Voxels on Fire

Ye Zhao Xiaoming Wei Zhe Fan Arie Kaufman Hong Qin *

Center for Visual Computing and Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400

Abstract

We introduce a method for the animation of fire propagation and the burning consumption of objects represented as volumetric data sets. Our method uses a volumetric fire propagation model based on an enhanced distance field. It can simulate the spreading of multiple fire fronts over a specified isosurface without actually having to create that isosurface. The distance field is generated from a specific shell volume that rapidly creates narrow spatial bands around the virtual surface of any given isovalue. The complete distance field is then obtained by propagation from the initial bands. At each step multiple fire fronts can evolve simultaneously on the volumetric object. The flames of the fire are constructed from streams of particles whose movement is regulated by a velocity field generated with the hardware-accelerated Lattice Boltzmann Model (LBM). The LBM provides a physically-based simulation of the air flow around the burning object. The object voxels and the splats associated with the flame particles are rendered in the same pipeline so that the volume data with its external and internal structures can be displayed along with the fire.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: Fire Propagation, Distance Field, Lattice Boltzmann Model, Splatting, GPU Acceleration

Figure 1: Fire on a volumetric table. The underlying noncombustible metal frame is revealed once the wooden outer layer is consumed.

1 Introduction

Simulating fire phenomena is important in many applications such as entertainment, visual simulation, battlefield visualization, and even landscape design. However, the visualization and animation of fire is difficult. Extensive studies [Beaudoin et al. 2001; Chiba et al. 1994; Lee et al. 2001; Nguyen et al. 2002; Perlin 1985; Perlin and Hoffert 1989; Perry and Ricard 1994; Reeves 1983; Stam and Fiume 1995] have been conducted using different approaches to model and render the dynamic behavior of fire. A good survey has been given by Nielson and Madsen [1999]. Over a decade ago, Perlin et al. [1985; 1989] presented a noise-based method to model fire where fractal perturbation is used to simulate its turbulent movements. This approach is easy to implement, however it cannot describe a fire front propagation or external effects such as wind. Reeves [1983] proposed the use of a particle system to model fire. The motion of the fire particles was affected by external forces, such as gravity. Due to the discrete nature of particles, a huge number of them was required to achieve good visual effects. Chiba et al. [1994] combined a user defined vortex-based velocity field and a 2D fuel map to describe the movement of fire. The finite difference solver for partial differential equations was used in both the work of Stam and Fiume [1995] and the work of Foster and Metaxas [1997] to simulate turbulent gas and fire. Qian et al. [1998] used a front tracking method to simulate infinitely thin premixed flame surface, which is explicitly represented by connected marker points. Recently, Nguyen et al. [2002] presented a method based on the Navier-Stokes equations to model fuel with hot gaseous products. Using the level set method to track the moving flame surface they produced realistic looking turbulent flames.

Unlike these studies, we focus in this paper on the modeling of fire front propagation and the burning consumption of objects represented by volumetric data sets, such as the table shows in Figure 1. Splatting is used to render the flames resulting in a realistic visual effect. King et al. [2000] first used textured splats in their work to avoid the computational complexity of large particle systems. In our earlier work [Wei et al. 2002], textured splats were adopted as the basic display primitives and the Lattice Boltzmann Model (LBM) was introduced to model the interaction of the fire

*Email:{yezhao, wxiaomin, fzhe, ari, qin}@cs.sunysb.edu

with wind. We expand our work with LBM to generate the external air velocity field that affects the movement of the fire front.

Related to our work of fire front simulation, both Perry and Picard [1994] and Beaudoin et al. [2001] simulated the spreading of fire on polygon meshes. In their work [Beaudoin et al. 2001], the air velocity field was generated by user defined functions related to environmental properties. Isosurfaces were built from dense grids around the flame skeleton by Marching Cubes, where the density values of grid points were calculated from flame skeletons according to the fire distribution properties. Lee et al. [2001] evolved the geodesics on polyhedral surfaces. They could simulate multiple fire fronts simultaneously as well as fire front merging. In other applications, some different surface propagation methods [Turk 1991; Ruuth et al. 2000] have also been proposed.

All these previous propagation approaches focused on simulating the fire front on surface triangles or polygons. In this paper, we describe a method that simulates the fire evolution by using an enhanced distance field on volumetric data sets. Given a specific density isovalue, we do not generate the isosurface from the volumetric data set. Instead, the fire front propagates directly on a virtual surface related to this isovalue in the volumetruc object. In each step, fire-front points move along a tangent direction of the object surface defined by external forces. Distance field information guarantees that the next position of these points always adheres to the virtual surface. Our distance field is created by a propagation method from a narrow band of space near the isosurface. Neighboring bands defined by different isovalues can be rapidly generated from a shell volume that saves the density relations between every voxel and its neighbors.

Along the fire front, flame particles are emitted according to the fuel value of the current position of the object. Their movement follows the air velocity field which is simultaneously computed with a graphics hardware accelerated LBM. We simply model the object burning comsumption by modifying the fuel property of object voxels and remove the burnt voxels from the rendering. The complicated combustion procedure and the object burning deformation, melting and breaking are left for future work. Splatting is used as our direct volume rendering method for both flame particles and object voxels. The object splats are created according to the user defined transfer functions so that the traditional volume rendering effects can be achieved. These splats are projected onto the screen plane with an opacity blending function to create the final image. In this way, the internal structures of the volumetric object can be rendered together with the fire using the same rendering framework.

Working on volumetric data sets has several advantages. The internal structure of the burning objects can be revealed once the outer layer is burnt and rendered uniformly with the fire flames to achieve high quality results. The consumption of fuel in volumetric objects is more plausible and easy to implement. Furthermore, due to the use of distance field the computation time has no direct relation to the topological complexity of the object. It mainly relates to the volume data size and is pre-calculated. In contrast, the computational time of polygon-based methods depends heavily on the number of triangles or polygons and their topological structures. Moreover, although we currently focus on volumetric data sets, our propagation method can be easily applied to polygonal objects.

