Modeling the Motion of a Hot, Turbulent Gas

Nick Foster and Dimitris Metaxas

Center for Human Modeling and Simulation
University of Pennsylvania, Philadelphia
{fostern | dnm}@graphics.cis.upenn.edu

Abstract

This paper describes a new animation technique for modeling the turbulent rotational motion that occurs when a hot gas interacts with solid objects and the surrounding medium. The method is especially useful for scenes involving swirling steam, rolling or billowing smoke, and gusting wind. It can also model gas motion due to fans and heat convection. The method combines specialized forms of the equations of motion of a hot gas with an efficient method for solving volumetric differential equations at low resolutions. Particular emphasis is given to issues of computational efficiency and ease-of-use of the method by an animator. We present the details of our model, together with examples illustrating its use.

Keywords: Animation, Convection, Gaseous Phenomena, Gas Simulations, Physics-Based Modeling, Steam, Smoke, Turbulent Flow.

1. Introduction

The turbulent motion of smoke and steam has always inspired interest amongst graphics researchers. The problem of modeling the complex inter-rotational behavior that arises as gases of different temperatures mix and interact with solid objects is still an open one. This behavior forms the part of so many everyday scenes (*e.g.*, steam rising from street gratings) that it remains an important topic in computer graphics.

There have been several previous approaches to modeling gas motion for computer graphics. Wejchert and Haumann [18] and Sims [13] modeled gases using the manual superposition of deterministic wind fields. This gives an animator control over the flow in an animation by placing vortices and flow field components by hand. More random motion, due to turbulence and diffusion, has proved amenable to spectral analysis. Shinya and Fournier [15], Stam and Fiume [16], and Sakas [12] define stochastic models of turbulent motion in Fourier space, and then transform them to give periodic, chaotic looking vector fields that can be used to convect gas particles or interact with simple objects.

These and similar approaches to modeling turbulent gases require that the animator has micro-control over the behavior of the gas. They characterize the visual behavior of gases without accurately modeling the physics-based components of gas flow. This leaves the animator with the sometimes difficult task of defining wind field parameters and small scale stochastic turbulence parameters wherever the visual characteristics of the flow vary significantly. For simple scenes and homogeneous effects this leads to good results which can be easily controlled. However, for scenes involving complex motion or a lot of interaction between a gas and other objects, it is almost impossible to manually create and control a natural looking animation. This is because the appearance of this kind of phenomena is very sensitive to the behavior of the gas as a volume. Rising steam, for example, is directed by the interaction and mixing between it and the surrounding air, as well as the convective flow field around static or moving objects. It would be prohibitively difficult to model these effects by hand even using existing methods for defining stochastic turbulence and laminar wind fields. The best way to achieve realism would be to model these effects in a physically accurate way, but the methods available to do so are inefficient, and tailored to computational fluid mechanics rather than computer graphics.

Another popular method has been to treat gases as collections of particles. Ebert, Carlson, and Parent [3], Reeves and Blau [11], and Stam and Fiume [17] reduced the complexity of the gas volume modeling problem in this way by using discrete particles to represent gaseous motion. Particle systems are generally efficient, but have two inherent drawbacks. First, a real gas is a continuous medium; selecting particular regions, and then estimating the interaction between them, can lead to unpredictable results for the animator. It is also unclear how interaction between volumes of gas is modeled using forces between particles. Often, the rotational component of such interaction still needs to be added manually. Second, the most visually interesting gaseous behavior is due to the fact that the gas being modeled is mixing with its surrounding medium. This medium has not been modeled in the particle system methods and so its effects can only be estimated. This may lead to visual simulations that have an unrealistic feel to them. Yaeger, Upson, and Myers [10] generated an excellent animation of the surface of the planet Jupiter by building a vorticity field from a particle-based motion system. The results were very realistic, but the method does not generalize to three dimensions and cannot account for flow around obstacles. In addition, a Cray X-MP was required to achieve reasonable computation times. A similar combination of vortex field and particle motion was used by Chiba *et. al.* [1] for their 2D simulations of flames and smoke. This technique does generalize to three dimensions and handles laminar gas flow around objects very nicely, but it isn't strictly physics-based. Again, this puts responsibility on the animator to achieve realism. These methods do show however, that the combination of visual simulation and physics-based simulation can lead to satisfying results for computer graphics.

In this paper we develop a new physics-based model specifically designed to realistically animate the complex rotational component of gaseous motion, effects due to regions of different temperature within a gas, and the interaction between gases and other objects. This work directly addresses the problem that no graphics models exist for the precise calculation of the turbulent, buoyant, or rotational motion that develops as a gas interacts with itself and solid objects. In the past, definition of this component of gas motion has been done via ad hoc methods or left to the skill of the animator. The paper's main contribution is a method for the efficient animation of both the turbulent and swirling behavior of a three-dimensional volume of hot gas in an arbitrary environment. The model we have developed accounts for convection, turbulence, vorticity and thermal buoyancy, and can also ac-

curately model gas flowing around complex objects. This gives rise to a number of realistic effects that could not be modeled previously, such as hot steam being vented into a boiler room or the rolling smoke cloud from an explosion. We show that not only is the proposed method accurate, it is also fast, straightforward, and can be used as a general graphics tool. Fast, because we use a simplified set of equations (compared to those used in the computational fluid dynamics literature) which are adequate for modeling the desired effects. Straightforward, because boundary conditions are set automatically and can be used to model different types of objects (rough or smooth for example). The model is mathematically nontrivial, but we will show that its solution proceeds in relatively simple computational steps.

