M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 287
Appears in: Proc. of the Fifth Eurographics Workshop on Anim. and Sim., Oslo, Norway, Sept 1994

Synthesizing Flames and their Spreading

C. H. Perry* **and R. W. Picard**
Perceptual Computing Section, Vision and Modeling Group
MIT Media Laboratory
Room E15-383, 20 Ames Street,
Cambridge MA 02139, U.S.A.
Phone: (617) 253-0384; Fax: (617) 253-8874
{perry, picard}@media.mit.edu

Abstract

We present a new technique for fire synthesis which combines a model for flames with a model for flame spreading. For the flame model, we show that minor changes to traditional particle system techniques exhibit significant improvements in realism, with little to no extra rendering costs. For flame spreading, we model physically-based evolution of the boundary between burning and non-burning regions of polygonally defined objects. Experiments which combine the flame synthesis and spread models are illustrated on polygonally-defined objects. The resulting new technique provides parameters for explicitly controlling wind or gravity, while simultaneously incorporating properties such as the geometry and flammability of the object being burned. Through the use of these parameters, the resulting fire can interact more naturally with the material it is burning, as well as with the surrounding environment.

1 Introduction/Motivation

Fire is one of Nature's greatest actors, able to evoke a wide range of feelings through its emotional and destructive power. As infinite as its repertoire may be, however, it has been extremely difficult to control. Valuable resources are spent trying to exploit the power of fire through pyrotechnic techniques, and ultimately the range of available effects is limited by the laws of physics. Successful computer synthesis of fires is therefore an attractive goal for both the smaller-budget producer and the hyperphysically-creative storyteller.

Synthetic fires need to be realistic and usable. The reigning particle system and noise-based techniques for fire synthesis [15][11] are only reasonably convincing in terms of the rendered flames, and to apply these flames to the burning of an object is a tedious task often involving manual frame-by-frame manipulation. Real fires are dynamic, four-dimensional textures whose flames interact with and adhere to their fuel, preferentially burning exposed corners and edges and lighting the scene around them. Simulated fires should take these interactions into account, and the simulation techniques should provide control over such interactions.

In this paper we present a technique for modeling fires that meets both of these criteria. Enhanced particle system flames and a new, physically-based and environmentally-dependent treatment of spread combine to yield convincing fires without the need for volume rendering or expensive flame modeling processes. Because the spreading takes place on polygonally-defined models, the fires are rendered with the objects, avoiding the usual need to cut and paste flames into the scene[1].

2 Background/Previous Work

Reeves [15] presented the first computer graphics treatment of a fire in the creation of the Genesis Demo sequence for the film Star Trek II: The Wrath of Khan. This sequence involved the ignition and subsequent burning of a dead planet, thus making it a treatment of fire rather than just flames. The fire was modeled with a two-level hierarchy of particle systems, one which controlled the spread over the planet and the other which dictated the form of the individual "explosions." The combination of spreading with a flame model, and the ease and speed of the particle-based technique gave exciting results. Nonetheless, even thousands of particles could not hide the discrete nature of the technique. The flames were unnaturally pointillistic as a result.

Perlin [11] presented a noise-based approach to flame modeling. He formed a fractal approximation to turbulence which he then used to perturb the index into a physically-inspired color/temperature ramp. Traversing through the space of the turbulence function to create upwards motion yielded convincing two-dimensional flames. He extended this work in 1989 [12] to three dimensions, but the already computationally-intensive process became even more burdened by the additional work needed to render volumes. Additional drawbacks to this noise-based, procedural approach become apparent when the fire is to be placed into a larger scene, for external effects such as wind and spread over objects are difficult to treat when the fires are abstracted from the world in this fashion.

Inakage [7] has taken a more physical look at the processes that form real flames. Combining a model for the emission and transmission of light in the regions near combustion with volume rendering techniques, he has rendered convincing examples of both diffusion (candle) and pre-mixed (Bunsen burner) flames. Unfortunately, this level of physically-based volume rendering is expensive and no clear methods for animation or extending the approach to large fires have been presented.

A recent work by Chiba, *et al.* [2] presents a powerful way of modeling vortex-based winds for shaping flames and smoke, as well as an innovative model-as-fuel form of fire spread. Their model is very similar in philosophy to the one presented here (certainly in combining flames and spread), yet important differences exist and will be addressed in Section 3.2.2.

*now at Rhythm and Hues Studios, 910 N. Sycamore Ave., Hollywood, CA 90038

[1] This work is a condensed version of [13].

3 Approach: the "bi-partite" hypothesis

Although *fires* and *flames* have been equated in graphics, there is an understood difference between them in the physical sciences. From [10],

> A *fire* is a set of physical and chemical phenomena, which include [sic] combustion, fluid flows, and pyrolysis or evaporation. When the combustion occurs in the gas phase, the luminous part of the gas is called the *flame*.

As the above definition of fire implies, physically-based fire synthesis should consider not only the luminous, combustive gases but also the processes which transport materials to and from the combustion region (fluid flows and pyrolysis). These aspects of real fires are what lead to non-isotropic spread and a self-supporting burning process.

This paper presents a non-physical flame model and an independent, physically-based fire spread model, and then a method of combining the two to form fires. Although the separation of combustion from flow and evaporation is inherently non-physical, the success of this work is based on the hypothesis that the division will work perceptually. This "bi-partite" hypothesis comes as much from a desire for simplicity (Occam's razor) as from the examples set by the spread modelling community who succeed in accurately modeling spread without treatment of combustion[19] (also [3]).

3.1 Flame Model: an enhanced particle system

The flames are modeled using a modified particle system technique (other attempts of the same include [15], [9], [17], and [2]). Minor adjustments to the shape and color of a rendered particle eliminate the pointillistic problems mentioned above. Additionally, particle systems do not carry the high computational cost associated with volume rendering a lattice.

Flames can be rendered with fewer particles and without the discrete artifacts that have been characteristic of the technique by adding dynamic geometries to the particles. This is a treatment of one of the "infinite degrees of freedom" mentioned in [15] and not an entirely new type of modeling primitive; thus, the term "particle" will still be used to describe the atomic elements of the flames.

