{\frac{-R}{c_p}}. \tag{8}$$

Defining a pressure perturbation variable π by

$$\Pi = \overline{\Pi} + \pi, \tag{9}$$

we assume the base state profiles obey the hydrostatic relationship

$$\frac{\partial \overline{\Pi}}{\partial z} = \frac{-g}{c_p \overline{\theta}}, \tag{10}$$

which reflects that the hydrostatic pressure of a parcel of air is caused by the weight of a column of air above it.

A two-dimensional version of the above equations can be derived by assuming that in one of the horizontal directions there is no flow and no change in any of the variables. Taking y to be the flowless direction, equation 2 is no longer needed, and simplifications are made to equation 4 and to the material derivative and Laplacian operators to account for zero derivatives in the y direction.

In addition to the basic equations of fluid motion, equations can be appended to the system which may or may not have feedback into the basic equations. Equation 11 is the prototypical passive scalar equation, which models an arbitrary scalar ψ being advected along with the fluid, and optionally diffusing through the non-negative diffusion coefficient ν_ψ.

$$\frac{D\psi}{Dt} = \nu_\psi \Delta \psi \tag{11}$$

Derivations of the equations of motion from first principles can be found in many textbooks for the interested reader [3, 11, 15, 17].

3.2.1 Texture Mapping Equations

Figure 2: Texture mapping after 0 and 400 time steps.

A convenient way to record the flow field history is through the dynamic evolution of texture map information. The idea is to initialize passive scalar variables with the original positions of the fluid parcels. These variables would obey equation 11, and let you know the original location of the parcel at any stage in the simulation, at the fixed grid locations. This *Eulerian* description is particularly useful in the rendering phase, since the texture mapping coordinate information is evenly spaced in the output image space. This technique is shown in figure 2 for the same simulation used to produce figure 1.

Suppose we are running a two-dimensional simulation on a rectangular domain of physical dimensions L_x by L_z. Define a horizontal texture map variable, ψ_x, with initial condition $\psi_x(x, z, 0) = x/L_x$ and a vertical texture map variable, ψ_z, with initial condition $\psi_z(x, z, 0) = z/L_z$. If both of these variables obey equation 11, then at a later time, t, $\psi_x(x, z, t)$ and $\psi_z(x, z, t)$ will contain the texture map coordinates at time $t = 0$ for the parcel at location x, z at time t, that is, they tell where the parcel of fluid "came from."

When implementing periodic boundary conditions, it is more desirable to keep track of displacement offsets from ψ_x and ψ_z because of the discontinuity of ψ_x and ψ_z as you cross periodic boundaries. Defining $p_x = \psi_x - x/L_x$ and $p_z = \psi_z - z/L_z$, we arrive at the following equations

$$\frac{Dp_x}{Dt} = -u/L_x + \nu_p \Delta p_x \tag{12}$$

$$\frac{Dp_z}{Dt} = -w/L_z + \nu_p \Delta p_z \tag{13}$$

3.3 Solution Method

The solution method for solving the system of equations is the fourth order Runge-Kutta scheme, using fourth order centered finite differencing for spatial derivatives on a regular grid with equal grid spacing. At boundary points and one point away, one-sided differencing is used. This solution method is briefly described below:

Ordinary differential equation solvers, such as the Runge-Kutta methods, solve the vector equation

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(\mathbf{y}), \\ \mathbf{y}(t_0) &= \mathbf{y}_0 \end{aligned} \tag{14}$$

The equation set 1 through 5 can be written in this form for the solution vector $\mathbf{y} = [u\ v\ w\ \pi\ \theta\ ]^T$ by moving the advective terms in the material derivatives over to the right hand side of their respective equations. The advective terms are those not involving partial derivatives with respect to time. The equations will now look like equation 14 where the prime in equation 14 denotes differentiation with respect to time. The right hand side of the equations become $\mathbf{f}(\mathbf{y})$.

The solution vector is initialized with values at the regularly spaced grid locations, then advanced forward in time according to the time integration scheme. This involves evaluating the function $\mathbf{f}(\mathbf{y})$ at each of the grid points, making use of the solution vector in a stencil of grid points surrounding the grid point being evaluated.

First and second derivative terms are replaced by their fourth order finite difference approximations, which can be found in [1]. The overall method is globally fourth order accurate in space and time, provided that the initial conditions, boundary conditions, and forcing functions are sufficiently smooth.

The fourth order accuracy is not required for production purposes, but the effort in achieving this added accuracy is not significant, and the increased accuracy allows for the use of coarser grids. For instance, comparing Runge-Kutta fourth order with Euler's method, four function evaluations per time step are required for Runge-Kutta compared with one for Euler, but this is almost offset by the time steps which can be 2.82 times larger, according to equation 15.

The time step limitation for stability for the advection problem, i.e. negligible diffusion, is

$$\Delta t < \frac{2\sqrt{2}}{c_* \sqrt{n} m} \Delta x, \tag{15}$$

where Δt is the time step, Δx is the grid spacing, n is the number of space dimensions, c_* is the speed of the fastest moving wave in

the system, and m is a factor that accounts for the spatial differencing method. For fourth order centered first derivatives, this factor turns out to be 1.372, compared with 1.0 for second order centered first derivatives.

Numerical methods for fluid dynamics can be found in a variety of places [5, 8], and an extensive book list and summary of available codes can be found at `http://chemengineer.miningco.com/msub74.htm` .

4 The Production System

This section describes the actual system built, which reflects the design goals of section 2, makes use of the model described in section 3, and also takes into account additional considerations specific to the traditional animation environment and the needs of *The Prince of Egypt.* In a traditional animation studio, most artwork and animation is two dimensional; the illusion of depth comes from the drawn or painted perspective, along with the camera moves and techniques available in the compositing software.

Many simulation and rendering techniques were used in the visual development stage of the film. Test animation resulted from three-dimensional simulations with temperature being visualized via volume rendering, two-dimensional simulations creating velocity fields used for line integral convolution of source imagery, as well as other techniques. By far, the biggest success was two-dimensional simulations of buoyant instabilities, where the temperature field was visualized as smoke. The plan was to use this technique to create "magical smoke" for the sequence *Playing with the Big Boys*, and the process was streamlined with this in mind.

4.1 Design Decisions

The components described in sections 4.2-4.4 were built to support two-dimensional simulations which use images and animations as input. The simulations output information at regular intervals which is later used in the compositor for rendering. Some of the advantages of these decisions are described below.

- **Control Through Layering:** Animators can build up libraries of elements produced by simulations, all of which can be easily repositioned, scaled, and even put into perspective within the compositor. The bottom of figure 3 shows two layers and how they were integrated into the final image above. The top element was scaled and had animating transforms to match to the motion of one of the magician's hands. The lower element had an animating transform to react to the sliding of one of his feet. Individual layers allow artists to make independent decisions for colors, opacities, rendering parameters, and transforms.

- **Speed:** Two-dimensional simulations allow for good interactivity in creating elements for later use. Some of the lower resolution final elements used in the film were created in under two minutes, and even the highest resolution simulations could be set up using the information gathered in simulations taking only a few minutes.

- **Deferred Rendering:**

  The texture mapping differential equations developed in section 3.2.1 and periodic output from the simulations allow for deferred rendering, using only a small fraction of the disk space required to save final images. Deferred rendering means that no rendering decisions need to be made at simulation time, and no simulation time is required at render time.

  This allows for a flexible system, where simulations can be run with a specific flow situation and final element in mind, such as rising smoke. Artists choose rendering parameters later, e.g. to alter final timing or to animate contour levels that make the smoke slowly dissipate. At render time, a library of potentially useful simulations is already built up, and rendering involves little more than appropriate resampling (see section 4.4).

Figure 3: Reactionary elements created by simple transformations.

4.2 Setting Up and Running Simulations

Although the code is capable of handling more general situations, such as analytically defined forcing functions, gravitational fields, and diffusion coefficients, only a subset of the functionality is available via the user interface. Images define the initial conditions for velocity and temperature. Scalar variables on the interface aid the software in interpreting the images, e.g. assigning values to the black and white limits of the images. Similarly, images and animations are employed to apply forcing terms to the momentum and energy equations. In addition, two images are used to optionally assign profiles to the horizontal velocity and the temperature as functions of z. This makes it easy to set up shear flows and stratified layers of density. Figure 4 shows the interface for starting simulations.

Using the simulation starting interface, animators can set other parameters such as the resolution, boundary condition types, output frequency, etc., and can monitor simulations in the viewer described below. If a simulation is evolving unsatisfactorily, an animator can quickly restart it using modified images or parameter settings. Before the simulation is run, the system performs a preprocessing step on the images, essentially resampling them and slightly smoothing them for the appropriate simulation resolution, and enforcing periodic conditions if needed. It also calculates the initial pressure field from the temperature field, ensuring that the hydrostatic relationship is satisfied for vertical columns of fluid. Figure 5 shows the input image summary before the preprocessing steps. Figures 4 through 6 are taken from example 2 discussed in section 5.

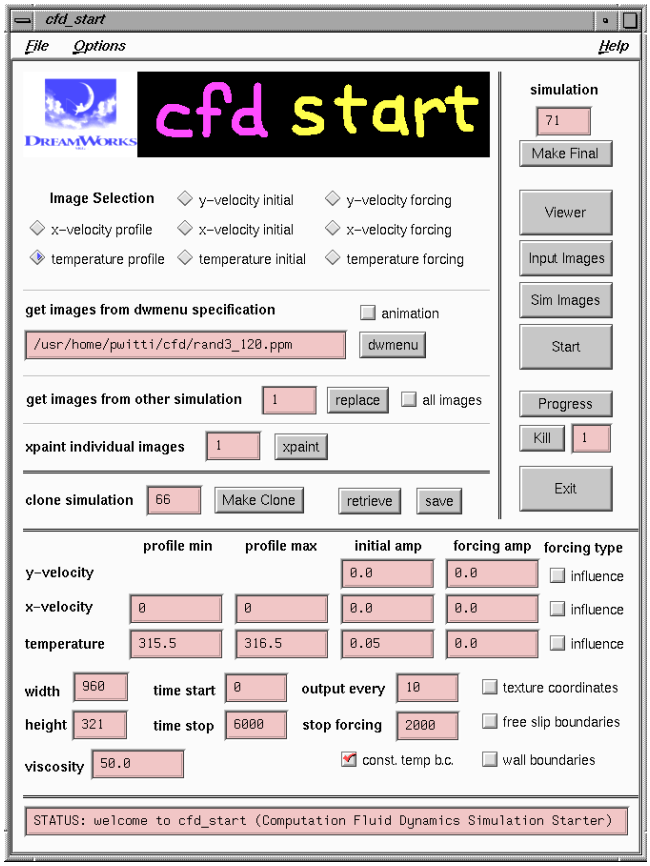

Figure 4: Simulation starter.

4.3 Previewing Simulations

As the simulations are running, or afterward, animators can preview and optionally render the results to disk via the interface shown in figure 6. This previewer is a simple mapping of the temperature values to the luminance of the black and white images. More rendering options described below are available in the compositor.

4.4 Rendering Simulations

The compositor is a graph-based system (DAG) where rendering operations are "nodes" in the graph.

- **Temperature Contours:** Two image generation nodes are provided in the compositor for rendering the temperature field, with temperature being mapped in a linear fashion to transparency. Values outside the linear range are clamped to "clear" or "solid." One node maps the results of simulations done on a rectangle with periodic sides onto a circle, as in middle of figure 8, and the other renders the rectangular temperature field.

  All of the parameters, such as the timing and threshold values, have animation curves. The rendering process involves reading the data from disk at the simulation resolution and performing resampling with a two-pass, one-dimensional cubic convolution kernel. It is important to do periodic extensions *before* resampling to avoid seams at the periodic boundaries, and to do thresholding *after* resampling to avoid stair-step effects for magnification near the threshold values.

- **Volume Rendering:** Volume rendering of the thresholded temperature field was supported for three-dimensional simulations in the visual development phase, but not in the production system.

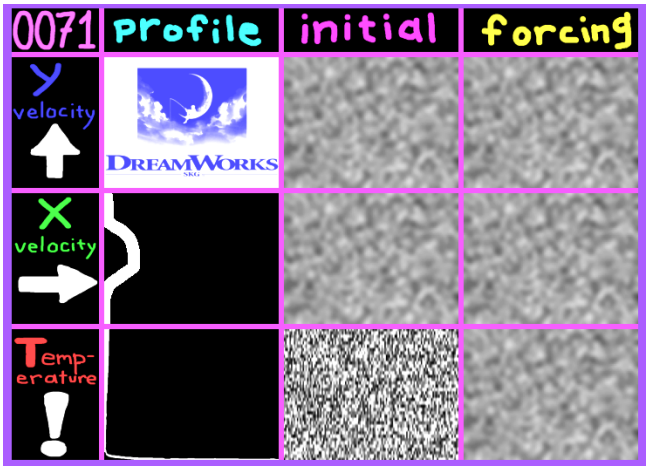

Figure 5: Input summary before preprocessing.

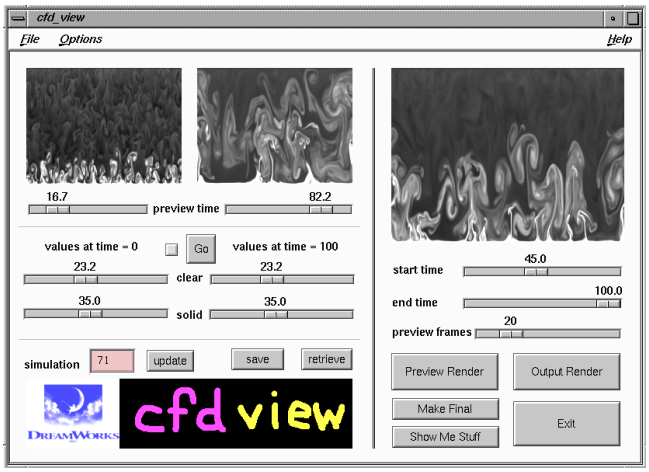

Figure 6: Simulation viewer.

- **Texture Mapping:** As described in section 3.2.1 and seen in figure 2, texture mapping is supported in the compositor. Inputs to this node are an image to be distorted, a simulation number, a reference time, and a current time. The image is distorted based on the flow field evolution between the reference time and the current time, using the texture mapping data for those two times.

- **Image Smearing:** Another rendering option supported in the compositor is the smearing of an image via line integral convolution using two-dimensional flow fields provided by the simulation. A single smearing uses one static flow field and a time range for the integration, provided by the user. Each output pixel receives its color from the colors visited along a flow integration path passing through the output pixel between the two specified times.

5 Examples

The example times quoted below are for a single processor SGI O2 with R10K floating point chip and processor chip. Calculation times

are given for simulation time steps. Simulation time steps and simulation time between final frames are roughly equal, for comparison purposes, using the following logic: According to equation 15, if a Mach number of 0.4 is used and the largest possible stable time step is used, then the fluid speed will travel the distance of about one grid point per time step. Unless the grid is extremely large, structures moving by one grid point corresponds to a reasonable speed for an animation. For render times, the quote is for producing 640 by 480 images.

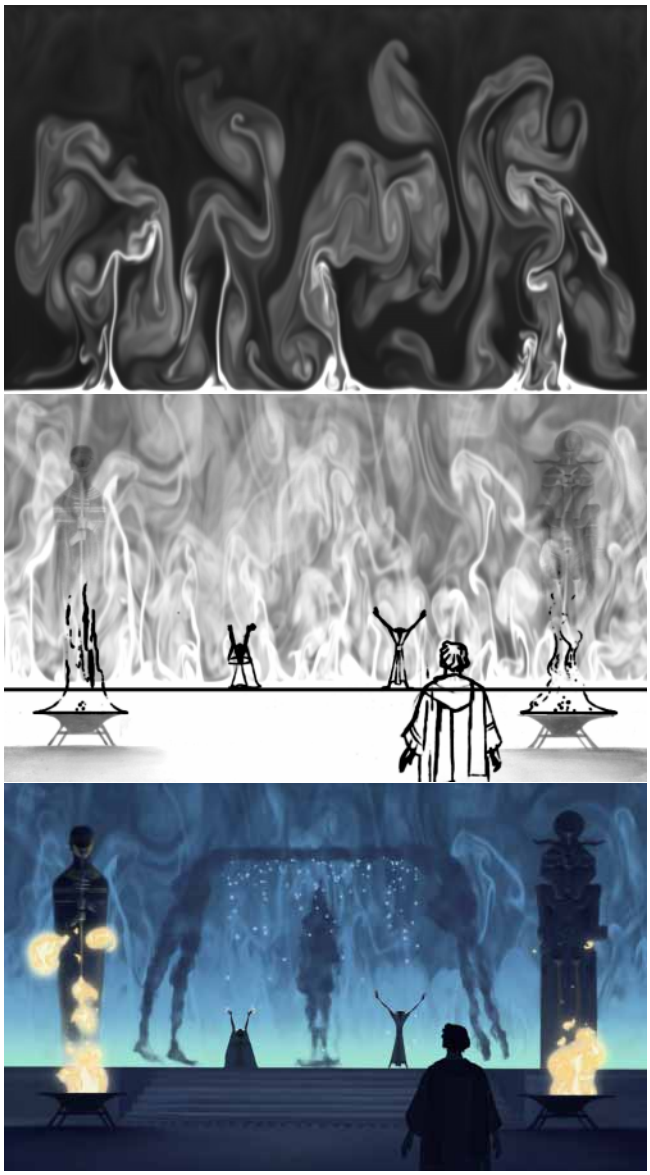

Figure 7: top) Temperature field. middle) Composition in scene. bottom) Final scene.

5.1 Example 1 - Image Used for Initial Temperature

In the first example, an image defines the initial temperature distribution and drives the dynamics of the simulation. The lettering in "SI99RAPH" is colder than the surrounding fluid, which causes it to sink. Conservation of mass dictates that there be areas of return flow as the cold fluid sinks, creating vortices. There is enough variation in the initial distribution such that the nonlinear equations result in pleasing graphic shapes and interesting dynamics.

This simulation was run on a 400 by 300 grid, with periodic sides. Figure 1 shows the temperature distribution at the start of the simulation and at two later times. Calculation time between time steps is 19.8 seconds, which include the texture mapping calculations. Render time for frames such as figure 1 is 3.16 seconds per frame.

As described in section 3.2.1, texture mapping information can be calculated along with the simulation to provide rendering information. Particle advection through the dynamically evolving velocity field is thus precalculated, eliminating the need to calculate particle trajectories at render time. Figure 2 shows the result of advecting the colors in an image along with the fluid for the simulation used to produce figure 1. An average render time for distortions such as those depicted in figure 2 is 9.8 seconds per frame.

5.2 Example 2 - Constant Heat Flux from Below

The second example simulates heat being introduced at the bottom of the domain creating "magical smoke" (see figure 7). The initial temperature distribution is a random noise pattern with an overall average temperature which is essentially constant except in a narrow layer near the bottom, where it is hotter. The images used for defining the initial conditions are shown in figure 5 and the other input values are the same as those shown in figure 4. The only images that are not scaled by zero, are the images used to define the unstable profile and the random perturbations in the initial temperature.

The simulation is performed on a 960 by 321 grid, with the rendering aspect ratio adjusted to make the shapes look taller and thinner than the actual simulation, which would otherwise promote rising plumes with essentially round circulation patterns. One time step calculation takes 36.3 seconds, and one rendered frame such as at the top of figure 7 takes 4.7 seconds to render.

5.3 Example 3 - Periodic Boundary Conditions in Action

In figure 8, "magical blood" is created by a simulation driven by a random forcing function in the temperature equation, defined by one of the input images. Using the circular rendering option and a periodic simulation domain creates a seamless texture mapping with the appearance of blood emanating from the center of the bowl. The final composite shows the circular shape being repositioned in perspective, registered to the bowl. Everything can be defined and rendered in one pass within the compositing package, including the animating perspective transformation. The simulation resolution is 150 by 151. Time step calculation time is 2.7 seconds per time step and rendering time is 1.57 seconds per frame.

6 Summary

This paper presents a complete production system which enables animators to access the beauty and realism embodied in the physically accurate equations of motion, the Navier-Stokes equations. With this system, animators can express themselves by controlling the simulation dynamics through a familiar user interface—the use of images and animations. Texture mapping features allow deferred rendering of flow distortions, with no need to recompute particle trajectories through a time-evolving velocity field. A compressible formulation and two-dimensional simulations allow for quick turnaround time in the creative cycle of creating/modifying simulations and applying the results within the compositor to the final scene.

While this production system emphasizes the needs of a traditional animation environment, many of the concepts apply outside

Figure 8: top) Temperature on periodic rectangular domain. middle) Circular domain remapping. bottom) Final scene.

this context as well. All of the equations, including the texture mapping equations, extend to three dimensions. One of the most useful ideas presented here for three-dimensional simulations is the implementation of an artificial speed of sound through the compressible formulation of the equations. Atmospheric researchers often use the compressible formulation because of its computational advantages over the incompressible formulation, even when using the actual speed of sound for pressure waves. For computer graphics purposes, an artificial speed of sound of an order of magnitude less than the actual one is often justified, and provides a mechanism for dramatic speed increases.

References

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover, 1974.

[2] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-Generated Watercolor. In *Computer Graphics*, pages 421–430. ACM SIGGRAPH, 1997.

[3] John A. Dutton. *The Ceaseless Wind*. Dover, 1986.

[4] David S. Ebert and Richard E. Parent. Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques. In *Computer Graphics*, volume 24(4), pages 357–366. ACM SIGGRAPH, 1990.

[5] C.A.J. Fletcher. *Computational Techniques for Fluid Dynamics*. Springer, 1990.

[6] Nick Foster and Dimitris Metaxas. Realistic Animation of Liquids. In *Graphical Models and Image Proc.*, volume 58(5), pages 471–483, 1996.

[7] Nick Foster and Dimitris Metaxas. Modeling the Motion of a Hot, Turbulent Gas. In *Computer Graphics*, pages 181–188. ACM SIGGRAPH, 1997.

[8] Bertil Gustafsson, Heinz-Otto Kreiss, and Joseph Oliger. *Time Dependent Problems and Difference Methods*. Wiley, 1995.

[9] Michael Kass and Gavin Miller. Rapid, Stable Fluid Dynamics for Computer Graphics. In *Computer Graphics*, volume 24(4), pages 49–57. ACM SIGGRAPH, 1990.

[10] Joseph B. Klemp and Robert B. Wilhelmson. The Simulation of Three-Dimensional Convective Storm Dynamics. *Journal of the Atmospheric Sciences*, 35:1070–1096, 1978.

[11] Sir Horace Lamb. *Hydrodynamics*. Dover, 1932.

[12] Karl Sims. Particle Animation and Rendering Using Data Parallel Computation. In *Computer Graphics*, volume 24(4), pages 405–413. ACM SIGGRAPH, 1990.

[13] Jos Stam and Eugene Fiume. Turbulent Wind Fields for Gaseous Phenomena. In *Computer Graphics*, pages 369–376. ACM SIGGRAPH, 1993.

[14] Jos Stam and Eugene Fiume. Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes. In *Computer Graphics*, pages 129–136. ACM SIGGRAPH, 1995.

[15] Philip A. Thompson. *Compressible-Fluid Dynamics*. Rensselaer Polytechnic Institute Press, 1988.

[16] Jakub Wejchert and David Haumann. Animation Aerodynamics. In *Computer Graphics*, volume 25(4), pages 19–22. ACM SIGGRAPH, 1991.

[17] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, Inc., 1991.

[18] Patrick Witting. *Numerical Investigation of Stratus Cloud Layer Breakup by Cloud Top Instabilities*. PhD thesis, Stanford University, 1995.

[19] Larry Yaeger, Craig Upson, and Robert Myers. Combining Physical and Visual Simulation - Creation of the Planet Jupiter for the Film “2010”. In *Computer Graphics*, volume 20(4), pages 85–93. ACM SIGGRAPH, 1986.