2. Developing a gas model for computer graphics

Before trying to model a hot gas for computer graphics purposes, it is important to have some intuition for those factors that influence its motion. Consider as an example, an old fashioned steam engine venting a jet of hot gas from its boiler. A governing factor in the motion of the gas is the velocity it has when rushing into the surrounding air. As it mixes with the slower moving air, the steam experiences drag (shearing forces), and starts to rotate in some places. This rotation causes more mixing with the air, and results in the characteristic turbulent swirling that we see when gases mix. A second important factor that governs gas motion is temperature. As the steam is vented, it tends to rise. Hotter parts of the gas rise more quickly than regions which have mixed with the cooler air. As the gas rises, it causes internal drag, and more turbulent rotation is produced. This effect is known as thermal buoyancy. Turbulent motion is further exaggerated if the gas flows around solid objects. At first the gas flows smoothly along the surface, but it eventually becomes chaotic as it mixes with the still air behind the object. Finally, even when conditions are calm, diffusion due to molecular motion keeps the gas in constant motion.

In the next sections we derive a "customized" numerical model for animating visually accurate gaseous behavior based on the motion components described above. We call the model customized, because it incorporates only the physical elements of gaseous flow that correspond to interesting visual effects, not those elements necessary for more scientific accuracy. The model is built around a physics-based framework, and achieves speed without sacrificing realism as follows.

A volume of gas is represented as a combination of a scalar temperature field, a scalar pressure field, and a vector velocity field. The motion of the gas is then broken down into two components: 1) convection due to Newton's laws of motion, and 2), rotation and swirling due to drag and thermal buoyancy. The rotational, buoyant, and convective components of gaseous motion are modeled by coupling a reduced form of the Navier-Stokes equations with an equation for turbulent mixing due to temperature differences in a gas. This coupling provides realistic rotational and chaotic motion for a hot gaseous volume.

In general, solving a nonlinear system throughout a 3D volume is much too time consuming for animation because any algorithm that does so accurately has a complexity of $O(n^3)$ [4]. However, the authors have recently shown that for computer graphics, realistic looking results can be obtained in a reasonable amount of time if such a system is suitably approximated and solved at very low resolutions [5]. For a gas this is done in two stages. First, we solve equations corresponding to the two motion components in a voxel environment containing rectangular approximations to arbitrary static objects. This significantly reduces scene complexity, makes the application of boundary conditions trivial, and yet keeps the basic structure of the objects intact allowing for interaction between them and the gas. Second, the solution proceeds using a finite difference approximation scheme which preserves

the turbulent and rotational component of gaseous motion even at very low resolution, making the scheme efficient and suitable for use as a general graphics tool. So even though the method is still $O(n^3)$, we have reduced n significantly (40–60 in the examples given).

The result is a scheme that calculates the movement and mixing of a gas within interesting environments in a visually and physically accurate way. The output from the system is a pre-sampled, regular grid of time varying velocity or temperature values, which, when combined with massless particles, can be rendered in a number of ways using popular volume density rendering methods.

For the following discussion of the method, we take a Newtonian approach and treat finite regions in space as individual gaseous elements. An element can vary in temperature and pressure and allows gas to flow through it with arbitrary velocity, but its position remains fixed. We now present the model used to calculate the components of gaseous motion mentioned above.

2.1. Convection and Drag

The velocity of gas in an element is affected by a number of factors. First, it is pushed along, or convected, by its neighbors. Second, the gas is drawn into adjacent regions of greater velocity (or lower pressure). This is called vorticity, or drag. Third, the element is affected by forces such as gravity. In some extreme gaseous phenomena there may also be motion caused by shock and pressure waves that arise because gas can be locally compressed. If, however, the class of effects that we want to model is restricted to day-to-day sub-sonic effects such as smoke from fires, steam from steam engines, and so on, then the terms due to the compressibility of the gas will have only a minor effect on the overall motion. Therefore, we make a simplifying assumption that locally, the gas is incompressible. Furthermore, we assume that motion due to molecular diffusion is negligible relative to other effects. When these assumptions are applied to the Navier-Stokes equations, which fully describe the forces acting within a gas, a reduced form can be derived. For brevity the full equations are not reproduced here, but the reduced form, without compressive effects, or gravity forces is

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \nabla \cdot (\nabla \mathbf{u}) - (\mathbf{u} \cdot \nabla) \mathbf{u} - \nabla p, \qquad (1)$$

where ∇ is the gradient operator, $\mathbf{u}$ is the velocity of the gas, $\cdot$ is the inner product operator, and p is the pressure of the gas. This equation models how the velocity of a gas changes over time depending on convection $((\mathbf{u} \cdot \nabla) \mathbf{u})$, its pressure gradient (∇p), and drag $(\nu \nabla \cdot (\nabla \mathbf{u}))$. It is generally combined with the continuity equation which models mass conservation and which is discussed later in this paper. The ν coefficient is the kinematic viscosity. Intuitively, small ν models a less viscous gas in which rotational motion is more easily induced. Equation (1) models the convective and rotational velocity in our customized gas.

2.2. Thermal Buoyancy

Forces due to thermal buoyancy also induce motion in a gas. If a hot gaseous element is surrounded by cooler elements, the gas will rise (or move against gravity in cases of interest to us). We model this effect by defining a buoyant force on a gaseous element, as

$$\mathbf{F}_{bv} = -\beta \mathbf{g}_v (T_0 - T_k), \qquad (2)$$

where $\mathbf{g}_v$ is gravity in the vertical direction, β is the coefficient of thermal expansion, T_0 is an initial reference temperature (a balmy $28^o C$ for the examples in this paper), and T_k is the average temperature on the boundary between a gaseous cell and the one above it. Although simple, this equation seems to work very well.

In order to use (2) to calculate buoyant forces, the evolution of temperature within the gas must also be modeled. Adjacent elements exchange energy by straight convection (hot gas flowing from one element to another) and also by small scale turbulent mixing through molecular collisions with adjacent elements. Thus, the change in temperature of a gas over time can be characterized as a combination of the convection and diffusion of heat from adjacent regions. The differential equation that governs this process is [14]

$$\frac{\partial T}{\partial t} = \lambda \nabla \cdot (\nabla T) - \nabla \cdot T\mathbf{u}, \tag{3}$$

where $\mathbf{u}$ is the velocity of the gas, T is its temperature, and λ can be chosen to represent both turbulent and molecular diffusion processes. The structural similarity between (1) and (3) should be apparent. The second term on the right describes how temperature at a point changes due to convection, whereas the first term on the right takes into account changes in temperature due to diffusion and turbulent mixing. By solving (3) for a volume of hot gas, it is possible to calculate the force on a gaseous element due to thermal buoyancy using (2). This force affects velocity and so can be added as a new term to (1) giving

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \nabla \cdot (\nabla \mathbf{u}) - (\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p + \mathbf{F}_{bv}. \tag{4}$$

Equations (3) and (4) together provide us with a model for the rotational and turbulent motion that makes the mixing of hot and cold regions of a gas so interesting to watch.

3. Building a Useful Animation Tool from the Model

To obtain realistic motion from a volume of gas, the governing equations must be solved over time in three dimensions. The authors recently showed that for liquids, such volume calculations can be made with computational times and accuracy acceptable for a computer animation application if the environment and equations are suitably approximated [5]. A similar method is used here to solve the gas motion equations. A voxel-based scene approximation is combined with a numerical scheme known as finite differences. For (3) and (4), this leads to a straightforward algorithm that solves for the motion of a hot gas and takes into account arbitrary (approximated) objects as well as animator-controlled special effects. In addition, the method can be solved over a coarse grid without losing any of the behavioral characteristics of the gas, making it relatively efficient for even complex scenes.

3.1. Modeling the Simulation Environment

In order to solve the gas motion equations so that they represent the behavior of a gas in an animation environment, we need to represent the scene in a meaningful way with respect to the equations. We first approximate the scene as a series of cubic cells to reduce its complexity, and to form a grid upon which we can define temperature, pressure, and velocity.

A collection of solid 3D objects can be approximated as a series of regular voxels that are axially aligned to a coordinate system x,y,z (see Fig. 1a). If a portion of the medium (gas) surrounding the objects is likewise voxelized using the same coordinate system, then the boundaries of the objects can be made to coincide with the faces of gas voxels (Fig. 1b). The resulting grid can be used to solve physics-based differential equations in an efficient and straightforward way [5].

Consider a single cell in this grid (Fig. 2). It can be identified by its position relative to the origin in the x, y, z, directions, as i, j, k, respectively. At the center of the cell, we define variables $T_{i,j,k}$ and $p_{i,j,k}$ to represent the average temperature and average pressure within the cell. Likewise, in the center of each face of the cell we define a variable to
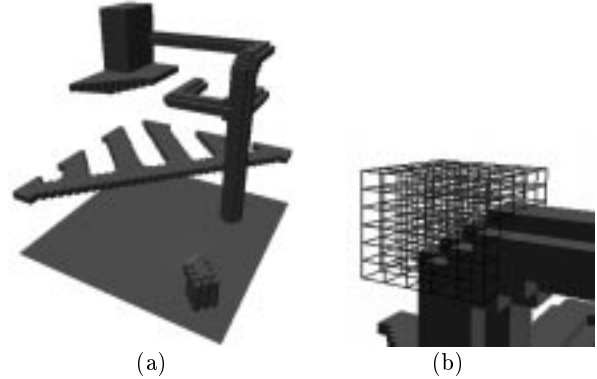

Figure 1: *Using regular voxels to approximate (a) a scene containing solid objects and (b) the medium around those objects.*
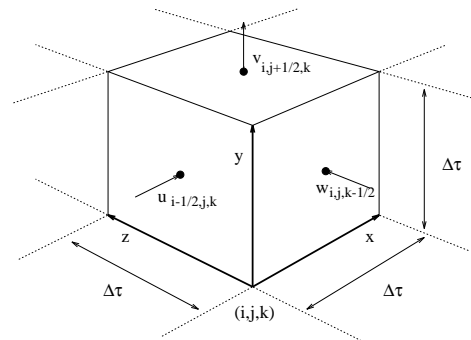

Figure 2: *Numbering convention for a single cell in the voxel grid. $\Delta\tau$ is the side length for each face of the cell.*

represent the gas velocity perpendicular to that face. This leads to the velocities u,v,w shown in Fig. 2. Intuitively, cells at i,j,k and $i+1,j,k$ will share the face velocity $u_{i+1/2,j,k}$. Once the environment over which we wish to calculate gas motion is discretized in this way, it is possible to calculate, using (3) and (4), how the temperature, pressure, and velocity throughout the grid vary over time. By using linear interpolation, the temperature (or velocity and pressure) at any point in the volume can be found. As an example, the u velocity of the gas at the center of the cell can be found from $(u_{i-1/2,j,k} + u_{i+1/2,j,k})/2$.