3.1.1 Particles with dynamic geometries

Each particle consists of a set of non-overlapping coplanar triangles which all share one common vertex: the world coordinate position of the particle itself (see Figure 1). The vertices of the individual triangles are defined relative to this point and are allowed to change in color (RGBA) and position only. Each vertex other than the center point belongs to exactly two triangles, thus the center is surrounded by triangles without gaps between them. This extra geometry is added to expand the range of rendering options for the particle without a huge increase in complexity.

By shading the triangles and allowing for transparency, the particles can be efficiently rendered as continuous-looking, fuzzy-edged shapes (see Figure 1. Gouraud shading was used in this implementation to keep the shading model *contemporary*, i.e., requiring nothing outside of the current graphics paradigm). In the case of a regular N-gon with many sides, this shape approximates a circle. When non-opaque particles overlap, their colors can be blended in one of many ways to achieve desired results.

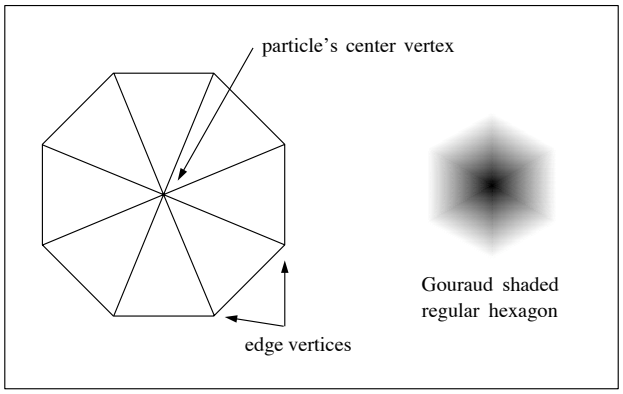

Figure 1: Diagram of a particle with the added geometry. The center and edge vertices can have different RGBA values so Gouraud shading the triangles yields fuzzy-edged structures rather than sharp points.

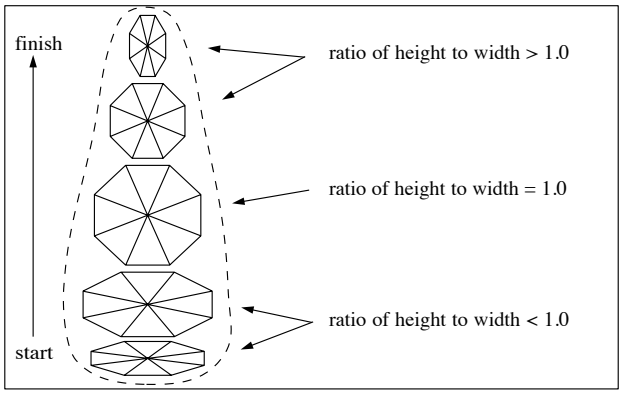

Figure 2: Approximating a candle flame with the rendered trajectory of a single particle. The ratio of the particle's height to its width changes in this example from less than 1 at the start to greater than 1 at the end, yielding the general shape of a candle flame. Note that the RGBA value of the center vertex as well as those of the edge vertices can vary along the trajectory as well.

As with traditional particle properties like position and velocity, the RGBA values and relative positions of the added vertices are allowed to vary with time. Thus the particles can be made to squash and stretch (varying position), change in color (varying RGB), fade in or out (varying transparency), and in general take on arbitrary forms with planar topology.

We hide the two-dimensionality of the particles by always rotating so that they face the viewer before rendering. The particles are therefore reduced from being truly three-dimensional entities to being spherically symmetric three-dimensional spheroids, that is, their rotational degrees of freedom are removed.

3.1.2 Constructing flames with particles: the *long exposure* method

The smallest flame considered in this research is the standard diffusion flame: a candle. This is the atomic element from which larger fires will be built in conjunction with the spreading model (via the bi-partite hypothesis).

We model a single flame as the complete trajectory of a

Figure 3: Candle flames. Each is the 25-step trajectory of a single, 20-triangle particle. The candlesticks were added for effect.

single particle, similar to the way a blade of grass was rendered in [16] (see Figure 2 for a conceptual sketch of this process). The positions and RGBA values of the added geometry are made to vary over time with the position and velocity of the particle, and as a result the open-shutter image of a single particle takes on candle-like shapes and colors (see Figure 3).

The function used to alter the geometries of the particles in Figure 3 is based on examination of the shapes of actual flames from pictures in the literature ([1] [5]), contained on the "Pyromania" CD-ROM of digitized fire footage[8], and from observation. We use a sinusoidal function based on the current "age" of the given particle, where age zero corresponds to the base of the flame. In Figure 3, the maximum age was 24 because the flames were the 25 step trajectories of a single particle.[2] More complex functions could clearly be devised for greater control over the shaping.

3.1.3 Flames in the wind

The examples in Figure 3 show particle system flames with a heretofore unexplained perceptual feature: significant bending in their shape around a generally vertical orientation. This is the result of placing the flames in a wind field[3].

Recall that to render a flame using the long exposure method we start a particle at the desired base of the flame ($\vec{p}(0)$) with a certain velocity ($\vec{v_p}$) and allow it to move through a certain number of steps, accumulating color in the framebuffer while doing so.

Using a (Eulerian) finite difference implementation of Newton's second law we can determine the trajectory of the particle given a force field ($\vec{F} = m\vec{a}$). This force field is the three-dimensional wind field $\vec{u}_\infty(\vec{p}, t)$ in which the flames exist. Note that the wind model presented here is one of many that could be used; there is a range of wind models (such as the innovative technique described in [2]) that could be easily substituted.

In the absence of external winds, candles and other flames burn "upwards," or, more specifically, in the direction opposite gravity. This is because the release of energy in the combustion region forms convection currents: the neighboring air rises after being heated by the flame, and cooler air from the surroundings flows in to replace it. Thus the idea of zero external wind is inappropriate around a flame which exists in both an atmosphere and a gravitational field.

We describe the external wind field $\vec{u}_\infty(\vec{p}, t)$ at a point $\vec{p}$ and time t as the vector sum of two parts, one due to the convection currents and the other independent of them:

$$\vec{u}_\infty(\vec{p}, t) = -k_g\vec{g} + \vec{w}(\vec{p}, t)$$

where $\vec{g}$ is gravity (assumed not to vary with time or position) and $\vec{w}(\vec{p}, t)$ is the convection-independent wind. The coefficient k_g is varied to scale the contribution of $\vec{g}$ to $\vec{u}_\infty(\vec{p}, t)$[4].

It is often helpful to divide $\vec{w}(\vec{p}, t)$ into large-scale and small-scale components, such that global behaviors can be defined with one function and more turbulent, local interactions with another. One example is if you want a generally vortex-shaped wind field to be perturbed by turbulence; it is easy to define the vortex portion as one function and the turbulent disturbances as another. Therefore the expression we use for $\vec{w}(\vec{p}, t)$ is

$$\vec{w}(\vec{p}, t) = \vec{w}_L(\vec{p}, t) + \vec{w}_S(\vec{p}, t)$$

where $\vec{w}_L(\vec{p}, t)$ and $\vec{w}_S(\vec{p}, t)$ define the large-scale and small-scale portions of the wind field, respectively. This paradigm has been used successfully in modeling turbulent wind fields for gaseous phenomena [18], and it represents a separation of local and global forces seen in reaction/diffusion systems [20] and random field models [14], to name a few.

The final expression for the wind and therefore the force field is thus

$$\vec{u}_\infty(\vec{p}, t) = -k_g\vec{g} + \vec{w}_L(\vec{p}, t) + \vec{w}_S(\vec{p}, t).$$

Normalizing the mass of the particles and plugging into $\vec{F} = m\vec{a}$ gives

$$\vec{u}_\infty(\vec{p}, t) = \vec{a}.$$

In terms of the update equation for the velocity $\vec{v_p}$ we have

$$\vec{v_p}(t+1) - \vec{v_p}(t) = \vec{u}_\infty(\vec{p}, t) = -k_g\vec{g} + \vec{w}_L(\vec{p}, t) + \vec{w}_S(\vec{p}, t).$$

If $\vec{w}(\vec{p}, t) = \vec{w}_L(\vec{p}, t) + \vec{w}_S(\vec{p}, t) = 0$, flames will bend only in the direction countering gravity and will exhibit no temporal dynamics. Realistic "flickering" or "wavering" of individual flames through time can be accomplished by giving $\vec{w}(\vec{p}, t)$ nonzero values.

The candle examples in Figure 3 use $\vec{g} = -\hat{j}$, $\vec{w}_L(\vec{p}, t) = 0$, and a small scale wind field

$$\vec{w}_S(\vec{p}, t) = ((\vec{p}(t) - \vec{p}(0)) \odot \hat{j})C\vec{N}(\vec{p}, t) \qquad (1)$$

where $\vec{N}(\vec{p}, t)$ is a function that returns the gradient of a band-limited noisefield as a 3-vector[5], C is a constant[6], and $\odot$ denotes taking the inner product. The coefficient $((\vec{p}(t) - \vec{p}(0)) \odot \hat{j})C$ increases in magnitude with the height of the particle, thus this function perturbs the position of the particle increasingly with distance from the base of the flame. This yields motion analogous to the motion of a flag, being fixed at one end and free to move at the other.

[2] In Figure 3, the vertices of the particle were placed on the unit circle, and they were all scaled by a given factor in each frame. The function used for the scale factor in the x direction (the width) was $\text{xfac} = 0.67 + 0.4 * \sin(\pi * \text{age}/24)$. The scale used for y was linear and increasing, with slope 0.032.

[3] which will be shown to also provide proper motion of flames when the object they are burning on is moved.

[4] in this work, k_g has been equal to 1.0. Note that if $k_g = 0$ and $\vec{w}(\vec{p}, t) = 0$, the flames will not take any shape - equivalent to the zero gravity case.

[5] like Perlin's noise function from [11], thus this flame model is a hybrid of noise-based and particle techniques.

[6] a value that gives pleasing motions is 0.04

$\vec{N}(\vec{p},t)$ is indicated as a function of time and position to represent the standard technique of animating by "moving through the noise field" in time. In this work, $\vec{N}(\vec{p},t)$ corresponds to the following functon:

$$\vec{N}(\vec{p},t) = \vec{N}(\vec{p}(t) - k_s T\hat{j})$$

that is, we move vertically downward through the noise field as simulation time (T) increases, scaled by the "sampling" factor k_s[7]. This gives the illusion of the perturbations moving upwards with the flame, and corresponds physically to disturbances in the convection current moving upwards with the current.

Although the finite difference scheme and use of terms like "position" and "velocity" imply and describe a physically-based model (albeit the physics of projectiles and not flames), the flame model is not meant to be physical and in fact a very non-physical technique was employed at times to keep the particle system flames from looking like sequences of discrete fuzzy blobs. At each time step, the velocity of the particle was normalized in magnitude ($\vec{v_p}(t+1) = \vec{v_p}(t+1)/|\vec{v_p}(t+1)|$), thus keeping the step distance between renderings constant. This form of fixing the magnitude of the velocity was not used for the flames in Figure 3, but it was a required step in the simulations of fires described later where the wind fields became more intense.

For a physical interpretation of these non-physical techniques, consider a linked chain of underwater sausages, or tubular clown-like balloons filled with Helium, held at one end and allowed to swing freely at the other in a wind. They can bend in the wind (or water) and even "double back" but their primary tendency is upward. Even this, however, isn't an exact physical comparison; this portion of the flame model is heuristic by necessity, not physically based, and by no means the only one of its kind that would give decent results.

3.1.4 Light emission

Because of the world-space definition of the particulate flames, a light source can be easily associated with individual flames by making its position $\vec{l}$ be a function of the particle positions $\vec{p}$. The mean of $\vec{p}$ over the frame lifetime of a particle is one such function; others remain to be examined.

3.2 Spreading Model: how fires move over objects

Fires spread because heat is transferred from the burning to the non-burning regions of the combustible fuel, eventually causing them to erupt into flames. The boundary between these regions is called the *surface of fire inception* (or the *inception boundary*) [19], and to properly model spread we must determine how fast this boundary travels into the unburnt regions.

3.2.1 Physics of flame spread

Let q be the heat flux rate (in Joules per unit area per unit time) through the boundary. The spreading rate we are interested in, v_s, is a one-dimensional quantity[8], namely, the speed (in unit distance per unit time) the boundary travels in the direction of its normal (see Figure 4). We can relate q to v_s by considering energy conservation across the boundary. On the burning side, in a time t a total of $qt\Delta A$ Joules travel through an area ΔA of the boundary. On the unburnt side, if the fuel density is ρ and the difference in thermal enthalpy between the fuel at ignition temperature and its original temperature is Δh (in Joules per unit mass), then the amount of energy needed for the region ΔA to move a distance $v_s t$ is $\Delta A v_s t \rho \Delta h$. Equating these two expressions and cancelling terms yields

$$q = \rho v_s \Delta h. \qquad (2)$$

This equation is sometimes referred to as the fundamental equation of fire spread [19]. Informally, it tells us that in order to determine the spread rate v_s, we must know the form of the heat flux rate, q.