The remainder of the paper is organized as follows: In the next section, we introduce the distance field generation. In Section 3, the fire propagation method on the virtual isosurface is described. Flame generation from the LBM is discussed in Section 4. In Sections 5 and 6, we present our splat rendering method and several simulation results.

2 Distance Field

A distance field is a scalar field that specifies a distance to a shape, where the distance is usually signed to distinguish between the inside and outside of the shape. Data set D representing a distance field to surface S is defined as: $D : R^3 \rightarrow R$ and for $p \in R^3$,

$$D(p) = sgn(p) \cdot min\{|p-q| : q \in S\} \quad (1)$$

where $sgn(p) = 1(-1)$ if p is inside (outside) of S, and $||$ is the Euclidean norm. For each voxel in the 3D volume data set grid, the distance to the closest point on the surface is stored. In our fire propagation method, we use an enhanced distance field to save not only the distance value, but also the surface point q in Equation 1 that defines this value. When a fire-front point moves to a position outside or inside the surface, we can easily move it back to its closest point on the surface using the distance field.

Distance fields have been used for morphing [Cohen-Or et al. 1998], virtual endoscopy or skeletal representation [Zhou et al. 1998; Zhou and Toga 1999], digital characters [Perry and Frisken 2001], and other applications. Usually the distance field is calculated from the object modeled as a polygon mesh [Jones 1996]. For the object modeled as a volumetric data set, a mesh may be generated first [Payne and Toga 1992]. Frisken et al. [2000] proposed a hierarchical computation for distance fields. Jones and Satherley [2001] gave a brief review of generation methods and proposed a chamfer distance transform method. Breen et al. [1998] created the distance field from a constructive solid geometry model in a two step process: First they calculated the shortest distances to a set of points within a narrow band around the evaluated surface. Second they used a fast marching method to propagate the shortest distances and closest points information over the whole volume. Our approach uses a propagating method similar to theirs, however, we generate narrow bands for any given isovalue directly via a special shell volume of the volumetric data set. Therefore, the fire fronts can evolve on different isosurfaces chosen by the user.

2.1 Shell Volume

For a traditional volumetric data set V which stores the densities of all the voxels, the shell volume is defined as a data set which has the same grid size as V but stores three special density ranges for every voxel. The basic idea of using the shell volume is that for a given isovalue we can promptly determine whether a voxel of V is inside, outside or on the corresponding isosurface. For this purpose, we save three ranges $(min0, max0)$, $(min1, max1)$ and (min-1, max-1) for each voxel by comparing the density values in its 26-neighborhood. The $(min0, max0)$ range is defined as follows: For each cell C denote d_c as the density of the cell and d_m as the minimum density of its neighbors (if one of its neighbors is out of grid boundary, d_m is set to 0). Then, $min0 = min(d_c, d_m)$ and $max0 = max(d_c, d_m)$. For the $(min1, max1)$ range of C, $min1$ is defined as the smallest $min0$ of its lower density neighbors (i.e., the neighbors of C which have the densities smaller than C). And $max1$ is defined as the largest $max0$ of its lower density neighbors. Finally, for the (min-1, max-1) range of C, min-1 is defined as the smallest $min0$ of its higher density neighbors and max-1 is defined as the largest $max0$ of its higher density neighbors. Straightforward application of the above definitions can produce overlapping ranges. In such cases, we modify either min-1 or $max1$ as needed to ensure that $max1 \leq min0$ and $max0 \leq min$-1.

To illustrate the use of the shell volume, consider a specific isovalue lying in the $(min0, max0)$ range of voxel C. From this we know that the corresponding isosurface lies between C and at least one of its neighbors. Hence C can be considered as lying on the isosurface. Now suppose that the isovalue lies in the $(min1, max1)$ range of voxel C. From this we can conclude that the isosurface

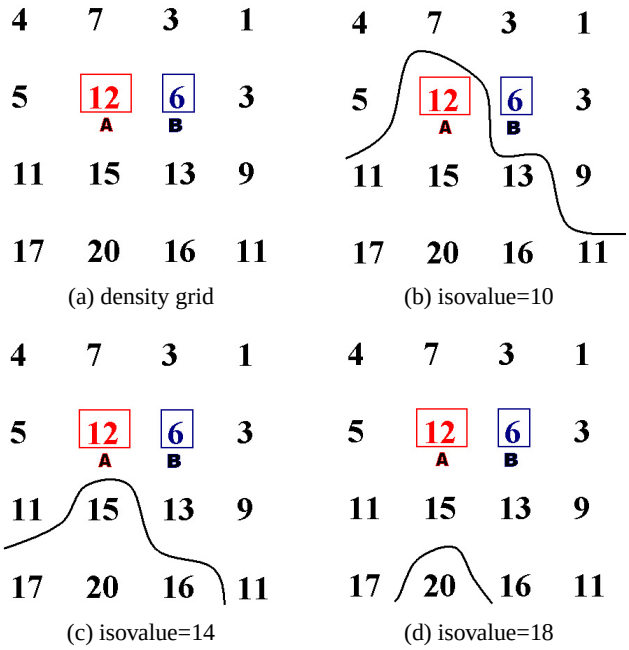


Figure 2: The shell volume is used to quickly and easily generate narrow bands for different isovalues

lies one neighbor away from C. Thus, at least one of C's neighbors can be considered as lying on the surface and C must be inside the isosurface (i.e., on the higher density side). All the voxels like C, lying one neighbor away from and on the high density side of the isosurface, comprise what is termed as the *inside narrow band*. Similarly, all the voxels lying one neighbor away from and on the low density side of the isosurface, comprise what is termed as the *outside narrow band*. And finally, the voxels between those two bands comprise what is termed as the *surface band*.

2.2 Distance Field Generation

Let us denote the inside, outside, and surface narrow bands as *INB*, *ONB*, and *SNB*, respectively. For each voxel p in *INB*, initial distance field information can be generated as follows: Find the neighbors belonging to *SNB* and identify q as the closest one to p. Rather than saving the scalar value $D(p)$ (defined in Equation 1) as traditional methods do, we instead save q which indirectly defines the $D(p)$ for p. The set of all such q defines the *INB* distance field. In the same way, we define the *ONB* distance field. Next, we propagate the computation through the whole volume in a way similar to the fast marching method of Breen et al. [1998]. Note that our distance volume provides an alternate representation of a standard distance field.

Figure 2a shows a 4×4 grid where the values represent the densities. As an example, the voxel A has its $(min0, max0)$, $(min1, max1)$ and (min-1, max-1) ranges as $(3,12)$, $(0,3)$ and $(12,15)$. For the voxel B, these ranges are $(1,6)$, $(0,1)$ and $(6,15)$. While the user chooses the isovalue as 10, A is in the *SNB* and B is in *ONB*. Figure 2b shows the corresponding isosurface. When the isovalue is 14 which lies in the (min-1, max-1) range of both A and B, Figure 2c shows that they are both in *ONB*. In Figure 2d, where the isovalue is 18, both of them do not belong to any narrow bands and their distance fields are calculated from propagation by the fast marching method.

3 Fire Propagation

To simulate fire propagation on a volumetric object, the fire front is represented by an evolving group of front points on a virtual isosurface. These front points can be considered as the emitters of the flames.

3.1 Forwarding Fire Front

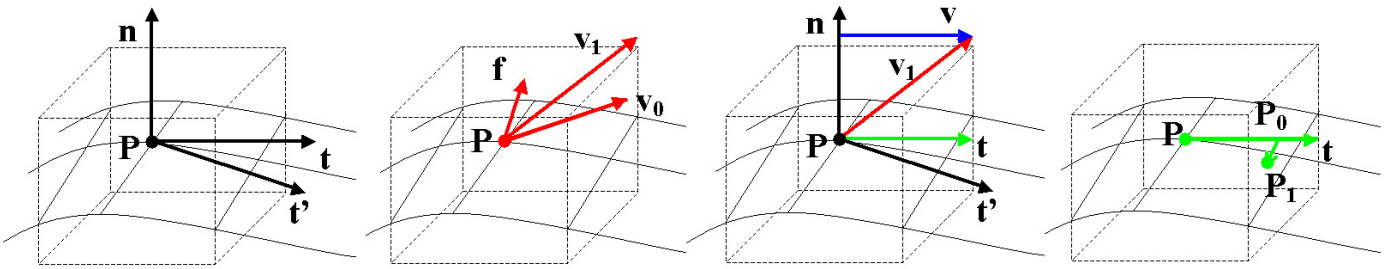
In each step, the fire front evolves on the virtual surface of the object. In our implementation, points which represent the fire front move forward based on the combined effect of their velocities and external forces, such as wind. The wind forces are calculated by LBM and the details will be given in the next section. When simulating point propagation on surface meshes, the points necessarily move across boundaries between different polygons [Beaudoin et al. 2001; Lee et al. 2001]. The polygon geometric properties, such as normal vectors and tangent vectors, are typically used to compute new positions of the points. In contrast, we model the fire propagation on a volumetric data set without generating an isosurface and no geometric entities of the surface are used directly. Nevertheless, the front points always adhere to a virtual isosurface. In order to achieve this, in each time step, every point moves in the following way:

1. Find the tangent plane of the virtual surface at the current position.
2. Calculate the forward velocity from the current velocity and the wind field velocity which represents the external force and is computed by LBM.
3. Adjust the forward velocity within the tangent plane. Compute a temporary target position from this velocity.
4. Define the next position as the closest point on the virtual surface to the temporary target position using the distance field information.

We find the tangent plane at a point by utilizing the property that it is perpendicular to the density gradient at that point. During preprocessing, we compute the gradient vectors using central differences for every regular voxel position of the whole volume data. If during simulation, a point is not located at a regular grid position, we simply use trilinear interpolation to calculate its gradient from its eight regular neighbors.

Figure 3 illustrates the procedure for calculating the next position, P_1, of a fire-front point P. First, the gradient $\mathbf{n}$ of point P is obtained. For this gradient, there are many vectors in the tangent plane perpendicular to $\mathbf{n}$, such as $\mathbf{t}$ and $\mathbf{t}'$ in Figure 3a. Figure 3b shows how we calculate the forward velocity $\mathbf{v_1}$ of P by the vector addition of the current velocity $\mathbf{v_0}$ and velocity $\mathbf{v_f}$ resulting from external forces. In the next step, as shown in Figure 3c, forward velocity $\mathbf{v_1}$ should be projected onto the tangent plane so that the point always moves along the surface. Note that the resulting vector $\mathbf{v}$ lies along the intersection of the tangent plane with the plane defined by $\mathbf{n}$ and $\mathbf{v_1}$. The projection, $\mathbf{v}$, is calculated directly from $\mathbf{v_1}$ and $\mathbf{n}$ as $\mathbf{v} = \mathbf{v_1} - \mathbf{v_1}\cdot\mathbf{n}$. This $\mathbf{v}$ is the tangent vector we want. We just move its start point to P and call it $\mathbf{t}$. Finally, we move P along $\mathbf{t}$ to a temporary target position P_0 by a predefined step size. Although P_0 may not be exactly on the isosurface, from the distance field we can find the closest point P_1 to P_0 which is on the surface, as shown in Figure 3d. Thus, P_1 is the next position of P. We also set the current velocity of P_1 to be in the direction of $\mathbf{t}$ for the next step of the calculation.

Note that one of the innovations of our method is the use of the distance field as a crucial tool for the fire front forwarding. This



(a) (b) (c) (d)

Figure 3: Calculating the next position P_1 of a fire front point P .

method is not limited to volumetric data sets. After computing the distance field of the polygonal objects, it can be straightforwardly applied to the fire front propagation on these objects. In this way, the propagation method does not need to handle complicated polygon surface properties, as previous studies do. We should mention that to achieve more accurate distance values, we need to increase the grid resolution. However, this computation could be finished in pre-computation and will not influence the simulation.

3.2 Adjusting Fire Front

A fire front consists of a group of points. While these points move forward on the virtual surface, the front should be kept orderly and the points should be uniformly sampled so that no self-intersections or large gaps between consecutive points appear. To satisfy this requirement, our method inserts or deletes points when necessary, and also handles self-intersections (the swallow tail problem) after each forwarding step. The similar method was also used by Lee et al. [2001].

If the gap between two consecutive points is too large, the fire front cannot represent the exact surface shape. This is addressed by inserting more points in the following way. We calculate the midpoint of the gap and use the distance field to get the closest point to it on the surface. This point is inserted and its velocity is set to the average of its two neighbors along the front. If, on the other hand, the gap is too small, or even zero, one of the points is deleted.

The swallow tail problem occurs when the fire front intersects with itself. We simply detect its occurrence by checking the directions of one point P relative to its two neighbors P_a and P_b along the front. If the angle between $\overline{P_aP}$ and $\overline{PP_b}$ is too large, there exists a sharp corner at P. To smooth out the swallow tail, we simply delete P. Although this method may occasionally delete some points where no swallow tail happens, it only removes sharp corners and because of our point insertion and deletion procedure, it does not adversely affect the propagation of the front.

Several fire fronts can propagate simultaneously and they may meet each other. When they meet, the area already consumed should not be burnt again. That means one fire front cannot propagate across another. We enforce this by initializing the fuel property of every voxel of the whole object. This fuel value also represents the ability of each voxel to emit different fluxes of flame particles. When a front point moves to a position already consumed, we simply delete it. As a result, two fire fronts may merge and then extinguish.

4 Fire Flames

The fire front emits particles into the air space to form the flames. These particles move according to the wind velocity field surrounding the volume object.

4.1 Wind Field

To model wind and other environmental effects that greatly affect the direction and speed of the propagation of the fire front and flame dynamics, we adopt the LBM approach, instead of using the traditional noise functions [Beaudoin et al. 2001]. In our recent work of Wei et al. [2002; 2003a; 2003b], LBM has been used to model open surface fire, smoke and specific wind field. To achieve interactive speeds, we also implemented the LBM computations on graphics hardware [Li et al. 2003]. The design of the LBM and its simple way of modeling complicated boundary objects make it ideal for our application. In what follows, we briefly review the LBM.

The LBM [Kandhai 1999; Muders 1995] is a numerical scheme for simulating viscous fluids using a regular lattice of cells and links. Motivated by kinetic theory, the population of fluid particles contained in a volume element of fluid is represented by a particle velocity distribution, or packet distribution, at each point in space. Whereas both space and time are discretized, the packet distribution $f_\alpha(\mathbf{x},t)$ is a continuous, real-valued function that specifies the density of particles at position $\mathbf{x}$, at time t with velocity vector $\mathbf{e}_\alpha$. Particle packets stream along lattice links from one site to another in discrete time steps. Between streaming steps, they undergo collision. As in kinetic theory, the collisions conserve mass and momentum (and energy for thermal models). In the limit of zero lattice spacing and time step, the LBM yields the incompressible Navier-Stokes equation.

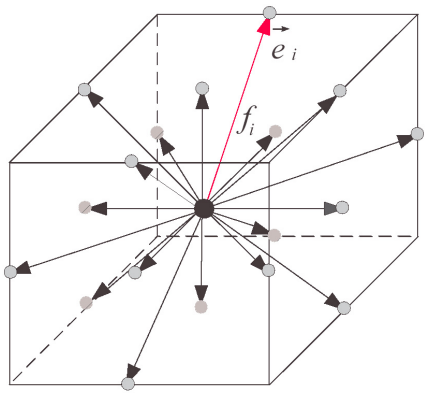
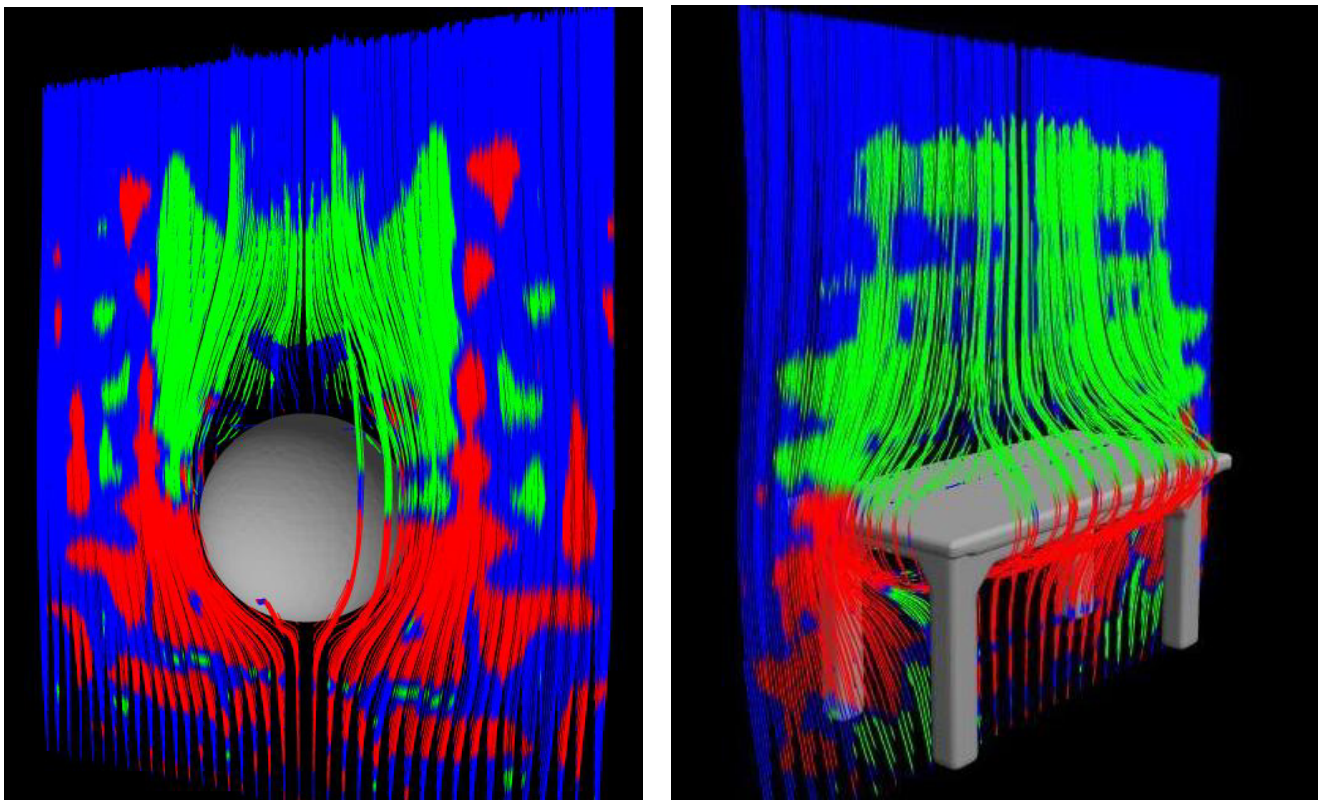


Figure 4: The D3Q19 lattice geometry

To balance between accuracy and speed, we employ the D3Q19 lattice which is a 3D lattice with 19 cells (the center cell with 18 neighboring cells), as shown in Figure 4. On this lattice, the packet



(a) (b)

Figure 5: Streamlines showing the wind field around two object boundaries: (a) sphere; (b) table.

distributions are denoted as f_i where i is a particular link with its velocity vector shown as $\mathbf{e}_i$. From the packet distributions, the macroscopic density (mass) ρ and velocity $\mathbf{u}$ are calculated as follows:

$$\rho = \sum_i f_i \qquad \mathbf{u} = \frac{1}{\rho}\sum_i f_i \mathbf{e}_i \tag{2}$$

where $\mathbf{e}_i$ is the velocity vector, associated with link i. At each time step, every cell updates its packet distribution values based on collision and streaming rules. *Collision* describes the redistribution of microscopic packets at each local node. *Streaming* describes the motion of the packet distribution values to the nearest neighbor along the links in the velocity directions. These two rules of the LBM can be described by the following equations:

$$collision: f_i^{new}(\mathbf{x},t) - f_i(\mathbf{x},t) = \Omega_i \tag{3}$$

$$streaming: f_i(\mathbf{x}+\mathbf{e}_i, t+1) = f_i^{new}(\mathbf{x},t) \tag{4}$$

where Ω is a general collision operator. Since, from a statistical perspective, the effect of collisions is to drive the system toward equilibrium, the BGK model [Kandhai 1999; Muders 1995] is employed to represent the collision operation as relaxation towards local equilibrium. Denoting the equilibrium packet distribution as f_i^{eq}, the BGK collision operator is represented as:

$$f_i(\mathbf{x}+\mathbf{e}_i, t+1) - f_i(\mathbf{x},t) = -\frac{1}{\tau}(f_i(\mathbf{x},t) - f_i^{eq}(\rho,\mathbf{u})) \tag{5}$$

$$f_i^{eq} = \rho(A_i + B_i(\mathbf{e}_i\cdot\mathbf{u}) + C_i(\mathbf{e}_i\cdot\mathbf{u})^2 + D_i(\mathbf{u})^2) \tag{6}$$

where τ is the relaxation time scale, determining the viscosity of the flow, and the coefficients A_i through D_i are constant for a given lattice geometry. The simplest way to define boundary conditions in the LBM is through bounce-back or periodic boundary conditions. However, to define the wind field around a burning object with arbitrary shape, a more accurate boundary condition is needed. In this paper, we adopt the curved boundary condition proposed by Mei et. al [2000]. The surfaces of the burning objects are resampled accurately as the intersection points on the links between fluid nodes and solid nodes. The incoming packet distributions are then calculated explicitly in a way similar to Equation 6. The density ρ and velocity $\mathbf{u}$ of nearby fluid nodes and the boundary velocity $\mathbf{u_w}$ of the intersection points are used in this computation. We have used this approach in our previous work to model the behavior of smoke in an urban canyon [Wei et al. 2003a] and the wind field around a moving object [Wei et al. 2003b]. In this paper we benefit from the simple and efficient way of LBM to generate the dynamic wind velocity field around different objects. In Figure 5, we show the wind field generated around two objects. Streamlines are originating from a plane cutting through the most active flow region of the grid. Red streamlines indicate out-coming flow towards the user, green streamlines indicate incoming flow away from the user and blue streamlines indicate the flow on the plane.

4.2 Flame Generation

A voxel where an emitter exists has a fuel value, that also represents the fuel of the emitter. In each simulation step, flame particles are emitted from the fire-front points with an initial velocity defined by the current wind velocity and the fuel of its emitter. The fuel of the emitters is consumed by emitting different numbers of particles. At the same time, the particles are advected by the LBM velocity field in the air space. Since the particle positions may not lie on the regular grid sites where the LBM velocities are known, trilinear

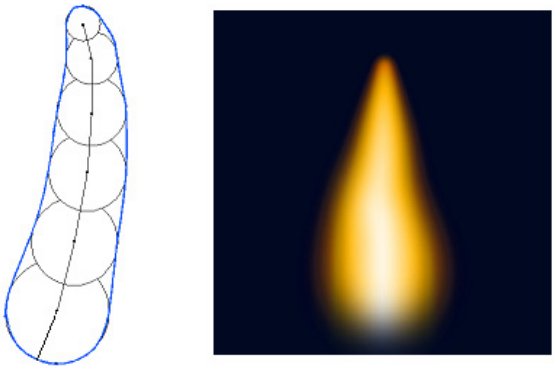
interpolation is used as needed.

The LBM velocity field is calculated as part of the simulation using graphics hardware acceleration, as detailed in our previous work [Li et al. 2003]. We compute both the developing velocity field and the evolving fire as the time goes on.

Flame particles have a finite life span determined by the fuel value of that part of the object from which they are emerging. To model this behavior, when each particle is emitted, it is given a lifetime value proportional to the fuel value of its emitter. The lifetime value subsequently decreases by some factor in every time step. When a particle's lifetime reaches zero, it is deleted and no longer contributes to the rendering. A flame list is created for each emitter on the fire front. This list connects all active particles emitted from a single emitter and represents a flame skeleton.

5 Rendering

In previous studies [Perry and Ricard 1994; Stam and Fiume 1995], fire was typically rendered as a collection of particles with surrounding properties set by the user and objects are rendered as traditional geometric primitives. Beaudoin et al. [2001] introduced well-defined contour effects to build individual flames. They modeled flames using implicit surfaces generated by Marching Cubes from dense grids around flame skeletons, assigned different colors to the resulting surfaces, and achieved good visual effects. Nguyen et al. [2002] rendered the fire as a participating medium with blackbody radiation using a stochastic ray marching algorithm and gave realistic fire results. Our method instead works on the volume data directly, therefore it benefits from direct volume rendering techniques without generating isosurfaces. In this way, external and internal structures of the objects can be rendered together with the fire flames in a uniform framework to achieve superior visual effects.



(a) (b)

Figure 6: Rendering width-decreasing splats with different colors to simulate a fire flame: (a) Decreasing width splats on a fire flame; (b) A fire flame rendered with decreasing width splats with multiple colors.

We use splatting [Westover 1990] to render the voxels and flame particles together. The traditional Gaussian kernel is used to create the splats. Flame particles are added as a different type of splat and these are sorted together with all the voxel splats according to their distance to the screen plane. The sorted splats are projected onto the screen in front-to-back order and the final image is created from their blending. Different opacity values must be assigned to different splats to enable the blending procedure. We assign opacity values to fire particles manually from a table while the opacity values of voxels are assigned by classic transfer functions. For flame splats, small opacity values make the fire translucent while large values hide the objects behind the fire. To keep the realistic shape of each flame, we use Gaussian sphere splats to represent flame particles on one flame skeleton. Because the number of particles emitted in a simulation of one flame is not dense enough to show the continuous shape of the flame, more splats are added just for the rendering. In addition, to model the decreasing width of the flame from the root to the tip, we simply decrease the radius of each particle splat moving from the root to the tip. The color variation caused by different wavelength distributions over the flame depends on the distance from the base of the flame. Our method assigns different colors to different parts of the splats according to their distance from the center to produce a single flame in colored layers. These effects are illustrated in Figure 6. For better visual effects, we use the noise based method, data shader [Brian and Mackerras 1993], to choose the color of the splats in order to synthesize the texture of the volume objects.

6 Results

We have implemented our method and tested it on several volumetric data sets using a 2.53GHz P4 CPU and 1GB rambus memory. Figure 7 shows the simulation of a single fire front on a sphere. As the fire evolves from the bottom to the top, the outside layer of the sphere is consumed by burning and the internal structures are revealed. We render fire particles and voxels together by traditional splatting. The volume objects appearance is controlled by transfer functions easily tuned by the user. Figure 8 simulates two fire fronts that start from two legs of a volumetric table and propagate over the table top. Eventually the two fronts merge together. The other two legs are not consumed because of the wind field direction. The wood layer of the table is burned and the internal noncombustible material is revealed. The computational results and the average simulation rates are shown in Table 1. The fire simulation algorithm

Table 1: Simulation results

	Volume data size	No. of object voxels	No. of flame particles	Propagation simulation time (msec)	Average simulation rate (frames/sec)
Sphere Fig. 7	64×64×64	22872	3745	15	24.6
			11167	31	
			12454	78	
Table Fig. 8	67×128×67	58837	5230	16	14.2
			11096	62	
			25814	156	

which includes propagation and flame particle generation runs at an average rate of 24.6 frames per second for the sphere and 14.2 frames per second for the table. This is an average value since the number of fire emitters and the flame length is dynamically changing during the simulation, which means that the number of particles being emitted is changing. Reducing the number of the flame particles further, the simulation can run much faster while losing some image quality.

We use the graphics hardware accelerated LBM [Li et al. 2003] to calculate the wind field around the object. The grid size for the LBM computation is $60 \times 80 \times 60$ and the speed is about 17 ms per step on an Nvidia GeForce4 Ti 4600 card with 128MB memory, while a comparable CPU computation speed is 384 ms per step.

7 Conclusions

We have presented a new method for simulating the evolution of the fire fronts and the burning of objects represented as volumetric data sets. Our method can be summarized as:

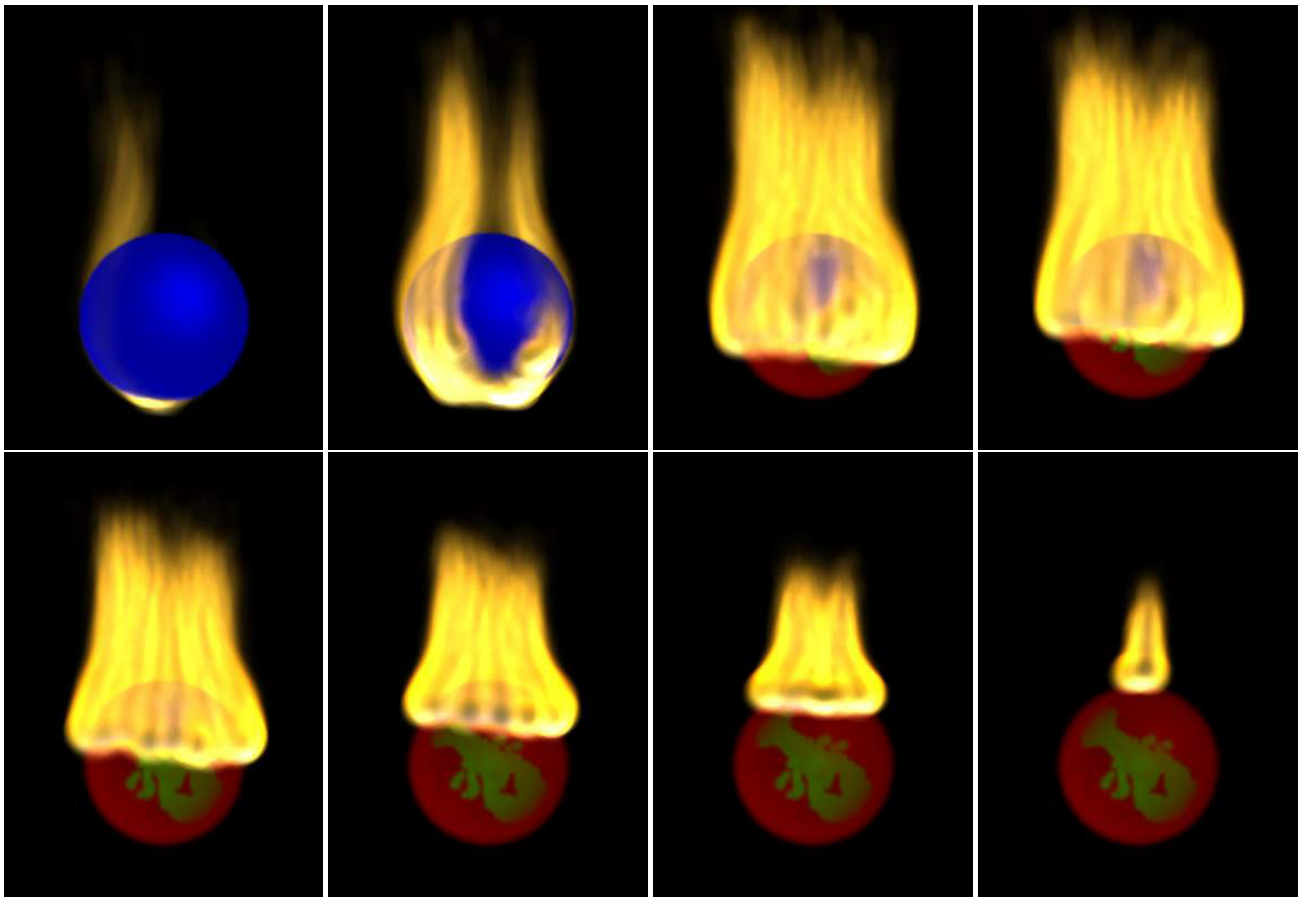


Figure 7: Simulation of one fire front evolving on a volumetric sphere. Once the outside spherical layer is consumed, the internal structures are revealed.

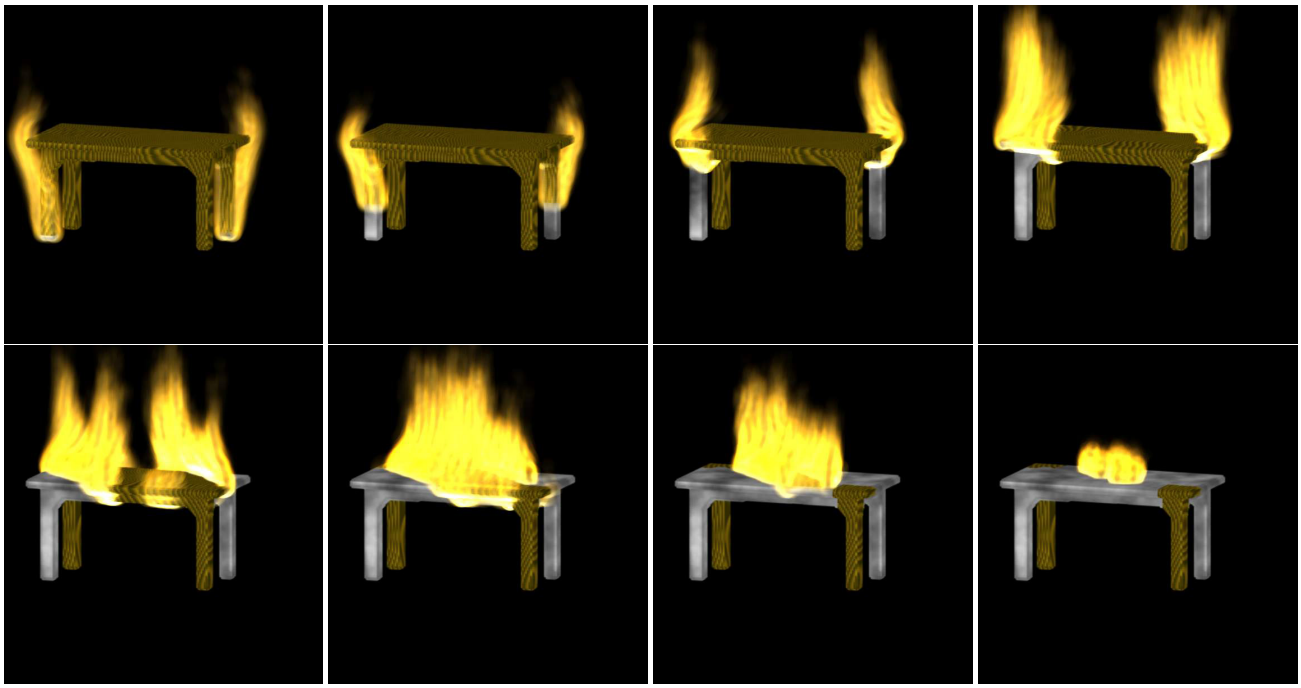


Figure 8: Simulation of two fire fronts evolving on a volumetric table simultaneously. The fire starts from two legs of the table and propagates to the table top. Eventually the two fronts merge together. Once the outside wooden layer is consumed, the noncombustible metal frame is revealed.

- Using enhanced distance field representation, fire-front points are guaranteed to stick to the virtual surface while propagating. This propagation method can be easily applied also to polygonal objects.

- A shell volume is employed to rapidly generate narrow bands, which are used in a fast marching method to create the distance field for the volumetric data for any given isovalue.

- The hardware-accelerated LBM generates a physically-based flow field around complex burning objects at an interactive speed.

- Fire flame particles emitted from the object surface move according to the flow field. They are rendered together with the object voxels by splatting to produce the realistic fire visuals while keeping the volume rendering effects of the object.

Our future work will focus on the fire combustion process and the burning effects on volume objects which are not addressed here. For example, volume objects may deform because of the heat and some objects will even melt or break. We plan to introduce physically-based methods to model such phenomena. At the same time, we will work on the new GeForce FX card which supports floating point computation to accelerate the LBM. The real time simulation will be implemented on higher resolution grids.

Acknowledgement

This work is partially supported by ONR grant N000140110034 and NSF grants IIS-0097646 and CCR-0306438.

References

BEAUDOIN, P., PAQUET, S., AND POULIN, P. 2001. Realistic and controllable fire simulation. *Proceedings of Graphics Interface*, 159–166.

BREEN, D. E., MAUCH, S., AND WHITAKER, R. T. 1998. 3D scan conversion of csg models into distance volumes. *Proceedings of Symposium on Volume Visualization, ACM SIGGRAPH*, 7–14.

BRIAN, C., AND MACKERRAS, P. 1993. Data shaders. *Proceedings of IEEE Visualization 1993*, 25–29.

CHIBA, N., MURAOKA, K., TAKAHASHI, H., AND MIURA, M. 1994. Two-dimensional visual simulation of flames, smoke and the spread of fire. *The Journal of Visulatization and Computer Animation 5*, 37–53.

COHEN-OR, D., LEVIN, D., AND SOLOMOVICI, A. 1998. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics 17*, 2, 116–141.

DOBASHI, Y., KANEDA, K., YAMASHITA, H., OKITA, T., AND NISHITA, T. 2000. A simple, efficient method for realistic animation of clouds. *Proceedings of SIGGRAPH 2000*, 121–128.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. *Proceedings of SIGGRAPH 1997*, 181–188.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. *Proceedings of SIGGRAPH 2000*, 249–254.

JONES, M. W., AND SATHERLEY, R. 2001. Using distance fields for object representation and rendering. *Proceedings of Eurographics*, 37–44.

JONES, M. W. 1996. The production of volume data from triangular meshes using voxelisation. *Computer Graphics Forum 15*, 5, 311–318.

KANDHAI, B. D. 1999. *Large Scale Lattice-Boltzmann Simulations*. PhD thesis, University of Amsterdam.

KING, S. A., CRAWFIS, R. A., AND REID, W. 2000. Fast volume rendering and animation of amorphous phenomena. *Proceedings of Volume Graphics*, 229–242.

LEE, H., KIM, L., MEYER, M., AND DESBRUN, M. 2001. Meshes on fire. *Proceedings of Eurographics Workshop on Computer Animation and Simulation*, 75–84.

LI, W., WEI, X., AND KAUFMAN, A. 2003. Implementing lattice boltzmann computation on graphics hardware. *The Visual Computer (to appear)*.

MEI, R., SHYY, W., YU, D., AND LUO, L. 2000. Lattice boltzman method for 3-d flows with curved boundary. *Journal of Computational Physics 161*, 680–699.

MUDERS, D. 1995. *Three-Dimensianl Parallel Lattice Boltzmann Hydrodynamics Simulations of Turbulent Flows in Interstellar Dark Clouds*. PhD thesis, University at Bonn.

NGUYEN, D. Q., FEDKIW, R., AND JENSEN, H. W. 2002. Physically based modeling and animation of fire. *Proceeding of SIGGRAPH 2002*, 721–728.

NIELSEN, T. E., AND MADSEN, S. T. 1999. *Modeling, Animation, and Visualization of Fire*. Master's thesis, University of Copenhagen, Denmark.

PAYNE, B. A., AND TOGA, A. 1992. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications 12*, 1, 65–71.

PERLIN, K., AND HOFFERT, E. M. 1989. Hypertexture. *Proceedings of SIGGRAPH 1989 20*, 3, 253–262.

PERLIN, K. 1985. An image synthesizer. *Proceedings of SIGGRAPH 1989 19*, 3, 287–296.

PERRY, R. N., AND FRISKEN, S. F. 2001. Kizamu: A system for sculpting digital characters. *Proceedings of SIGGRAPH 2001*, 47–56.

PERRY, C. H., AND RICARD, R. W. 1994. Synthesizing flames and their spreading. *Proceedings of Eurographics Workshop on Animation and Simulation*, 1–14.

QIAN, J., TRYGGVASON, G., AND LAW, C. K. 1998. A front tracking method for the motion of premixed flames. *Journal of Computational Physics 144*, 52–69.

REEVES, W. T. 1983. