Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes

Jos Stam
Eugene Fiume
Department of Computer Science, University of Toronto[1]

Abstract

Developing a visually convincing model of fire, smoke, and other gaseous phenomena is among the most difficult and attractive problems in computer graphics. We have created new methods of animating a wide range of gaseous phenomena, including the particularly subtle problem of modelling "wispy" smoke and steam, using far fewer primitives than before. One significant innovation is the reformulation and solution of the advection-diffusion equation for densities composed of "warped blobs". These blobs more accurately model the distortions that gases undergo when advected by wind fields. We also introduce a simple model for the flame of a fire and its spread. Lastly, we present an efficient formulation and implementation of global illumination in the presence of gases and fire. Our models are specifically designed to permit a significant degree of user control over the evolution of gaseous phenomena.

Keywords: fire, smoke, gaseous phenomena, diffusion, advection, warped blobbies, light transport, multiple scattering, particle systems, turbulence.

1 Introduction

The interplay of light with gases, aerosols and dust spans across the most visually delicate and the most explosive of natural phenomena. The depiction of these phenomena has been of great interest to computer graphics for over a decade, and their application to computer animation is clear.

Any graphical model of a gaseous phenomenon must have three components: a representation of the gas, a model for its spatiotemporal behaviour, and an illumination model to determine its appearance. Graphical models for gaseous phenomena have fallen into two classes. Models from one class combine the representation of the gas and its motion into a solid texture, animating the gas by varying parameters of the texture [14, 7]. Obtaining convincing motion with this approach becomes difficult because of the non-physical nature of the parameters and the cost associated with visualizing the solid texture. These parameters can be related, however, to a statistical model of turbulence [20].

We prefer the alternative approach, which keeps the three components intact, and animates the gas using wind fields. The earliest related work in computer graphics is by Reeves [17], in which particle systems were used to evoke a variety of visual effects, including fire. Recent work has focussed on the depiction of specific phenomena such as steam, mist, clouds or fire (e.g., [24, 4]). In these models, a wind field both advects (i.e., displaces) and diffuses "blobs" of density over time. Thus an immediate benefit is that the motion of a gas can be seen in real-time. However, the regular shape of the blobs often makes the gas look artificial. In this work, we improve the blob model by nonuniformly modifying the shape of the field over time. The field is warped by the surrounding wind field in a manner consistent with a real blob of gas (see Figure 8). The corresponding results are more convincing, using an order of magnitude fewer particles than our previous work [24].

The rendering of gases has been an active area of computer graphics research since the seminal work of Blinn [2]. Most, if not all, of the algorithms first sample the density and the optical properties of the gas on a grid and then solve for illumination. In a first pass, the effects of multiple scattering are resolved. Depending on the scattering distribution functions employed, different possibilities arise. For isotropic or constant scattering, a zonal radiosity-style method can be employed [18]. For arbitrary scattering, researchers have used a spherical harmonics expansion [10], a discretization of the angles [11] or brute-force ray-tracing [1]. These approximations are known respectively as the P_N method, discrete ordinates and Monte-Carlo simulation in the radiative transfer literature [6]. Once the effects of multiple scattering are computed, a final image is obtained by voxel-traversal algorithms [7, 19, 12]. However, the grid approximation has two deficiencies: the grid is memory intensive for complicated gas geometries, and it introduces sampling artifacts that are evident in animation. The appearance of artifacts can be alleviated by prefiltering the data on the grid prior to voxel traversal rendering [21], at the cost of diminished visual detail.

In view of these limitations, we have developed a new rendering algorithm which is an extension of the ray-tracing progressive refinement solution for radiosity [25]. Specifically, shooting operations both to and from gaseous blobs are derived. In addition, we use a diffusion approximation to resolve the effects of multiple scattering. Once the scattered intensity is resolved, it can be integrated to yield images of the gas from arbitrary viewpoints using an efficient blob tracer [24]. Our algorithm has been designed to capture the essential visual features of gaseous phenomena as efficiently as possible.

We are fascinated with the depiction of fire. The general phenomenon of fire and combustion "in spite of their fundamental importance and practical applications, are far from being fully understood. This is due, above all, to their interdisciplinary character and great complexity" [5]. Perhaps because of this, there has been little actual progress in visual models for fire. A simple laminar flame was texture mapped onto a flame-like implicit primitive and then volume-traced by Inakage [8]. Recently, two groups of re-

[1] 10 King's College Circle, Toronto, Ontario, Canada, M5S 1A4
E-mail: {stam|elf}@dgp.toronto.edu

searchers have worked on the problem of the spread of fire.[1] Perry and Picard apply a velocity spread model from combustion science to propagate their flames [15]. On the other hand, Chiba et al. compute the exchange of heat between objects by projecting the environment onto a plane [4]. The spread of the flame is a function of both the temperature and the concentration of fuel. In this paper, we present a similar model in three dimensions for the creation, extinguishing and spread of fire. The spread of the fire is entirely controlled by the amount of fuel available, the geometry of the environment and the initial conditions (i.e., where we light the match). Although our model relies on physical equations, it should not be confused with a physically accurate simulation of fire. Indeed, the latter is still an active area of research in combustion science. This paper focuses on producing convincing physically-motivated depictions of the motion and appearance of fire.

The structure of the paper is as follows. In the next section, we discuss the general methodology we use to model gaseous phenomena. In Section 3 we introduce diffusion equations for the evolution of the density, temperature and diffuse intensity of gases. In Section 4 an efficient solution to the diffusion equations is derived using "warped" blobs. In Section 5 we outline our solution to the intensity field. Then, in Section 6 we advance our simple flame model, discussing the results in Section 7. Finally, in Section 8 we discuss our conclusions, pointing towards future research.

2 Overview of the Method

A gas is described by physical quantities that vary as a function of space and time. These quantities include the density of gas particles, and the surrounding velocity and temperature fields. Generally, the relationships governing these quantities are strongly coupled and are known as the Navier-Stokes equations. For example, the temperature field introduces gradients in the velocity field, but the velocity field advects and diffuses the temperature field. With reacting gases such as fire, additional equations are needed to account for the underlying chemical reactions. The full physical simulation of this set of equations is prohibitively expensive. Their nonlinear nature and the wide range of scales required would severely strain computational speed and memory available on even the most powerful computers. Even if such a simulation were available, it would be of limited interest to computer graphics, because of the user's inability to control the phenomenon. Certain effects would not be achievable. The degree of user control must be balanced against the need for the turbulent behaviour of a gas. Turbulence is difficult to model without an underlying physical or statistical model driving it.

In our model, an animator governs the behaviour of a gas by specifying a wind field. The animator manages the global motion of the gas by using smooth fields, and controls the small scale behaviour by modifying the statistical parameters and scale of a turbulent wind field as in [24]. The latter field adds complexity to the motion. In addition, we assume that the motion field is incompressible so that its density ρ_f is constant. Once the wind field is given, we employ an advection-diffusion type equation to compute the evolution of both the density field and the temperature field. Diffusion type equations are employed for two reasons. First, they capture the main characteristics of many transport phenomena. Second, they are simple enough to be understood by an animator with a limited knowledge of physics. Indeed, most of us are familiar with diffusion processes since they are ubiquitous in everyday life (e.g., milk dissolving in a coffee cup). Given a user-specified wind field $\mathbf{u}(\mathbf{x}, t)$, the evolution over time of a scalar field $\theta(\mathbf{x}, t)$ is assumed to be governed by the following diffusion process:

$$\frac{D\theta}{Dt} = \kappa_\theta \nabla^2 \theta + S_\theta - L_\theta,$$

where $D\theta/Dt = \partial\theta/\partial t + \mathbf{u} \cdot \nabla\theta$ is the *total derivative* giving the variation over time of the field θ on a particle advected by the wind field $\mathbf{u}$. Apart from the wind field, the evolution is characterized by a diffusion coefficient κ_θ, sources S_θ and sinks L_θ. For rapidly evolving phenomena such as the propagation of light, only the *steady state* $D\theta/Dt = 0$ of this equation is considered.

By approximating the scalar field by a set of fuzzy particles, a user is able to visualize the effect of the wind field in real-time on a graphics workstation. Finally, when the user is satisfied with a particular behaviour, the fuzzy particles can be rendered by a global illumination algorithm for high-quality animations.

3 Diffusion Processes

3.1 Density and Temperature

The main physical characteristics of a gas are described by its density ρ, its temperature T, its velocity $\mathbf{u}$ and its radiative properties.[2] The evolution of the density is given by a diffusion equation. The diffusion coefficient in this case models the mixing caused by the small scales of the turbulence not modelled by our wind fields [24]. The sink term is modelled as simple decay over time at a constant rate: $L = \alpha\rho$. The source is either specified by a user or is related to a simple model of chemical reactions (see Section 6). A similar equation for the evolution of the temperature is reached by assuming that the kinetic energy is negligible compared to the heat released by the gas, and that buoyancy and pressure fluctuations are small [5]:

$$c_p \rho_f \frac{DT}{Dt} = \kappa_T \nabla^2 T + S_T - L_T.$$

Often, this reduced equation is utilized in theoretical investigations of the propagation of flames [3]. It is too simple to yield physically accurate simulations. However, we employ this equation because it captures some essential features of the flame. The specific heat at constant pressure c_p characterizes the efficiency of the fluid to release heat. The density of the fluid ρ_f is supposed to be constant, since our wind fields are incompressible[3]. The exact form of the source, sink and absorption terms depend on the type of gas. We explore this specifically in the case of fire in Section 6.

3.2 Intensity Field

A gas modifies the intensity field of light $I_\lambda(\mathbf{x}, \mathbf{s})$ by scattering, absorption and emission. The emission of a gas in local thermodynamic equilibrium is proportional to *black-body emission* [6]:

$$Q_\lambda = E_\lambda B_\lambda(T) = E_\lambda \frac{2h}{\lambda^5 c} \left[\exp\left(\frac{hc}{\lambda k T} \right) - 1 \right]^{-1},$$

where E_λ models the contribution of each wavelength λ to the the emission, h is Planck's constant, k is Boltzmann's constant and c is the speed of light. The total emission over all frequencies is proportional to the fourth power of the temperature:

$$\int_0^\infty B_\lambda(T)\, d\lambda = \sigma_{SB} T^4, \qquad (1)$$

[1]This is of course in the context of computer graphics. There is an enormous literature on fire in various areas, including forestry, chemistry, and physics.

[2]To shorten the notation, explicit dependence of the fields on all its variables is not always included.

[3]The density ρ_f of the fluid should not be confused with the density of microscopic particles ρ advected and diffused by the fluid.

where σ_{SB} is Stefan-Boltzmann's constant. In order to shorten the notations somewhat, we drop the explicit dependence on wavelength from the following radiative properties. The scattering properties of a gas are characterized by its *albedo* Ω and its *phase function* $p(\mathrm{s},\mathrm{s}')$. The albedo gives the fraction of light that is scattered versus that which is absorbed. The phase function models the spherical distribution of the scattered light. Although various distributions exist for different types of gases, we assume that the following reduced description is sufficient

$$p(\mathrm{s}\cdot\mathrm{s}') = 1 + \bar{\mu}\,\mathrm{s}\cdot\mathrm{s}'. \qquad (2)$$

where $\bar{\mu} = 3/4\pi \int_{-1}^{+1} \mu p(\mu) d\mu$ is the *first moment* of the phase function and characterizes the anisotropy of the scattering. Values of $\bar{\mu}$ near $+1$ indicate a preference for forward scattering. Values near -1 indicate predominantly backward scattering. A direct consequence of the simple distribution is that the scattered intensity depends only weakly on its angular variable:

$$\mathcal{S}\{I(\mathrm{x},\mathrm{s})\} = \frac{1}{4\pi}\int_{4\pi} p(\mathrm{s}\cdot\mathrm{s}')I(\mathrm{s}')\ d\mathrm{s}' = I^0(\mathrm{x}) + \frac{\bar{\mu}}{3}\mathbf{I}^1(\mathrm{x})\cdot\mathrm{s}. \quad (3)$$

where I^0 and $\mathbf{I}^1$ are known as the *average intensity* and *average flux* respectively. These two functions actually correspond to the first four coefficients of the intensity field into a spherical harmonics basis. In fact, the derivations outlined in this paper can be generalized to higher-order expansions[22].

The number of interactions per unit length of the light field with the gas is proportional to the density of the gas and is equal to $\sigma_t\rho$, where σ_t is called the *extinction cross-section.*

The gas diminishes the intensity of light traveling along a ray $\mathrm{x}_u = \mathrm{x}_0 - u\mathrm{s}$ by absorption and outscatter, and increases the intensity through self-emission and inscatter. The intensity reaching a point x_0 along a direction s is then equal to two contributions. The first is a portion of the intensity leaving the point of intersection x_b of the ray with a background surface. The second contribution is the light created within the gas:

$$I(\mathrm{x},\mathrm{s}) = I^{out}(\mathrm{x}_b)\tau(0,b) + \int_0^b \sigma_t\rho(\mathrm{x}_u)\tau(0,u)J(\mathrm{x}_u,\mathrm{s})d\mathrm{s}. \quad (4)$$

where $\tau(v,w) = \exp(-\int_v^w \sigma_t\rho(\mathrm{x}_u)du)$ is the *transparency*, and the *source intensity* is the sum of single-scattering, multiple scattering and self-emission:

$$J = J_s + J_d + (1-\Omega)Q, \qquad (5)$$

The *single-scattering intensity* J_s accounts for the first scatter of the incident intensity I_i entering the gas: $J_s = \Omega\mathcal{S}\{I_i\}$. The *diffuse intensity* J_d on the other hand, is entirely created within the gas through the phenomenon of multiple scattering. Since we have restricted ourselves to simple scattering distributions, the diffuse intensity in fact satisfies a diffusion equation (see Appendix A). This is a well known approximation in transport theory and is valid when the number of interactions of light with the gas is high, specifically when the dimensionless number $\sigma_t\rho l_0$ is high, where l_0 is a length characterizing the scales involved. For atmospheric scattering the approximation is usually considered to be valid when the density is higher than 0.01 [9].

4 Blob Solution of Diffusion Equations

Diffusion equations can be solved numerically using stable finite difference schemes [16]. In this case, the solution is sampled on a grid. Hence, the method can only resolve scales that are bigger than the grid spacing. These values, then, are interpolated to yield a "smoothed" version of the exact solution. Unfortunately, for three-dimensional problems grid-based schemes are intractable, because of memory limitations. Consequently, we develop an alternative method of solution by generalizing the above smoothing process. Generally, the approximate solution can be represented as the convolution of the exact solution with a *smoothing kernel* $W_\sigma(r)$:

$$\theta_\sigma(\mathrm{x},t) = \int W_\sigma(|\mathrm{x}-\mathrm{x}'|)\theta(\mathrm{x}',t)\ d\mathrm{x}',$$

where the σ corresponds to the grid spacing. The smoothing kernel is normalized and tends towards a delta distribution as $\sigma \rightarrow 0$. The latter two conditions are required such that the exact solution is recovered when the grid spacing goes to zero. Because the smoothing kernel depends solely on distance, the approximation is of order two [13]. Therefore, the field θ can always be replaced by its smoothed equivalent to within the order of accuracy of the smoothing process itself, e.g.,

$$(\theta\gamma)_\sigma = \theta_\sigma\gamma_\sigma + O(\sigma^2). \qquad (6)$$

Instead of a grid, we consider a set of samples $\{\mathrm{x}_k(t)\}_{k=1}^N$ evolving over time. By assigning a mass $m_k(t)$ to each sample, we represent the density as $\rho(\mathrm{x},t) = \sum_{k=1}^N m_k(t)\delta(\mathrm{x}-\mathrm{x}_k(t))$. Then, a smoothed approximation of the density field is given by:

$$\rho_\sigma(\mathrm{x},t) = \sum_{k=1}^N m_k(t)W_\sigma(|\mathrm{x}-\mathrm{x}_k(t)|),$$

i.e., the density is a superposition of *blobs* centred at the sample locations. This representation induces one for other fields using Eq. 6, i.e., $(\rho\theta)_\sigma = \rho_\sigma\theta_\sigma$. Hence,

$$\theta_\sigma(\mathrm{x},t) = \frac{1}{\rho_\sigma(\mathrm{x},t)}\sum_{k=1}^N m_k(t)\theta_k(t)W_\sigma(|\mathrm{x}-\mathrm{x}_k(t)|),$$

where $\theta_k(t) = \theta(\mathrm{x}_k(t),t)$.

An approximate solution to the diffusion equation is obtained naturally when the samples move along the wind field. Because this equation is linear, it is enough that each blob satisfies it. A sufficient (and convenient) condition for each blob k to satisfy the diffusion equation is that the smoothing kernel is a Gaussian:

$$W_{\sigma_k(t)}(r) = \frac{1}{(2\pi)^{3/2}\sigma_k^3(t)}\exp\left(-\frac{r^2}{2\sigma_k^2(t)}\right),$$

with standard deviation proportional to $\sqrt{\kappa_\theta t}$. The diffusion then expands the size of the blobs over time. The sink term is satisfied if the coefficients $\theta_k(t)$ satisfy

$$\frac{d}{dt}\theta_k(t) = -L_\theta(\mathrm{x}_k(t),t).$$

We use the source term of the density to generate new blobs at each time step of the simulation. From the source term, we can define a probability density distribution for each time t:

$$\varphi_\rho(\mathrm{x},t) = S_\rho(\mathrm{x},t)\bigg/\left(\int S_\rho(\mathrm{x}',t)\ d\mathrm{x}'\right).$$

Quite often, the source term is constant on a given domain, and the density distribution is uniform. The location x_k of the new blob hence is determined by generating a random point from this distribution. The initial mass of each blob is a function of the

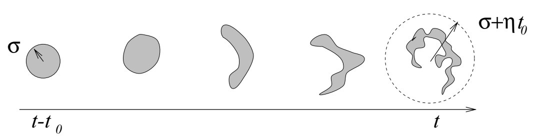


Figure 1: Blob Warping

number N_0 of new blobs per time step and of the initial size σ_0 of each blob:

$$m_k = (2\pi)^{3/2}\sigma_0^3/N_0 S_\rho(\mathbf{x}_k, t)\Delta t,$$

where Δt is the time step. The animator determines the number of new blobs and the initial size.

The preceding procedure is a generalization of our blob solution for the evolution of a density distribution in a moving fluid [24]. However, after a large amount of diffusion, that is, with sufficient simulation time, the increasing variance causes the spherical shape for each blob to become apparent. Real gases, and especially fire, are poorly approximated by a superposition of spherical blobs, with the exception perhaps of very billowy smoke. The problem is due to the fact that as the blobs get bigger, the "shape" of the actual distribution is not uniform but is in fact advected by the velocity field (see Figure 1). To account for this nonuniformity, we shall more accurately track the shape of each blob as the advection of a regular blob over a fixed time t_0. To achieve this, we could take samples in the blob at some time $t - t_0$ and then trace each sample over an interval t_0 through the wind field. However, this introduces sampling artifacts. A better approach is to backtrace from the warped blob toward the initial blob (see Figure 1). For each point $\mathbf{x}$, there corresponds a point $\mathbf{x}^{-1}$ obtained by tracing the point back through the wind field for a time t_0:

$$\mathbf{x}^{-1} = \mathbf{x} - \int_t^{t-t_0} \mathbf{u}(\mathbf{x}(t'), t')\, dt'. \qquad (7)$$

To include this warping in our simulations, we replace the smoothing kernel by one evaluated at the backtraced points:

$$W_{\sigma_k(t)}(|\mathbf{x} - \mathbf{x}_k(t)|) \longrightarrow W_{\sigma_k(t)-\eta t_0}\left(|\mathbf{x}^{-1} - \mathbf{x}_k^{-1}(t)|\right),$$

where η is a factor accounting for the spread of the blob due to advective effects; it is a function of the magnitude and scale of the wind field at the point $\mathbf{x}$. In our implementation, it is a user-specified constant. The extra cost of the blob-warping method is the evaluation of the above integral, which can be achieved by a simple Eulerian scheme.

5 Resolving the Intensity Field

As in the density and temperature fields, we represent the source intensity into a superposition of blobs:

$$J(\mathbf{x}, \mathbf{s}) = \frac{1}{\rho(\mathbf{x})}\sum_{k=1}^{N} m_k J_k(\mathbf{s}) W_\sigma(|\mathbf{x} - \mathbf{x}_k|).$$

The angular variation of each coefficient in this expansion is determined by the shape of the phase function and is therefore (see Eq. 3):

$$J_k(\mathbf{s}) = J_k^0 + \mathbf{J}_k^1 \cdot \mathbf{s}.$$

These coefficients are computed by utilizing an extension of the shooting algorithm used in diffuse environments [25]. The surfaces of the environment are first discretized into an ensemble of patches. At each step of the algorithm, the outgoing intensity I^{out} from one patch is shot into the environment and is collected at the other patches of the environment. We extend this work by including shooting operations from a patch to the blobs and conversely, from a blob to the patches. In addition, we develop shooting operations from non-physical lights, such as directional sources, to the blobs. These shooting operations resolve the single scattering intensity. The diffuse intensity is obtained by solving Eq. 11. A summary of the algorithm is given next.

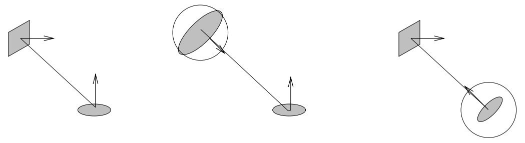


Figure 2: Patch to Patch Blob to Patch Patch to Blob

```
Set source intensity to emission: $J = (1-\Omega)Q$.
Collect intensity from light sources: $J \leftarrow J + \Omega\mathcal{S}\{I_{lights}\}$.
do
    Shoot from patches to other patches.
    Collect intensity from patches: $J \leftarrow J + \Omega\mathcal{S}\{I_{patches}\}$.
    Compute diffuse intensity: $J_d$.
    Add to source intensity: $J \leftarrow J + J_d$.
    Shoot source intensity to the patches.
Until converged
```

This algorithm is similar to the one presented in [11].

5.1 Shooting Operations

To derive our shooting operations, we consider the basic interchange of intensity between a surface area A_2 and an area dA_1 separated by a distance d_{12} along a direction $\mathbf{s}_{12}$:

$$I_{12}^{in} = I_2^{out} F_{12}, \qquad (8)$$

where F_{12} is the form factor between the two surfaces. When A_2 is a disk, the form factor can be approximated by [25]:

$$F_{12} = \frac{A_2}{A_2 + d_{12}^2}(\mathbf{n}_1 \cdot \mathbf{s}_{12})(-\mathbf{n}_2 \cdot \mathbf{s}_{12}),$$

where $\mathbf{n}_1$ and $\mathbf{n}_2$ denote the normals at A_2 and dA_1, respectively. The contribution of the intensity leaving A_2 to the outgoing intensity I_1^{out} is then given by $I_{12}^{out} = r_1 I_{12}^{in}$, where r_1 is the reflectance of the receiving patch.

A single shooting operation from a patch to a blob is a special case of Eq. 8, since an infinitesimal receiving surface at the centre of the blob can always be aligned along $\mathbf{s}_{12}$, i.e., $\mathbf{n}_1 \cdot \mathbf{s}_{12} = 1$. The contribution to the source intensity at the blob is then

$$J_{12}(\mathbf{s}) = I_2^{out} F_{12} \Omega (1 + \bar{\mu}\, \mathbf{s} \cdot \mathbf{s}_{12}).$$

Therefore, the first two coefficients in the angular expansion of the source intensity are updated as

$$J_1^0 \leftarrow J_1^0 + \Omega I_2^{out} F_{12} \text{ and } \mathbf{J}_1^1 \leftarrow \mathbf{J}_1^1 + \Omega\bar{\mu} I_2^{out} F_{12}\, \mathbf{s}_{12}.$$

Similarly, the shooting operation from a blob to a patch is achieved by considering a disk of area $A(\sigma) = 2\pi\sigma^2$ at the centre of the blob aligned with the direction $\mathbf{s}_{12}$, i.e., $-\mathbf{n}_2 \cdot \mathbf{s}_{12} = 1$. The outgoing intensity at the patch then is equal to

$$I_1^{out} \leftarrow I_1^{out} + r_1 J_2(\mathbf{s}_{12}) F_{12}. \qquad (9)$$

This step can be sped up by constructing a hierarchical tree-data structure of the blobs [24]. Instead of shooting from a single blob, we shoot from the centre of mass of each blob cluster.

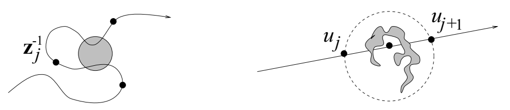


Figure 3: Inverse Warp of the Ray

We have also included shooting operations from non-physical light sources to the blobs, since these lights are extremely useful in creating many visual effects. The contribution of a directional light source of intensity I_{dir}^0 and direction $\mathbf{s}_0$ is achieved by:

$$J_1(\mathbf{s}) \leftarrow J_1(\mathbf{s}) + I_{dir}^0 \Omega (1 + \bar{\mu}\, \mathbf{s} \cdot \mathbf{s}_0).$$

The shooting operations from point lights and spotlights can be derived likewise.

Shadowing by surfaces is included in these calculations by multiplying each form factor by an occlusion term. Similarly, partial shadowing caused by the gas is included by multiplying the form factor by the transparency between the shooting and receiving elements. Aliasing can be avoided by shooting from a subdivision of the shooting blob/patch [25].

We could derive similar shooting operations between blobs. However, since usually there is a large overlap of blobs, the disk form factor is not accurate. Also, in the case of many blobs, these shooting operations become very expensive. Instead, we solve for the first two coefficients in the angular expansion of the source intensity using the diffusion approximation (see Eq. 11).

5.2 Multiple Scattering

When the blob representation of the diffuse intensity is inserted into the diffusion equation (Eq. 11) we get the following equation:

$$\sum_{k=1}^{N} m_k J_{d,k}^0 \left(\nabla \kappa_d(\mathbf{x}) \nabla W_k(\mathbf{x}) - \alpha_d(\mathbf{x}) W_k(\mathbf{x}) \right) + S_d(\mathbf{x}) = 0,$$

where $W_k(\mathbf{x}) = W_{\sigma_k}(|\mathbf{x} - \mathbf{x}_k|)/\rho(\mathbf{x})$. By setting $\mathbf{x} = \mathbf{x}_l$, for $l = 1, \ldots, N$ we get a system of N-linear equations for the unknowns $J_{d,k}^0$ which can be solved using an LU-decomposition for example [16]. Once we compute a solution, the coefficients of the source intensity are updated:

$$J_k^0 \leftarrow J_k^0 + J_{d,k}^0 \;\text{ and }\; \mathbf{J}_k^1 \leftarrow \mathbf{J}_k^1 + \mathbf{J}_d^1(\mathbf{x}_k),$$

where $\mathbf{J}_d^1$ is given by Eq. 12. This method is in fact equivalent to a finite-element solution for the diffusion equation. When the gas does not intersect any surfaces, the boundary conditions are naturally satisfied by the blob representation. For more details about the method and the boundary conditions, see [23].

5.3 Integrating the Transport Equation

Once an approximation of the source intensity is computed, the intensity field at any point in the environment is obtained by integrating the scattering equation along a ray (Eq. 4). By truncating the domain of the blobs, we reduce the number of blobs intersecting a particular ray. These intersections define a partition of the ray into disjoint intervals $[u_j, u_{j+1}]$, with $j = 0, \ldots, M-1$ [24]. To take into account the warping of the blob, we transform each interval backwards as illustrated in Figure 3. The density on each interval is approximated by its value at the point $\mathbf{z}_j^{-1}$ calculated by backwarping the midpoint $\mathbf{z}_j = (\mathbf{x}_{u_j} + \mathbf{x}_{u_{j+1}})/2$ of the interval:

$$\rho_j = \sum_k m_k W_{\sigma_k} \left(|\mathbf{z}_j^{-1} - \mathbf{x}_k^{-1}| \right),$$

where the sum is over the blobs which overlap the j-th interval. Consequently, both the transparency and the source intensity on each interval are approximated by:

$$\begin{aligned} \tau_j &\approx \exp(-\sigma_t (u_{j+1} - u_j) \rho_j) \;\text{ and} \\ J_j(\mathbf{s}) &\approx 1/\rho_j \sum_k m_k J_k(\mathbf{s}) W_{\sigma_k} \left(|\mathbf{z}_j^{-1} - \mathbf{x}_k^{-1}| \right), \end{aligned}$$

respectively. The evaluation of the integral can be performed by traversing the intervals from front to back:

```
for all rays in ray-trace tree do
    $I = 0$
    $\tau_{tot} = 1$
    for $j = 0, \ldots, M-1$ do
        $I \leftarrow I + \tau_{tot}(1 - \tau_j) J_j(\mathbf{s})$
        $\tau_{tot} \leftarrow \tau_{tot} \tau_j$
        if $\tau_{tot}$ <EPS then exit
    end for
    $I \leftarrow I + \tau_{tot} I^{out}(\mathbf{x}_b)$
end for
```

6 A Simple Fire Model

Flames result from the combustion of fuels and oxidizers. As the molecules of these compounds meet at a sufficiently high temperature, a chemical reaction becomes possible. The resultant burning compounds are called the *flame*. We are not interested in a complete physical model for this reaction, but rather with those mechanisms essential to a good visual representation. In particular, we shall derive simple but effective models for the evolution of density fields giving the concentration of flames, smoke, and fuel. Flames and smoke can be subsequently rendered. We first describe the source and sink terms appearing in the diffusion equations for the density of the fuel, flame and smoke.

Given the density ρ_{fuel} of the fuel and its temperature T_{fuel} at a given point, the rate of production of the flame density ρ_{flame} is given by the *Arrhenius formula*. Assuming a constant concentration of oxidants [3],

$$S_{\rho,\text{flame}} = L_{\rho,\text{fuel}} = \nu_a \exp\left(-\frac{T_a}{T_{\text{fuel}}} \right) \rho_{\text{fuel}}. \qquad (10)$$

T_a is the *activation temperature*, which is directly related to the energy E_a released during the reaction by $T_a = E_a/R$, where R is the universal gas constant. The term ν_a is a "frequency", depending on the exact nature of the combustibles, characterizing the rate of the reaction.

Most naturally occurring fires create smoke particles as the flame cools down. To our knowledge, no definite analytical models of this exist. To model the creation of a density of smoke ρ_{smoke}, we use an equation similar to that used to produce burning fuel:

$$S_{\rho,\text{smoke}} = \nu_b \exp\left(-\frac{T_{\text{flame}}}{T_s} \right) \rho_{\text{flame}},$$

where T_s is the temperature below which smoke particles start to form, and ν_b is another material-dependent constant.

The initial temperature of the flame is related to the heat released during the reaction modelled by Eq. 10. The source term appearing in the diffusion equation for its evolution is then $S_{T,\text{flame}} = c_p S_{\rho,\text{flame}} T_a$. The temperature of the flame decreases mainly through radiating heat. Since emission dominates in flames, the loss of heat is equal to the contributions of the radiation from all angles and all wavelengths:

$$L_{T,\text{flame}} = \int_0^\infty \int_{4\pi} \sigma_{t,\lambda} \rho Q_\lambda(T) \; ds \; d\lambda = 4\pi \rho \alpha_{\text{flame}} \sigma_{SB} T^4,$$

where the *average absorption cross-section* is defined by

$$\alpha_{\text{flame}} = \frac{\int_0^\infty \sigma_{t,\lambda} E_\lambda B_\lambda(T)\ d\lambda}{\int_0^\infty B_\lambda(T)\ d\lambda} = \frac{1}{\sigma_{SB} T^4} \int_0^\infty \sigma_{t,\lambda} Q_\lambda\ d\lambda,$$

Notice that we have used Eq. 1.

The temperature of the fuel rises because of radiated heat from nearby flames. The source term for the temperature of the fuel is then the fraction of this radiation which is incident upon the fuel. This is similar to a shooting operation from a flame blob to a patch (solid fuel) or to a blob (liquid fuel) (see Eq. 9):

$$S_{T,\text{fuel}} = \rho_{\text{fuel}} \alpha_{\text{fuel}} F_{12} \sigma_{SB} T^4_{\text{flame}},$$

where α_{fuel} is the average absorption cross-section of the fuel. To achieve real-time simulation, we only shoot from a couple of blob-clusters. The loss of temperature due to radiation is similar to the loss term for the temperature of the flame:

$$L_{T,\text{fuel}} = 2\pi \rho_{\text{fuel}} \alpha_{\text{fuel}} \sigma_{SB} T^4,$$

6.1 Implementation

We have implemented the fire model for a fuel map defined on solid objects. This corresponds to non-burning objects coated with a flammable substance. Note that we do not model the change in geometry caused by the burning process. The user specifies the fuel density as a texture map and assigns burning properties (such as the specific heat) to each object. This is analogous to attributing reflection properties to an object in a renderer. To begin a fire simulation, the user metaphorically strikes a match by indicating the origin(s) of combustion, and the simulation commences. The flame model simulation is summarized in the following algorithm.

```
for each time frame do
    for each solid object in the scene do
        Update temperature of the object
        Generate flame blobs and update fuel map
    end
    for all flame blobs do
        Update temperature and density
        Generate smoke blobs
        Move and diffuse
    end
    for all smoke blobs do
        Update density
        Move and diffuse
    end
end
```

The temperature of the fuel is updated using a Crank-Nicholson finite difference scheme, since the domain of the fuel texture map is two-dimensional and there is no advection [16]. Typical resolutions for our simulations were 20×20. The evolution of the temperature and the density of the flame was performed using our blob method. We computed the evolution of the density of the smoke likewise. We allowed an animator to explore the parameter space by mapping the various physical quantities of the model into a graphical interface.

7 Results

We have developed an interactive implementation of the above models. As in [24], the user specifies wind fields and the effects of turbulence and the scale of the wind fields on blobs can be immediately visualized. Various parameters (such as field positioning, scaling, and magnitude) can be modulated in real time.

Figure 4 illustrates a spread simulation. The simulation is synchronized to the passage of real time so that an accurate evolution of the simulation can be subsequently rendered. Figure 5 gives one frame of a fire scene. As described above, a smoke density can also be produced. Rendered smoke can be seen at four stages in Figure 6. Each frame rendered at video resolution took approximately 20 minutes of CPU time on an SGI Indigo 2 with a R4000 processor. This includes rendering smoke and fire, the illumination caused by fire, shadowing and self-shadowing, using roughly 1000 blobs.

Other nuances of our models can be gleaned from different renderings of steam densities. Figure 7 illustrates the effect of multiple scattering at different albedos, with one image having only single scattering performed. A constant phase function was employed, and the images in this figure contain approximately 150 blobs. Solving for the diffuse intensity required one LU-decomposition, which took approximately one second of CPU time. Figure 10 shows four frames from an animation of an observer flying around a backlit cloud with predominantly forward scattering ($\bar{\mu} = 0.75$).

Figure 8(left) depicts the use of unwarped spherical blobs that are allowed to expand uniformly in all directions. At right, we see a much more convincing depiction in which the blobs are themselves warped due to advection. The cost of warping is directly related to solving Equation 7. Ten steps of an Euler integration proved adequate. In practice, then, the rendering time only increased by a factor 10.

Figure 9 illustrates a more “artistic” use of our fire/smoke model.

8 Conclusions and Future Work

We have presented a general model for the representation, animation, and illumination of gaseous phenomena, paying particular attention to a model for fire. While the mathematics characterizing these phenomena is technically complicated, their implementation is efficient, and a nonexpert user can control the evolution of complex gaseous phenomena, such as multiple fires with turbulent smoke and steam swirls.

An issue for further work is *ease* of control over these phenomena. We stated at the outset that the user’s control is largely based on the prior specification of turbulent wind fields. For modelling purposes, this is quite appropriate for interactive simulations involving steam. However, the fire model has a large set of interdependent parameters which are not necessarily easy to manipulate. Thus getting the fire to look “just right” can be problematic.

In many cases, such as in the creation of heat fields, it is quite feasible to generate wind fields dynamically. Here, wind fields can be specialized to a one or more blobs, giving them additional buoyancy, for example.

Also, we would like to model the automatic destruction of objects while burning. We are contemplating the use of more accurate models for the reaction causing the flame, so that we can model explosions more effectively.

Acknowledgements

The financial support of the Natural Sciences and Engineering Research Council of Canada and of the Information Technology Research Centre of Ontario is gratefully acknowledged. The helpful suggestions of the referees are greatly appreciated.

A Diffusion Approximation

The diffusion approximation for diffuse intensity is given by [6]

$$\nabla \kappa_d(\mathbf{x}) \nabla J_d^0(\mathbf{x}) - \alpha_d(\mathbf{x}) J_d^0(\mathbf{x}) + S_d(\mathbf{x}) = 0, \qquad (11)$$

where

$$\begin{aligned} \kappa_d(\mathbf{x}) &= ((1 - \Omega\bar{\mu}/3)\sigma_t \rho(\mathbf{x}))^{-1}, \\ \alpha_d(\mathbf{x}) &= (1 - \Omega)\sigma_t \rho(\mathbf{x}) \text{ and} \\ S_d(\mathbf{x}) &= \sigma_t \rho(\mathbf{x}) \left(J_s^0(\mathbf{x}) - \kappa_d(\mathbf{x}) \nabla \cdot \mathbf{J}_s^1(\mathbf{x}) + Q(\mathbf{x}) \right). \end{aligned}$$

The directional coefficient in the angular expansion is proportional to the gradient of J_d^0:

$$\mathbf{J}_d^1(\mathbf{x}) = \kappa_d(\mathbf{x})(-\nabla J_d^0(\mathbf{x}) + \sigma_t \rho(\mathbf{x}) \mathbf{J}_s^1(\mathbf{x})). \qquad (12)$$

For the exact form of the boundary condition see [23], for example.

References

[1] P. Blasi, B. Le Saec, and C. Schlick. "A Rendering Algorithm for Discrete Volume Density Objects". *Computer Graphics Forum*, 12(3):201–210, 1993.

[2] J. F. Blinn. "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces". *ACM Computer Graphics (SIGGRAPH '82)*, 16(3):21–29, July 1982.

[3] J. D. Buckmaster, editor. *Frontiers in Applied Mathematics. The Mathematics of Combustion*. SIAM, Philadelphia, 1985.

[4] N. Chiba, K. Muraoka, H. Takahashi, and M. Miura. "Two-dimensional Visual Simulation of Flames, Smoke and the Spread of Fire". *The Journal of Visualization and Computer Animation*, 5:37–53, 1994.

[5] J. Chomiak. *Combustion. A Study in Theory, Fact and Application*. Abacus Press/Gordon and Breach Science Publishers, New York, 1990.

[6] J. J. Duderstadt and W. R. Martin. *Transport Theory*. John Wiley and Sons, New York, 1979.

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

[8] M. Inakage. "A Simple Model of Flames". In *Proceedings of Computer Graphics International 89*, pages 71–81. Springer-Verlag, 1989.

[9] A. Ishimaru. *VOLUME 1. Wave Propagation and Scattering in Random Media. Single Scattering and Transport Theory*. Academic Press, New York, 1978.

[10] J. T. Kajiya and B. P. von Herzen. "Ray Tracing Volume Densities". *ACM Computer Graphics (SIGGRAPH '84)*, 18(3):165–174, July 1984.

[11] E. Languénou, K.Bouatouch, and M.Chelle. Global illumination in presence of participating media with general properties. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 69–85, Darmstadt, Germany, June 1994.

[12] M. Levoy. "Efficient Ray Tracing of Volume Data". *ACM Transactions on Computer Graphics*, 9(3):245–261, July 1990.

[13] J. J. Monaghan. "Why Particle Methods Work". *SIAM Journal of Scientific and Statistical Computing*, 3(4):422–433, December 1982.

[14] K. Perlin. "An Image Synthesizer". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):287–296, July 1985.

[15] C. H. Perry and R. W. Picard. "Synthesizing Flames and their Spread". *SIGGRAPH'94 Technical Sketches Notes*, July 1994.

[16] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1988.

[17] W. T. Reeves. "Particle Systems. A Technique for Modeling a Class of Fuzzy Objects". *ACM Computer Graphics (SIGGRAPH '83)*, 17(3):359–376, July 1983.

[18] H. E. Rushmeier and K. E. Torrance. "The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium". *ACM Computer Graphics (SIGGRAPH '87)*, 21(4):293–302, July 1987.

[19] G. Sakas. "Fast Rendering of Arbitrary Distributed Volume Densities". In F. H. Post and W. Barth, editors, *Proceedings of EUROGRAPHICS '90*, pages 519–530. Elsevier Science Publishers B.V. (North-Holland), September 1990.

[20] G. Sakas. "Modeling and Animating Turbulent Gaseous Phenomena Using Spectral Synthesis". *The Visual Computer*, 9:200–212, 1993.

[21] G. Sakas and M. Gerth. "Sampling and Anti-Aliasing of Discrete 3-D Volume Density Textures". In F. H. Post and W. Barth, editors, *Proceedings of EUROGRAPHICS '91*, pages 87–102. Elsevier Science Publishers B.V. (North-Holland), September 1991.

[22] J. Stam. Forthcoming Ph.D. thesis, Department of Computer Science, University of Toronto, 1995.

[23] J. Stam. "Multiple Scattering as a Diffusion Process". In *Proceedings of the 6th Eurographics Workshop on Rendering*, Dublin, Ireland, June 1995.

[24] J. Stam and E. Fiume. "Turbulent Wind Fields for Gaseous Phenomena". In *Proceedings of SIGGRAPH '93*, pages 369–376. Addison-Wesley Publishing Company, August 1993.

[25] J. R. Wallace, K. E. Elmquist, and E. A. Haines. "A Ray Tracing Algorithm for Progressive Radiosity". *ACM Computer Graphics (SIGGRAPH '89)*, 23(3):315–324, July 1989.

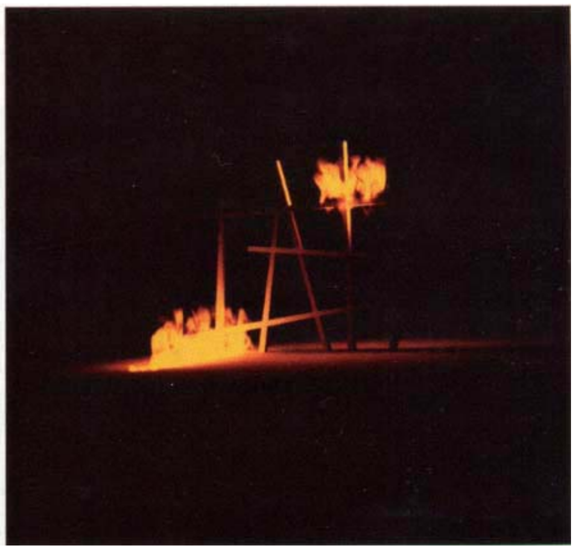


Figure 4: Spread of fire

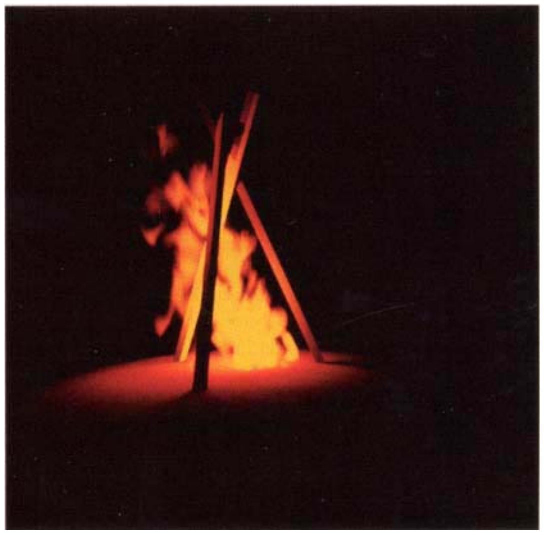


Figure 5: A smokeless rendered fire image.

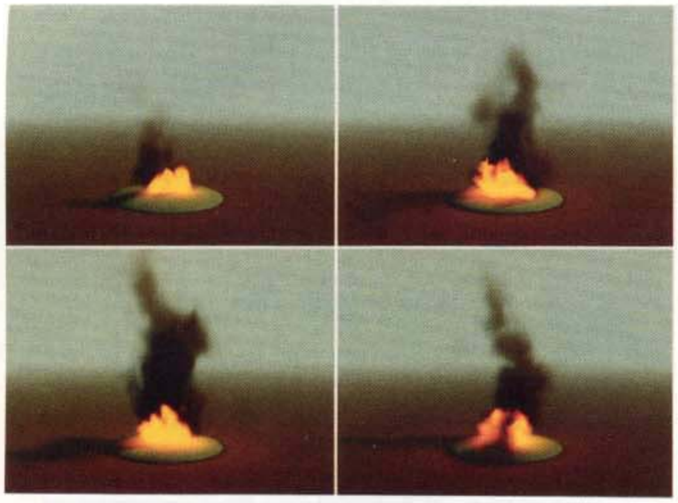


Figure 6: Bonfire with smoke.

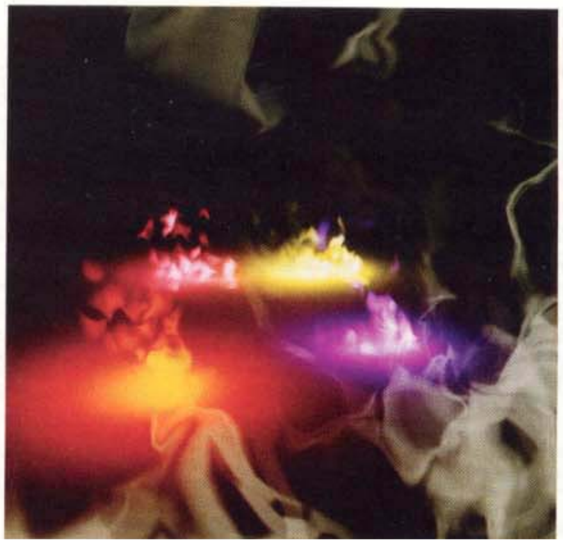
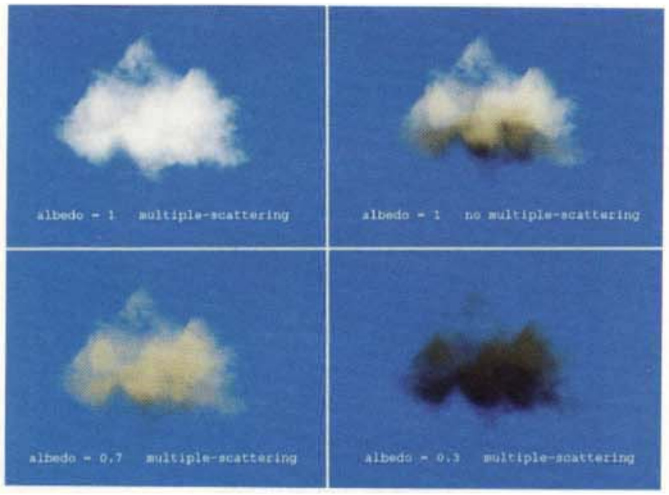


Figure 9: Multi-coloured Fire


albedo = 1   multiple-scattering
albedo = 1   no multiple-scattering
albedo = 0.7   multiple-scattering
albedo = 0.3   multiple-scattering
Figure 7: Effect of multiple/single scattering with varying albedo.

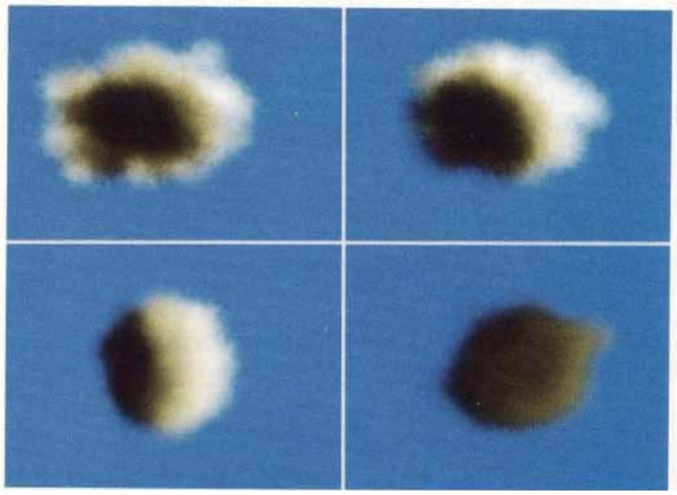


Figure 10: Backlit cloud with anisotropic forward scattering.

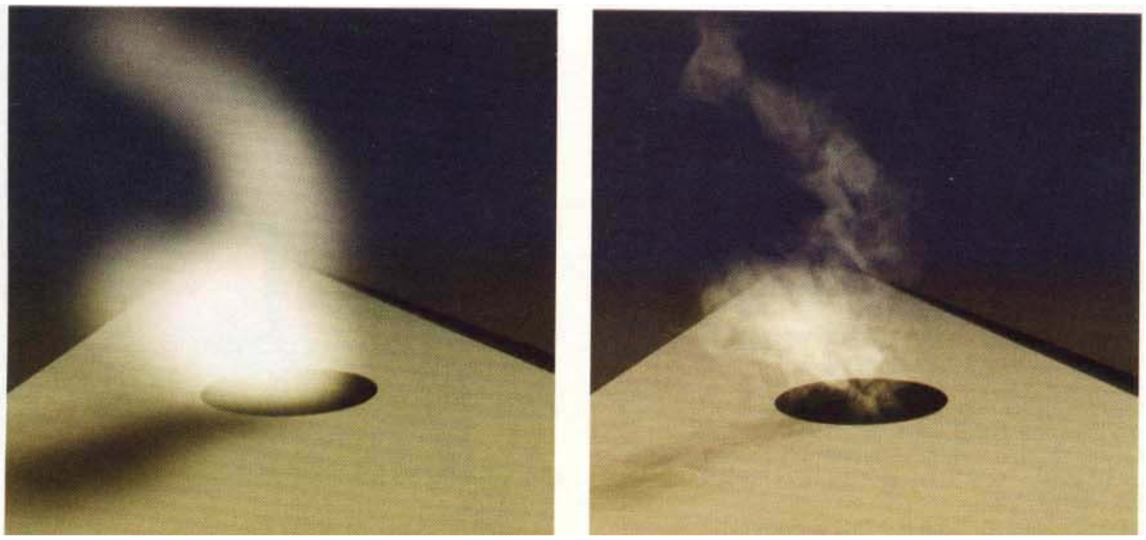


Figure 8: Spherical blobs vs. warped blobs