The true form of q is complicated and difficult to solve [4]; however, the experimental results from the community studying flame spread can aid in approximating it. They have identified three modes of heat transfer which are dominant among all the contributions to q.

Before describing these modes in detail it is necessary to describe the different regimes of spread. There are two kinds of spread recognized: *concurrent* or wind-assisted spread and *opposed* spread. In the former, the spreading direction v_s has a positive component in the direction of the external wind

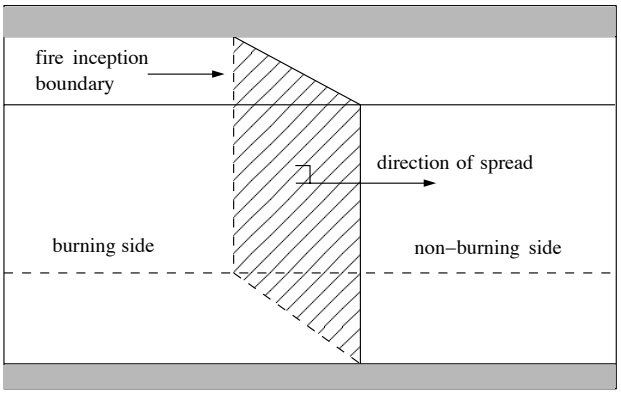

Figure 4: Cross-section through the fuel. The surface of fire inception moves through the unburnt fuel in the direction of its normal. Heat flows through the boundary from the left to the right and determines the spread rate v_s.

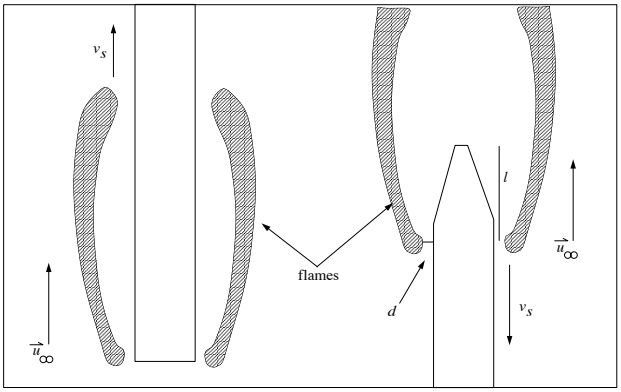

Figure 5: Concurrent and opposed spread. In concurrent spread (left), the flames travel in the same direction as the external wind $\vec{u}_\infty$. In opposed spread, they travel in the opposite direction. Note how in the concurrent case the flames lie close to the unburnt fuel; this is why this kind of spread is faster and more treacherous than opposed spread.

[7] typical values of k_s will be discussed in Section 3.4.

[8] note the difference between $\vec{v_p}$, the vector-valued particle velocity and v_s, the spread control point speed.

(called $\vec{u}_\infty$), while in the latter it is negative (see Figure 5). Gravity induces an upwards wind in the form of convection currents around the flames, so in the absence of any external wind $\vec{u}_\infty$ is purely a function of gravity. A fire started in the middle of a vertical wall with only gravity-induced winds will therefore consist of both types of spread, as the upward evolution will be concurrent and the downward will be opposed.

Fuels are also divided into two categories: thermally thin and thermally thick. Thermally thin fuels are those (like a sheet of paper) whose thickness is such that the temperature gradient across them is negligible, i.e., the temperature is constant through the fuel. Thermally thick fuels, however, do maintain a temperature gradient and the thickness affects the rate of spread.

A given mode of heat transfer has been shown to dominate each of the four categories of spread (these being concurrent and opposed flow on both thermally thin and thick fuels, see [19] and [3]). Since a given fire could contain a combination of these, they will all be described.

Opposed flow over thermally thin fuels

Opposed flow over thermally thin fuels is dominated by heat conduction through the gas. For a sheet of thickness L and width w, the expression Lwq gives the amount of energy per second needed for ignition. The heat is assumed to come from a flame located a normal distance d from the surface of the fuel over an average distance l in the tangential direction upstream from the surface of flame inception. Therefore,

$$Lwq = lw\lambda_g(T_f - T_i)/d$$

is a reasonable approximation of this quantity (λ_g is the thermal conductivity of the gas, T_f is the flame temperature, and T_i is the temperature of the unburnt fuel). The velocities of the gas near the inception boundary need to be small for this mechanism to function, thus l and d will be the same order of magnitude and

$$q_o = \lambda_g(T_f - T_i)/L \tag{3}$$

where the subscript o is to denote opposed flow. This is equation 6 from [19].

Opposed flow over thermally thick fuels

Opposed flow over thermally thick fuels[9] is dominated by heat conduction through the solid. The expression for simple conduction (3) can be used, except with λ_g replaced by λ_s, the thermal conductivity of the solid, and L treated specially for more accuracy[10] [19]. This is included for completeness: In this work we have only considered thermally thin fuels.

Concurrent flow over all fuels

Concurrent flow over fuels of all thickness is dominated by radiation from the flames. The flames stretch in the flowing air currents to lie over unburnt regions of the solid fuel and they radiate heat to its surface. This mode explains the rapid upwards spread we see when we light a match and hold it upside down, for example. An approximate model treats the flames basically as linear in shape with an orientation angle Θ_f relative to the surface and a length h_f. The expression for q_c (concurrent flux rate) is

$$q_c = \epsilon_f \sigma_b T_f^4 h_f \sin\Theta_f / L$$

[9]Polymethylmethacrylate, or PMMA, is the standard thermally thick test fuel used.

[10]For our purposes, the relative difference between concurrent and opposed rates of spread is the key visual feature, making the thin fuel approximation of (3) valid; that is, L is constant.

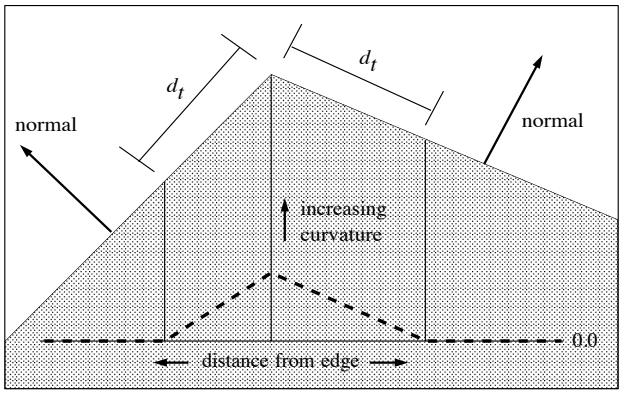

Figure 6: The estimation of surface curvature, shown on a cross-section through a polygonally-defined surface. Curvature is considered zero when further than d_t from an edge, and ramps up to a maximum value (which is a function of the angle between the normals, Θ) as the distance to the edge goes to zero.

where ϵ_f is the emissivity of the flame and σ_b is Boltzmann's constant (see equation 7 in [19]).

For a given fuel, q can be approximated using a combination of these major modes, and v_s can be determined from the fundamental equation of fire spread. We have approximated q by summing the weighted contributions of the opposed and concurrent spread mechanisms, using the expression[11]

$$q = A_o q_o + A_c q_c.$$

The coefficients A_o and A_c depend on the orientation of the spreading direction with respect to the external wind $\vec{u}_\infty$ such that $A_o = 0$ when the flow is concurrent and $A_c = 0$ when the flow is opposed. Since horizontal flow is considered to be the same as opposed flow there are situations where both the concurrent and opposed mechanisms transfer heat to the unburnt fuel and both A_o and A_c are non-zero. The values used in this work are $A_o = 1.0$ (the opposed mechanism always contributes), and A_c is computed as the dot product of the normalized spreading direction vector and the normalized projection of $\vec{u}_\infty$ in the plane of the polygon, with negative values clipped to zero (i.e., the flow is not concurrent).

One factor missing from this approximation to q is the effect of surface curvature on the spread rate. Fire burns preferentially where more oxygen is exposed, and in the case of a burning object this will be on external edges and corners. Likewise, when the amount of oxygen is reduced, flames will spread more slowly. This happens on internal edges where the solid angle of exposed air is less than 2π. The curvature is estimated using the normal of the current polygon and the normal of the adjacent polygon: if the angle Θ between the normals is zero, the curvature is zero.

Because our surfaces are defined by polygons there is local flatness everywhere except when precisely on an edge. To give the illusion that the polygons represent curved surfaces, for points within a predefined threshold distance d_t of an edge the curvature value at the edge is approximated by

[11]for simulations run in this research, T_f, T_i, and L were held constant and the non-varying value used for q_o was 0.09. The value for q_c was 0.13. These values have been scaled for visually agreeable synthesis - consult tabulated values for more physical results.

$1 - d/d_t$ where d is the perpendicular distance to the edge from the current point, and then the curvature is used to scale the velocity[12]. This gives a linear ramp up to the true curvature value for points approaching the edges.

To include curvature into the expression for q, we have scaled q by the coefficient $(1 - d/d_t)\Theta/\pi$. Other functions of the surface curvature may also be used, but this simple approximation gives visually satisfactory behavior with real-time performance.

The fundamental equation (2) gives our final expression for the spread rate:

$$v_s = (1 - d/d_t)\frac{\Theta(A_o q_o + A_c q_c)}{K} \quad (4)$$

where $K = \pi\rho\Delta h$ formally, but represents a factor that sets the rate of the spreading and is best tuned visually[13].

3.2.2 Implementation of spread model

To try and preserve *computability*, we have chosen to represent the inception boundary with individual *spread control points*, such that the polyline connecting the points is an approximation to what would be the true curve. The points are given in counterclockwise orientation such that the burning region is to the left and the nonburning region is to the right of the line. In our implementation, the control points are stored in a linked list.

Note the differences between this implementation of spreading and that in [2]. Their method creates a "fuel" lattice that contains all possible locations where fires can spread. To minimize any spatial aliasing, the number of lattice sites N along a dimension would be quite large. In addition to the extra resources needed to maintain this lattice, the form of conduction they present is $O(N^d)$ where d is the number of dimensions in which the fires spread (two were used in [2]). The technique presented here spreads fire on the polygons already used for modeling, thus requiring little extra resources for efficient, three-dimensional spread.

When begining a spread simulation, a predefined number of control points[14] are placed at the ignition site on a polygon and given evenly spaced radial directions out from that point in the plane of the polygon. Each point evolves through time based on the value of v_s calculated from (4) and the spreading direction at that point. To avoid undersampling the surface, when two points exceed a threshold distance d_s from each other a new point is inserted between them with its own heading[15].

The inception boundary must be confined to move on the surface of an object for the real magic of burning to take place. We have chosen to implement the spreading equations on polygonally defined objects since they are so often used in computer graphics; however, they have not been proven to be the ideal representation for this form of spreading.

We pre-compute the matrices needed to rotate one polygon's normal to the normal of each of its neighbors and store these with each polygon[16]. If a spread control point will cross an edge in a given iteration, its heading vector is rotated using the prestored matrix so that it lies in the plane of the new polygon (see Figure 7). If no polygon shares that edge, the spread point is terminated at the point of intersection with the edge.

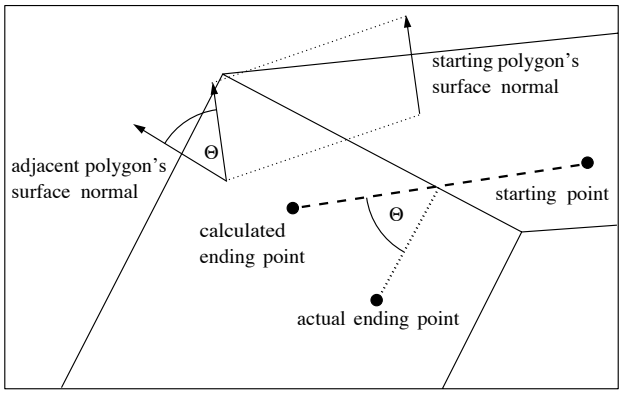

Figure 7: Keeping the spreading on the polygonal model. When the calculated future position of a spread control point lies outside of the current polygon, we split the movement into two steps at the point of intersection with the edge. The portion of the segment outside the polygon is rotated into the neighboring polygon using a pre-stored matrix, forcing the spread to remain on the object.

On topologically non-planar surfaces, we must be careful to prevent the boundary from wrapping around and crossing itself, or we risk burning regions already designated as burnt ("double burning"). This can be accomplished by marking all burnt places, but this is computationally cumbersome as it requires a finer representation (Nature finds it easy to do in parallel). Alternatively, one can test the segment formed by the current and new position of a given control point for intersection with the rest of the boundary, which itself is just a set of line segments. In certain cases where double burning is not an issue, such as with topologically planar surfaces that undergo little to no world-motion, one can eliminate the intersection test and increase the speed of the simulation. Figure 8 shows the segments which must be tested for two arbitrary control points to properly prevent double burning. We have used the three-dimensional segment intersection algorithm described in [6].

The evolution of the spreading boundary can be visualized by drawing a connected line or curve between the control points. This system runs in real time on a 100 MHz SGI Indigo2 Extreme for a reasonable new-point threshold distance d_s.

3.2.3 Other uses for spreading models

It is interesting to note that with a different governing equation for v_s, this model can be used to spread anything over the surfaces of polygonally-defined objects. Spills, for instance, are just the evolution of a spreading boundary under the force of gravity. This force would be relatively easy to combine with capillary forces to model absorption of dyes on fabrics as well.

3.3 Fires: combining flame and spread

Success of the bi-partite hypothesis is based on successful solution of three distinct subproblems, the last of which is being able to combine the first (flame) and the second (spreading) models in such a way as to produce fires.

Simply, we combine the models by treating each spread control point as a potential source of flame particles. As

[12] a typical d_t value used is 0.1 times the average edge length of the object's polygons.

[13] the value used here was $K = \pi/2$.

[14] Eight or more points were used in the experiments here. With too few the discrete nature of the boundary can be seen early in the simulation.

[15] see Section 3.4 for typical values of d_s.

[16] Note that Θ in equation 4 is just the angle of this rotation.

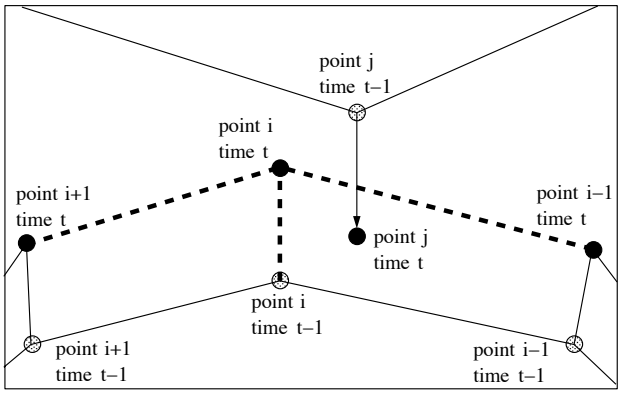

Figure 8: Spreading into regions that have already been burnt can be prevented by testing the boundary for intersections with itself. For point i, the dashed segments in the diagram must be tested for intersection with the segment defined by point j's location at time t and point j's location at time $t-1$ for point j to be allowed to move. Since we must loop through all the control points, we need only test two segments per point, not all three.

the boundary evolves over time, each spread control point records the total linear distance it has traveled and "drops" a new flame particle at its current position after a particular distance has been traveled, resetting its distance counter to zero when this happens. By adjusting the minimum drop distance d_d (less than the linear diameter of a single flame particle, typically), different densities of flames can be achieved. The position of the spread control point becomes the origin of a new flame particle, which will exist for a certain duration based on the flammability of the fuel being burned. Thus, we have independent control over both flame density ρ_f and flame lifetime l_f, making it easy for the model to incorporate changes due to the geometry and flammability of the material on fire.

The flame particles on the surface of the polygonally defined objects are assigned initial velocities $\vec{v_p}$ (not to be confused with spread control point velocities v_s) parallel to the outward facing normal of the polygon they are rooted on. This is done primarily so that the flames lie on the exteriors of objects where they will be shaped in the wind field, but the choice of the normal as the initial particle velocity is otherwise arbitrary.

Once flames are placed on the object and given an initial velocity $\vec{v_p}$ they will appear as they should in the three-dimensional world since the flame model already handles wind fields in world coordinates. If the burning object happens to move in the world, it creates a large-scale wind field in the direction opposite the motion that can be easily realized by setting $\vec{w}_L(\vec{p},t)$ equal to the negative of the motion vector.

Flames in larger-than-candle fires, while they tend to remain distinct, often bend towards and away from other flames and often seem to combine with other flames. This perceived "communication" can be achieved without $O(n^2)$ particle-to-particle interaction calculations by varying the scale of the input $\vec{p}$ to the noise field $\vec{N}(\vec{p},t)$ in equation (1) ($\vec{N}(\vec{p},t) = \vec{N}(k_p\vec{p},t)$). Because of the interpolated nature of the noise, if the positions $\vec{p}$ of particles are scaled by k_p to bring the linear dimension of the conflagrant object to on the order of one lattice site in the noise space, there will be continuous transitions in $\vec{w}_S(\vec{p},t)$ across any flames that are attached to that object. In the limiting case where $k_p = 0$, $\vec{w}_S(\vec{p},t)$ yields flames that vary with time and not with position[17].

Another visual cue: surface charring

The location of an individual flame particle on the surface of a conflagrant object tells us more than just where to place the flame. After the fuel is extinguished, for example, this location is just where one expects to see the blackening effects of char.

The example shown in Figure 9 places a circular, fuzzy, translucent black particle at the flame location once the flame is extinct. This geometry blends upon rendering with the underlying object and gives the illusion of char on the object. If a texture map were used with the object, it could be similarly changed to show char.

Notes on parallelization

Nature controls fires in real time due to massive parallelization of the computations involved; it is interesting to note how both parts of the bi-partite model lend themselves well to parallelization thanks to the computational triviality of updating each individual element. Determination of the value of the noise-based wind field can be done on an individual flame basis, thus each flame could inhabit a processor and compute its own trajectory independent of the others. Spreading calculations could also occur in parallel as each control point moves without knowledge of the others, with a single-neighbor communication required in checking for double burn.

3.4 Controlling the synthesis

This paper introduces a long list of variables and constants that shape the results of the fire synthesis model. The specific values mentioned are guidelines only - the real power of writing *controllability* and *computability* into the model is in exploring these parameter spaces interactively.

3.4.1 Sample parameter values

We have implemented the fire model in C using a mixed-model combination of the SGI graphics library (GL) and X/Motif. Motif was selected because of its ability to easily create and manage value-changing widgets. Table 1 lists the labels, variable names, and value ranges of some of the Motif "scales" (sliders) that were found to be useful in exploring the dynamic range of the model. It is in NO WAY meant to be an exhaustive list.

The ranges of the parameters in Table 1 measured in "distance units" were chosen for polygonal models of a particular size (the edge length of a given polygon in a model was typically about 0.5 units). Because many graphics objects are defined in arbitrarily-scaled "object coordinate systems," one is advised to scale these ranges to the proper typical edge length of a model's polygons.

3.4.2 A typical interaction with the system: playing with fire

Imagine that your goal is to burn a graphical object the size of a building. Typically, you would either start with a polygonal model or you would make one; suppose in this case that the model you choose is a unit cube. You start a fire somewhere on the side of the cube using the default settings for the various model parameters, and you find that these don't quite give the slow, controlled spread you are

[17] typical values for k_p will be discussed in Section 3.4.

Figure 9: Example of charring the surface. Fuzzy black particles are placed on the object surface at the position where each flame particle extinguishes. Note how the texture of the surface can be seen through the translucent char.

Figure 10: An example fire that shows some of the capabilities of the flame model. This fire was the result of spreading over a single triangle, not shown.

imagining. One technique would be to change the physical model parameters of fire temperature, etc., however, it is probably easier to simply reduce the global time scale K.

Once the spread seems to move at a nice rate you begin to focus on the flames themselves. The default settings tend to place very few flames on the cube and you find that your resulting fire looks unfortunately like a bunch of individual candles. Decreasing the drop distance d_d or decreasing the spread control point separation d_s will increase the density of the flames on the surface and you adjust these until the desired effect is achieved.

Now you find that there are plenty of flames, but the ones on the left side of the cube and the ones on the right side tend to move together in a very correlated fashion. This is generally symptomatic of having an inverse flame cohesion k_p that is too low for the desired fire scale. Flames in real fires tend to show more "cohesion" when the fires are small - a burning matchbook, for instance, will not appear to be composed of individual flames. The best solution is to raise k_p until the world coordinates of the points on the cube are scaled agreeably to the inherent scale of the noise field.

The flames now seem to be flickering "too much," that is, you find that there is no perceivable memory between frames - the frame to frame images of a given flame are uncorrelated. Most likely either the flame noisyness C is too high, magnifying so small perturbations in the shape from the noise field, or the flame sampling parameter k_s is too high, causing world space to map to entirely different regions of the noise field each frame. You experiment with reducing both until you get flames you like.

The current parameters now could be used to burn other objects, and you decide to add a garage to your cube-shaped building. Simply add the new polygons to the original model, and the fire behaves accordingly.

This process of diagnosing problems with the synthesis and finding easy solutions among a large parameter space is made possible by the many forms of control and the rapid computation guaranteed with a contemporary model. This kind of interactivity leads to rapid and repeatable solutions to synthesis problems.

4 Examples

4.1 Flames

In addition to the candle flame example of Figure 3, we include two images that showcase the range of the flame model. Figure 10 shows a larger-scale fire that was ignited on a single triangle, and Figure 11 shows a fire attached to a face model.

4.2 Spread over test object

We have created a polygonal test object (Figure 12) that contains both internal (less than 2π steradians) and external (greater than 2π steradians) edge solid angles to show the effects of different surface curvatures on the spread rate.

name	variable	section	low value	high value
global time scale	K	3.2.1	0.004	0.4
drop distance	d_d	3.3	0.01	1.0 (distance units)
spread control point separation	d_s	3.2.2	0.01	0.2 (distance units)
inverse interflame cohesion	k_p	3.3	0.1	10.0
flame noisyness	C	3.1.3	0.0	0.04
flame sampling	k_s	3.1.3	0.0	1.0
curvature distance	d_t	3.2.1	0.01	0.5 (distance units)
flame lifetime	l_f	3.3	1	1000 (frames)

Table 1: This table shows names, corresponding variables and sections in the text, and value ranges for some of the more useful widgets used to interact with the model.

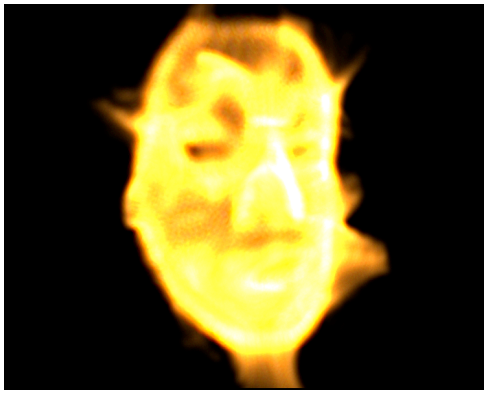

Figure 11: Flames evenly distributed over a 2000+ polygon face model. The model itself was not rendered in order to see if the flames yield enough structure on their own.

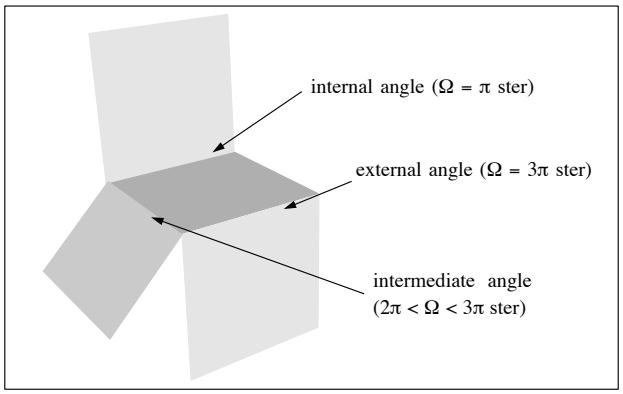

Figure 12: Test object.

A fire is started in the middle of the center polygon of the model and allowed to spread over the surface (see the sequence in Figure 13). The fire does not wrap around to the back sides of the polygons because they are considered to be thermally thin; therefore the process of burning up one side burns both sides.

5 Future Work

Certain effects of burning have not been fully addressed. For example, fuel consumption should be treated with greater detail than just the lifetime of the flames, i.e., the underlying objects may change as they are consumed, and the model needs to reflect this. Additionally, smoke should be released throughout the burning.

Our choice of polygonal spreading is also not final. For example, modeling with superquads or other curved surfaces might make the curvature measure easier and more accurate. These remain topics of investigation by the authors.

6 Conclusion

We have developed a new technique for rendering flames and modeling their spread over polygonally defined objects. By using a physically-motivated model of fire spread and a modified particle system to render flames, we have been able to render realistic flames simultaneously with objects. Unlike previous techniques, this one provides direct control over worldly parameters such as external wind and gravity which influence the spreading. The new technique also allows explicit control over flame density and duration, making it easy to adapt the model to materials with different geometry or flammability. The spreading model developed here also appears to have applications beyond fire, such as for spills and spreading of fluids such as oil, or for dye "bleeding."

Acknowledgements

The authors would like to thank Ali J. Azarbayejani and Michael B. Johnson for assistance and advice throughout the development of this work, not to mention their invaluable help with production (along with John Underkoffler and Michael A. Casey). Thanks also to Erik Brisson and Ken Musgrave for helpful discussions and comments.

A final *thanks* to everyone in the Perceptual Computing Section of the Media Lab for creating and maintaining a wonderful working environment.

References

[1] J. A. Barnard and J. N. Bradley. *Flame and Combustion*. Chapman and Hall, second edition, 1985.

[2] 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.

[3] A. C. Fernandez-Pello. Controlling mechanisms of flame spread. *Combustion Science and Technology*, 32:1–31, 1983.

[4] A. C. Fernandez-Pello. Flame spread modeling. *Combustion Science and Technology*, 39:119–134, 1984.

[5] A. G. Gaydon and H. G. Wolfhard. *Flames: Their Structure, Radiation, and Temperature*. Chapman and Hall, Limited, second edition, 1960.

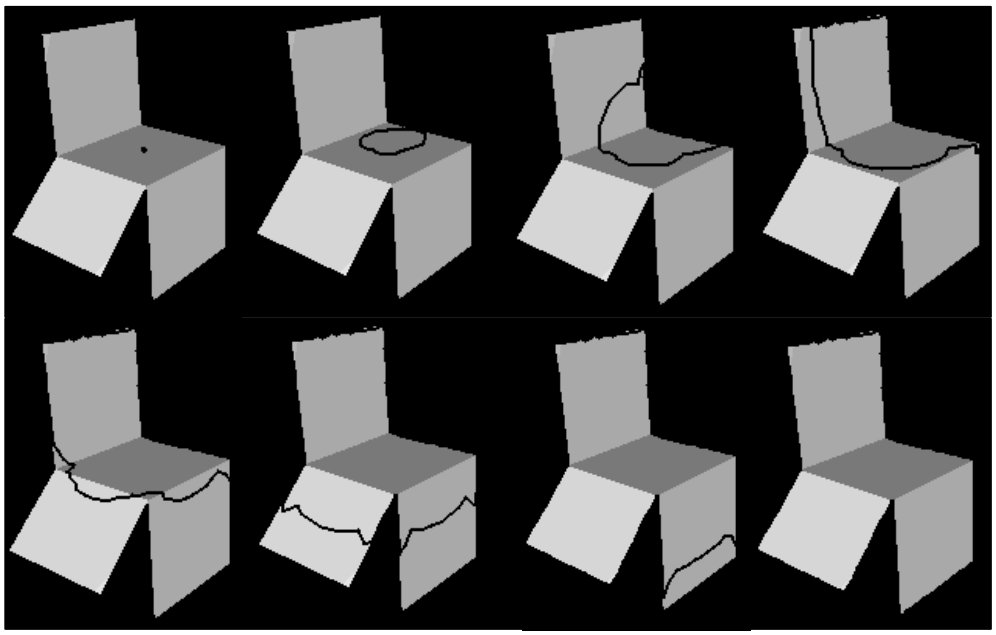

Figure 13: Spreading results on test object. Ignition occurs in the middle of the central polygon and spreads over the surface. Note the rapid upward spread and the preferential burning at the edges.

[6] Andrew S. Glassner, editor. *Graphics Gems*. Academic Press Professional, 1990.

[7] Masa Inakage. A simple model of flames. In *Proceedings of CG International*, pages 71–81. Springer-Verlag, 1990.

[8] VCE Inc. *Pyromania! Playing with fire*. P.O. Box 921226, Sylmar, Ca. 91392-1226, 1993. Movie Effects Studio Volume 1 CD-ROM.

[9] N. Magnenat-Thalmann, D. Thalmann, and S. Beland. The integration of particle and polygon rendering using an A-buffer algorithm. In *Eurographics '86*, pages 161–169, 1986.

[10] Henri E. Mitler. Mathematical modeling of enclosure fires. In E. S. Oran and J. P. Boris, editors, *Numerical Approaches to Combustion Modeling*, chapter 23, pages 183–223. American Institute of Aeronautics and Astronautics, Inc., 1991.

[11] K. H. Perlin. An image synthesizer. *Computer Graphics Proceedings, Annual Conference Series*, 19(3):287–296, 1985.

[12] K. H. Perlin and E. M. Hoffert. Hypertexture. *Computer Graphics Proceedings, Annual Conference Series*, 23(3):253–262, 1989.

[13] Christopher Harton Perry. Synthesizing interactive fires. Master's thesis, Massachusetts Institute of Technology, Media Laboratory, 20 Ames Street., Cambridge, MA 02139, June 1994. Also available as Perceptual Computing Technical Report #277.

[14] Rosalind Picard. Structured patterns from random fields. In *Proceedings of the 26th Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 1992.

[15] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2), April 1983.

[16] W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics Proceedings, Annual Conference Series*, 19(3):313–322, 1985.

[17] K. Sims. Burning letters fire simulation, August 1990.

[18] J. Stam and E. Fiume. Turbulent wind fields for gaseous phenomena. *Computer Graphics Proceedings, Annual Conference Series*, 1993.

[19] F. A. Williams. Mechanisms of fire spread. In *Sixteenth Symposium (Int.) on Combustion, MIT*, pages 1281–1293. Combustion Institute, Pittsburg Penna., 1976.

[20] Andrew Witkin and Michael Kass. Reaction-diffusion textures. *Computer Graphics Proceedings, Annual Conference Series*, 25(4):299–308, 1991.