3.2. Applying the Equations to the Grid

To solve (3) and (4) we recast them to a form that is applicable to the regular voxel grid, using a numerical method called finite differences. A differential term such as

$$\frac{\partial T}{\partial y}$$

is approximated using a Taylor series to give a new expression for the derivative,

$$\frac{\partial T}{\partial y} = \frac{1}{2h}(T(y+h) - T(y-h)) + O(h^2), \tag{5}$$

where h is the finite distance over which the derivative is being taken, and $O(h^2)$ denotes that terms of order 2 or higher exist. Likewise, a second order derivative,

$$\frac{\partial^2 T}{\partial y^2},$$

is written as

$$\frac{\partial^2 T}{\partial y^2} = \frac{1}{h^2}(T(y+h) - 2T(y) + T(y-h)) + O(h^2), \qquad (6)$$

where h is as before. If h is taken to be $\Delta\tau$, the grid width, then for a single voxel, we approximate (6) such that

$$\frac{\partial^2 T}{\partial y^2} = \frac{1}{\Delta\tau^2}(T_{i,j+1,k} - 2T_{i,j,k} + T_{i,j-1,k})$$

using terms that correspond directly to variables on the voxelized grid, and ignoring terms of order 2 or higher (in h). Using this basic technique, (3) is first expanded as a series of first and second order differential terms,

$$\frac{\partial T}{\partial t} = \lambda\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right) - \frac{\partial Tu}{\partial x} - \frac{\partial Tv}{\partial y} - \frac{\partial Tw}{\partial z}, \qquad (7)$$

and then completely rewritten in terms of the free variables on the finite grid,

$$
\begin{aligned}
T_{i,j,k}^{n+1} =\ & T_{i,j,k}^{n} + \Delta t\{(1/\Delta\tau)[(Tu)_{i-1/2,j,k}^{n} - (Tu)_{i+1/2,j,k}^{n} \\
& + (Tv)_{i,j-1/2,k}^{n} - (Tv)_{i,j+1/2,k}^{n} + (Tw)_{i,j,k-1/2}^{n} \\
& - (Tw)_{i,j,k+1/2}^{n}] + \frac{\lambda}{\Delta\tau^2}[(T_{i+1,j,k}^{n} - 2T_{i,j,k}^{n} + T_{i-1,j,k}^{n}) \\
& + (T_{i,j+1,k}^{n} - 2T_{i,j,k}^{n} + T_{i,j-1,k}^{n}) \\
& + (T_{i,j,k+1}^{n} - 2T_{i,j,k}^{n} + T_{i,j,k-1}^{n})]\},
\end{aligned}
\qquad (8)
$$

where a term such as $(Tu)_{i+1/2,j,k}^{n}$ represents the temperature flow between cells (i,j,k) and $(i+1,j,k)$, and is calculated as

$$(Tu)_{i+1/2,j,k}^{n} = \frac{u_{i+1/2,j,k}^{n}}{2}(T_{i,j,k}^{n} + T_{i+1,j,k}^{n}).$$

Using (8), the temperature at the center of cell i,j,k at time $t + \Delta t$ can be found in terms of the temperatures at time t in adjacent cells. T^{n+1} denotes the value of T at time $t + \Delta t$, while T^{n}, denotes the value at time t. It is simply a matter of plugging in the old values of T in order to find the new value. In a similar way, (4) is also expanded as first and second order differentials, written in terms of cell face velocities and cell pressures, and then solved to find $\mathbf{u}_{i,j,k}^{n+1}$ in terms of $\mathbf{u}_{i,j,k}^{n}$ (see Appendix A). Thus, to find how the velocity and temperature change over a time interval Δt, (8) and (17) are applied simultaneously to each cell in the grid. Because $\Delta\tau$ is a constant, this calculation involves only floating point multiplication and addition, making it reasonably efficient. The change in pressure for a cell is calculated separately and is a fortunate side effect of mass conservation which is described in the next section.

3.3. Ensuring Accuracy

The approximation of the animation environment as regular voxels is the main source of efficiency for our algorithm. The drawback, however, is that low resolution variable sampling can introduce error into the calculation. Because the free variables $\mathbf{u}$ and T are sampled at fixed positions in space $\Delta\tau$ apart, an error of order $O(\Delta\tau^2)$ is introduced into T and $\mathbf{u}$ when the finite difference approximation is applied to the voxels (the $O(h^2)$ terms from (5) and (6)). For temperature this is not significant, but for $\mathbf{u}$ it represents mass that has been created (or destroyed) as a side effect of the algorithm. This means that each cell in the scene acts as a small gas source or sink, slightly altering the total mass of gas in a scene. To correct for this change in mass, we need to ensure that at any point in the scene (unless we specifically want a source or sink), the mass of gas flowing in, is the same as the mass flowing out. This can be characterized by a constraint equation that is actually part of the Navier-Stokes equations,

$$\nabla \cdot \mathbf{u} = 0. \qquad (9)$$

For a single grid cell, the left hand side of (9) is approximated using the Taylor series method, and rewritten in terms of the grid variables, giving

$$
\begin{aligned}
(\nabla \cdot \mathbf{u})_{i,j,k} =\ & \frac{1}{\Delta\tau}[u_{i+1/2,j,k} - u_{i-1/2,j,k} + v_{i,j+1/2,k} \\
& - v_{i,j-1/2,k} + w_{i,j,k+1/2} - w_{i,j,k-1/2}],
\end{aligned}
\qquad (10)
$$

where $(\nabla \cdot \mathbf{u})_{i,j,k}$ is the mass divergence at the center of the cell. For mass to be conserved, this scalar field must be zero in every cell. This requires a solution to the classic three dimensional Poisson equation. The computational method described by Harlow and Welch [8] was one of the earliest in print, and although that approach is two-dimensional in scope, it can be modified so that it is suitable for our gas model.

We define a potential field, ψ, which is sampled at the center of each grid cell and is initially zero everywhere. Then, for every frame of animation, we iterate over the grid, updating ψ according to

$$
\begin{aligned}
\psi_{i,j,k}^{h+1} =\ & \frac{2}{8/\Delta\tau^2}\{-(\nabla \cdot \mathbf{u})_{i,j,k} + \frac{1}{\Delta\tau^2}[\psi_{i+1,j,k}^{h} + \psi_{i-1,j,k}^{h} + \psi_{i,j+1,k}^{h} \\
& + \psi_{i,j-1,k}^{h} + \psi_{i,j,k+1}^{h} + \psi_{i,j,k-1}^{h}]\} - \psi_{i,j,k}^{h},
\end{aligned}
\qquad (11)
$$

where $(\nabla \cdot \mathbf{u})_{i,j,k}$ is given by (10). This field is considered to have converged, i.e., the iteration stops, when, for every cell in the grid,

$$\left|\frac{|\psi_{i,j,k}^{h+1}| - |\psi_{i,j,k}^{h}|}{|\psi_{i,j,k}^{h+1}| + |\psi_{i,j,k}^{h}|}\right| < \epsilon. \qquad (12)$$

For the examples given later in this paper, ϵ is taken to be on the order of 10^{-4}, and convergence is achieved in about 8-20 iterations per frame.

After convergence, the ψ field represents the relative discrepancy in mass between adjacent cells. By adjusting $\mathbf{u}$ according to the gradient in ψ, $\mathbf{u}$ can be made to satisfy (9) directly [8]. The velocity components on the grid cell faces are adjusted to correct for the divergence field by

$$
\begin{aligned}
u_{i+1/2,j,k}^{n+1} &= u_{i+1/2,j,k}^{n+1} - \frac{\psi_{i+1,j,k} - \psi_{i,j,k}}{\Delta\tau}, \\
v_{i,j+1/2,k}^{n+1} &= v_{i,j+1/2,k}^{n+1} - \frac{\psi_{i,j+1,k} - \psi_{i,j,k}}{\Delta\tau}, \\
w_{i,j,k+1/2}^{n+1} &= w_{i,j,k+1/2}^{n+1} - \frac{\psi_{i,j,k+1} - \psi_{i,j,k}}{\Delta\tau}.
\end{aligned}
\qquad (13)
$$

The temperature, $T_{i,j,k}^{n+1}$, need not be changed. This final step makes the necessary small adjustments in the velocity field to preserve mass and ensure that the calculation remains physically accurate. In addition, it can be shown that the gradient in the pressure field, $p_{i,j,k}$, is equal to the gradient in $\psi_{i,j,k}$ [8]. Because (4) depends only on the gradient in p, we can use the ψ field directly when calculating gas motion, instead of calculating the pressure.

3.3.1. Stability

An important issue with respect to accuracy is the numerical stability of the algorithm. Instability can occur when small oscillations in the variables resonate and dominate the solution. With the model we have described this can happen when the velocity of any part of the gas allows it to move further than $\Delta\tau$ in a single timestep. To ensure stability for an animation with a maximum gas velocity of $|\mathbf{u}|$, the timestep, Δt, must be set according to,

$$\Delta t\, |\mathbf{u}| < \Delta\tau. \qquad (14)$$

For all the examples given in this paper Δt was set to $\frac{1}{30}Sec$, to achieve the standard animation framerate. This is an order of magnitude lower than the maximum stable timestep for even the most violent of the examples shown. A further
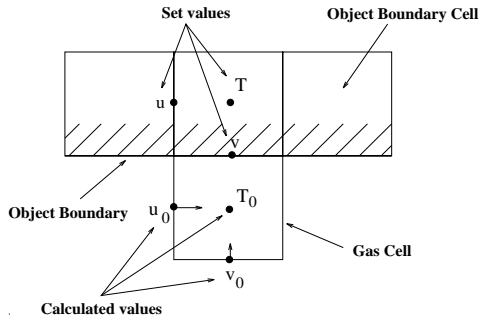
Figure 3: *Setting temperature and velocity conditions at the boundary between a gas and an object.*

Object Type	u	v	T	Result
Rough and rocky	$-u_0$	0	T_0	Lots of turbulence close to the object
Concrete	0	0	T_0	Some turbulence, object slows flow
Smooth Plastic	u_0	0	T_0	No turbulence, flow unaffected
Open Window	0	v_x	T_x	Gas can flow in or out depending on T_x and v_x
Hot Fan	0	v_x	T_x	Hot gas is forced into the scene
Steaming soup	0	0	T_x	Gas cells next to boundary are heated

Table 1: *Examples of different object boundary conditions. A subscript x represents a value chosen by the animator. A subscript 0 means that the value is taken directly from the adjacent gas cell (see Fig. 3).*

condition for numerical stability is a necessary feature of the finite-difference method and it forces a lower bound on the kinematic viscosity, ν. Linear analysis has shown that for the Navier-Stokes equations, ν must satisfy [4]

$$\nu > (\Delta t/2)max[u^2, v^2, w^2] \qquad (15)$$

for the system to remain stable.

3.4. Boundary Conditions for Special Effects

The regular voxel grid makes application of the gaseous motion equations efficient and straightforward. It also makes it easy to specify temperature, pressure, and velocity along the edges of solid objects so that interaction between objects and gas can be modeled accurately. Such "boundary conditions" can also be used to specify special effects involving gas flowing into or out of the environment. Referring to Fig. 3, the application of the finite difference forms of (3) and (4) to the gas cell may require grid values from an adjacent object cell. These values are set automatically depending on the type of material or object that the cell represents.

For example, a hot radiator cannot allow gas to pass through it, so the velocity, **u**, (v in the 2D figure) is set to zero for cell faces that represent the radiator boundary. Tangentially however, we want gas to flow freely along the surface. Therefore u, is set equal to the external tangential velocity u_0. Temperature flows freely from the radiator to the air, so the temperature, T, within the boundary cell is set to the desired temperature of the radiator. If a heating fan were being modeled instead of a radiator, then **u** on the object cell faces would be set to model air flowing into the environment. For a standard obstacle, such as a wall or table, **u** is set to zero and T is set to the ambient temperature. The pressure is more difficult to set with a desired effect in mind. Therefore the object pressure is simply set equal to the external gas pressure so that it has no local effect on the flow. There are no restrictions on how boundary conditions can be set. Some examples of **u** and T for interesting effects are given in Table 1.

3.5. The Turbulent Gas Algorithm

The complete algorithm for animating turbulent gas has two stages. The first involves decisions that need to be made by an animator in order to create a particular effect. The steps the animator must take are:

1. Subdivide the environment into regular voxels with side length $\Delta \tau$. The environment need not be rectangular, any arrangement is acceptable as long as voxel faces are aligned.
2. Select boundary conditions for velocity and temperature similar to those in Table 1.
3. Consider viscosity, thermal expansion, and molecular diffusion, and set ν, β, and λ accordingly (1/10 $\Delta \tau$ or higher

for little visible turbulence, 1/100 $\Delta \tau$ or lower for greater swirling).
4. Determine Δt from the minimum of $\frac{1}{30}^{th}$ of a second and the largest stable timestep given by (14) and (15).

After the parameters for the animation have been chosen, the automatic part of the process proceeds as follows:

5. Apply boundary conditions to the sides of objects chosen to simulate fans, heaters, sources, or sinks. Set the boundary velocity of other objects to zero, and set interior temperatures to the ambient temperature.
6. Use the finite difference approximations of (3) and (4) to update the temperature and velocity, $T_{i,j,k}$ and $\mathbf{u}_{i,j,k}$, for each cell (making use of ψ instead of pressure).
7. Use (10) to find the divergence field, $(\nabla \cdot \mathbf{u})$, for the gas to conserve mass.
8. While the iteration convergence condition, (12), is not satisfied,

   - Sweep the grid, calculating the relaxation adjustment, ψ, for each cell using (11).

9. Update the cell face velocities, $\mathbf{u}_{i,j,k}$, using (13).
10. Goto step 5.

This algorithm has been implemented on an SGI Indigo2 workstation using a simple interface to allow an animator to define obstacles, heat or steam sources and sinks, as well as moving fans, and to include them in an animation.

3.6. Rendering

There have been many approaches to rendering gaseous phenomena presented in recent years. A good discussion of them can be found in [17] and is not repeated here. To best illustrate the contributions of this paper, a rendering method involving suspended particles has been used. Massless particles are introduced into a scene and used to represent the local density of light-reflecting (or absorbing) matter. Once introduced, the particles are convected using the velocity field calculated from (4). The change in position of a particle k, at $\mathbf{x}_k$, over a single timestep is found from

$$\mathbf{x}_k^{n+1} = \mathbf{x}_k^n + \Delta t\, \mathbf{u}_{\mathbf{x}}^n,$$

where $\mathbf{u}_{\mathbf{x}}$ is found from the particle's position in the grid using linear interpolation. The particles themselves can be introduced as part of a boundary condition (proportional to T or **u** for example) or distributed however the animator wishes. The particles have no effect on the calculated motion, they are just used for rendering purposes to visualize how the density of smoke or steam changes as the gas medium moves.

Figure	Cell Resolution	Calc. Time (s/frame)	ψ cycles	Render Time (M/frame)
4	60x35x60	15.0	8	23
5	40x60x40	24.0	10	38
6	40x50x40	28.0	13	45
7	60x60x45	49.0	20	14

Table 2: *The calculation and rendering times for each of the examples. Cell resolution is approximate because the scenes are not rectangular. Cells that play only a small part in the motion of the gas are not used.*

For each frame of animation, the instantaneous distribution of particles is used as a density map for use with a volume renderer. There is no straightforward physics-based way to determine what density volume each particle represents or how many particles to use. This is dictated by the particular effect the animator wants (lots of very dense particles for smoke from burning tires, very few for smoke from a candle flame). The general formula for the examples shown here is to set each particle to represent $1/50^{th}$ of the volume of a single cell, and adjust its density according to the desired effect. The volume renderer used is similar to that described by Ebert and Parent [2]. For each pixel in an image, a viewing ray is cast through the density volume to find the effective opacity of the particle cloud as seen from the viewer. If desired, the ray can be subdivided, and for each subdivision, a ray is cast through the volume towards each light source. This significantly increases the cost of rendering, but it does allow for smoke and steam to self shadow and to fall under the shadow of other objects. This technique has been implemented as a volume shader for use with the BMRT implementation of the RenderMan Standard [7]. This shader was used for all the examples in this paper. It should be noted that the particle representation of suspended matter also makes the method ideal for rendering using Stam and Fiume's warped blobs [17].

4. Results

This paper has shown that the motion of a hot gas can be accurately calculated using an efficient low-resolution technique. In the following examples we illustrate the kind of rotational motion and gas/object interaction that is well suited to the method. All of the examples were calculated on an SGI Indigo2 with 64 Mb of memory. Table 2 gives the calculation times for each example, the approximate resolution of the environment, and the rendering time for a single image. Table 3 gives more specific information about each example including the width of each cell, the λ and ν coefficients, and the maximum gas velocity in the example.

Steam Valve

The images shown in Fig. 5 demonstrate the interaction of hot steam with solid objects. The voxel version of this environment is shown in Fig. 1. The steam is forced into the environment by setting both T and $\mathbf{u}$ boundary conditions on a set of voxels representing a pressure release valve. The input velocity is $0.3\,m/s$, and the steam temperature is $80^{o}C$. This is consistent with steam being vented from a boiler. The result is the billowing effect of the cloud of steam. In the animation, turbulence builds up just in front of the nozzle as the steam is vented at high velocity.

The same environmental conditions were also used to animate the interaction of steam from three separate valves. Three frames from this animation are shown in Fig. 6. The rotation caused by the cooling and mixing of the gas can be seen clearly in the full sequence. In both of the valve cases massless particles were introduced at an average rate of $2000/s$.

Figure	ν	$\Delta\tau$	λ	$|\mathbf{u}|$ max m/s
4	0.005	0.05	0.4	0.15
5	0.002	0.1	1.0	0.35
6	0.002	0.1	1.0	0.50
7	0.01	1.0	3.5	3.4

Table 3: *Parameters used to calculate each of the examples. In each case the thermal expansion coefficient, β, was 10^{-3}.*
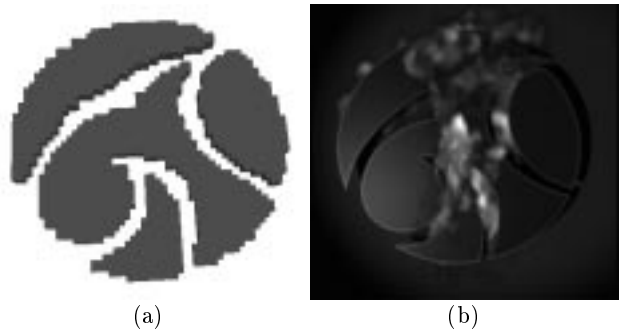

(a)   (b)

Figure 4: *a) A voxel approximation of the SIGGRAPH 97 logo. b) Smoke flowing smoothly around the approximation following the contours of the original shape.*

Smoke Stack

Figure 7 shows three frames from an animation of smoke rising from a chimney on a hot day. The boundary cells on the left of the grid are set to model a light wind of about $2\,m/s$ that occasionally gusts up to $3\,m/s$. We set the wind velocity to evolve according to the random-walk expression

$$\mathbf{u}_w^{n+1} = 0.98\,\mathbf{u}_w^n + 0.24\,\Delta t\,\phi(t),$$

where ϕ is a random number generator in the range [-1,1]. The value of $\mathbf{u}_w$ is clamped to lie in the range [2,3]. The constant coefficients have no physical significance, they are just parameters that have worked well for previous simulations. This wind is allowed to exit freely from the other end of the grid using the open window boundary condition from Table 2. The light wind sets up unstable conditions at the top of the tower causing the looping and swirling of the smoke as it moves. The smoke leaves the chimney at $0.8\,m/s$ with a boundary temperature of $46^{o}C$. From Table 2 it can be seen that nearly twice as many ψ iterations are required per frame. This is because the gusting windfield has a large component in every cell, so it takes longer for (11) to converge.

SIGGRAPH Logo

The final animation demonstrates that despite the low resolution voxel approximation, flow around complex objects can be accurately represented by our model. Figure 4a shows a voxel representation of the SIGGRAPH 97 logo, and Fig. 4b shows a frame from a sequence depicting smoke rising smoothly through it. The velocity along the boundary of the logo is set to zero to prevent smoke drifting into the artificial corners created by the approximation. When the smoke is released just beneath the symbol (with a temperature of $50^{o}C$) it flows over the object and conforms fairly closely to the original boundary.

5. Discussion of Limitations

The technique described in this paper derives efficiency by solving accurate equations at a low resolution. This is a compromise to try and preserve realism, and as such, it comes

with some limitations. Primarily, the method can only resolve rotational motion at a resolution lower than or equal to the grid resolution. From the examples shown, good effects can be achieved, but this means that grid resolution has to be increased to get finer motion within an existing scene. If we double the resolution and halve $\Delta\tau$ to get the same sized environment, we also have to halve the timestep, Δt, so that the system remains stable (from (14) and (15)). This is an inherent problem with finite differences. It could be compensated for by using a multi-resolution grid which would impose less of an overhead than using a higher resolution everywhere, but that is left as a topic for future work.

A second limitation of a finite difference grid is that cell orientation can affect the results. A gas jet oriented so that it travels diagonally through the cubic cells will tend to exhibit more diffusion than if it were moving parallel to an axis. In general, differences due to such diffusion is not significant (see Fig. 6), but it is something that can often be avoided by selecting grid orientation based on desired gas motion rather than objects in the static environment.

It is also desirable to integrate the gas model with other computer graphics techniques so that dynamic objects can interact with a gas. There is some discussion about how an iterative relaxation step like that described in Sec. 3.3 can be used to incorporate moving objects into animations of liquids in Foster and Metaxas [6] and Metaxas [9]. The methods used there are also applicable to the algorithm described in this paper, although that has not been explored in any detail.

6. Concluding Remarks

Numerous techniques exist for animating hot gases for computer graphics. Nearly all of them concentrate on achieving a visual approximation to the characteristic motion of a gas while getting as high a frame rate as possible. This sacrifices rotational and turbulent motion and often requires the animator to micro-control the flow. In this paper we have presented a new, alternative approach that models different scales of gas motion directly. This method accurately animates gaseous phenomena involving hot and cold gases, turbulent flow around solid obstacles, and thermal buoyancy, while leaving enough freedom for the animator to produce many different effects. The model is physics-based and achieves efficient computational speeds by using a combination of scene approximation and low resolution volume calculation. We have shown that even at these low resolutions, the characteristics of complex motion in the model are retained, and that exciting results can be obtained.

7. Acknowledgements

Appendix A: Finite Difference Form of the Motion Equations

The full expansion of (4) into first and second order derivatives is straightforward. Considering just the u velocity component for brevity, results in the following expression.

$$\frac{\partial u}{\partial t} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) - \frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} - \frac{\partial uw}{\partial z} - \frac{\partial p}{\partial x} \quad (16)$$

The finite difference scheme outlined in section 3.2 is then applied, (replacing p with ψ) giving the expression used to update the $u_{i+1/2,j,k}$ face velocity for cell i,j,k,

$$u_{i+1/2,j,k}^{n+1} = u_{i+1/2,j,k}^n + \Delta t\{(1/\Delta\tau)[(u_{i,j,k}^n)^2 - (u_{i+1,j,k}^n)^2$$
$$+ (uv)_{i+1/2,j-1/2,k}^n - (uv)_{i+1/2,j+1/2,k}^n + (uw)_{i+1/2,j,k-1/2}^n$$

$$-(uw)_{i+1/2,j,k+1/2}^n] + (\lambda/\Delta\tau^2)(u_{i+3/2,j,k}^n - 2u_{i+1/2,j,k}^n$$
$$+ u_{i-1/2,j,k}^n + u_{i+1/2,j+1,k}^n - 2u_{i+1/2,j,k}^n + u_{i+1/2,j-1,k}^n$$
$$+ u_{i+1/2,j,k+1}^n - 2u_{i+1/2,j,k}^n + u_{i+1/2,j,k-1}^n)$$
$$- \frac{1}{\Delta\tau}(\psi_{i,j,k}^n - \psi_{i+1,j,k}^n)\}, \quad (17)$$

where values that aren't defined on the grid are found by averaging as before.

References

1. Chiba, N., Ohkawa, S., Muraoka, K., and Miura, M., "Two-dimensional Simulation of Flames, Smoke and the Spread of Fire", J. of Vis. and Comp. Animation, 5(1), 1994, pp. 37–54.

2. Ebert, D.S., and Parent, R.E., "Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques", SIGGRAPH '90, Computer Graphics, 24(4), 1990, pp. 357–366.

3. Ebert, D.S., Carlson, W.E., and Parent, R.E., "Solid Spaces and Inverse Particle Systems for Controlling the Animation of Gases and Fluids", The Visual Comp., 10, 1994, pp. 179–190.

4. Fletcher, C.A.J., "Computational Techniques for Fluid Dynamics," Springer Verlag, Sydney, 1990.

5. Foster, N., and Metaxas D., "Realistic Animation of Liquids," Graphical Models and Image Proc., 58(5), 1996, pp. 471–483.

6. Foster, N., and Metaxas D., "Controlling Fluid Animation," Proceedings of CGI '97, To appear, 1997.

7. Gritz, L., and Hahn, J.K., "BMRT: A Global Illumination Implementation of the RenderMan Standard", J. of Graphics Tools, to appear, 1997.

8. Harlow, F.H., and Welch, J.E., "Numerical Calculation of Time-Dependent Viscous Incompressible Flow," Phys. Fluids, 8, 1965, pp. 2182–2189.

9. Metaxas, D., "Physics-Based Deformable Models: Applications to Computer Vision, Graphics and Medical Imaging", Kluwer-Academic Publishers, 1996.

10. Yaeger, L., Upson, C., and Myers, R., "Combining Physical and Visual Simulation - Creation of the Planet Jupiter for the Film "2010" ", SIGGRAPH '86, Computer Graphics 20(4), 1986, pp. 85–93.

11. Reeves, W.T., and Blau, R., "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems", SIGGRAPH '85, Computer Graphics 19(3), 1985, pp. 313–322.

12. Sakas, G., "Modeling and Animating Turbulent Gaseous Phenomena Using Spectral Synthesis", The Visual Computer, 9, 1993, pp. 200–212.

13. Sims, K., "Particle Animation and Rendering Using Data Parallel Computation", SIGGRAPH '90, Computer Graphics 24(4), 1990, pp. 405–413.

14. Shaw, C.T., "Using Computational Fluid Dynamics", Prentice Hall, London, 1992.

15. Shinya, M., and Fournier, A., "Stochastic Motion - Motion Under the Influence of Wind", Proceeding of Eurographics '92, September 1992, pp. 119–128.

16. Stam, J., and Fiume, E., "Turbulent Wind Fields for Gaseous Phenomena", SIGGRAPH '93, 1993, pp. 369–376.

17. Stam, J., and Fiume, E., "Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes", SIGGRAPH '95, 1995, pp. 129–136.

18. Wejchert, J., and Haumann, D., "Animation Aerodynamics", SIGGRAPH '91, Computer Graphics 25(3), 1991, pp. 19-22.

Figure 5: *An animation of steam discharge into a boiler room.*
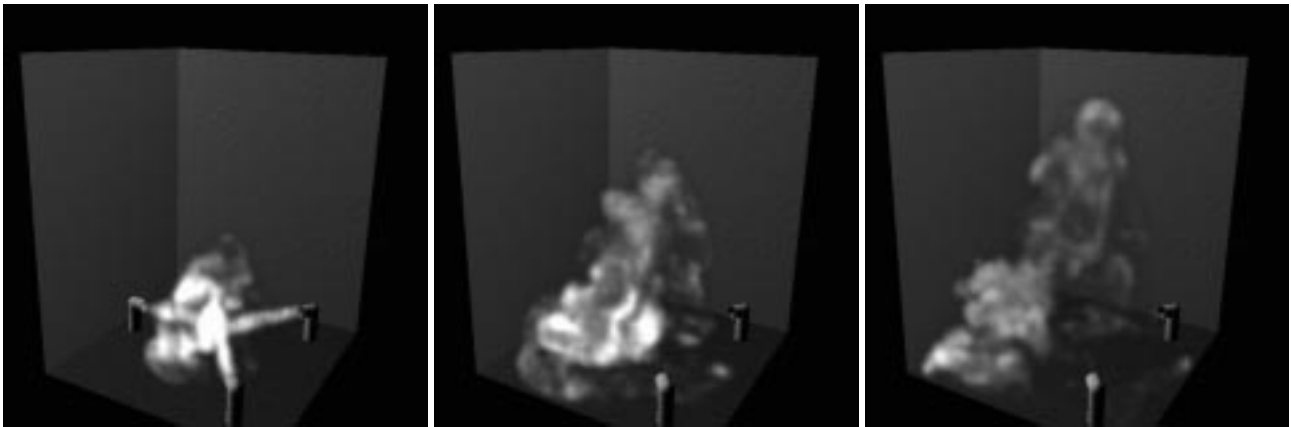

Figure 6: *Steam from three nozzles converges to cause vorticity and turbulence.*

Figure 7: *Turbulent smoke rolls out of a chimney into a light, gusting wind.*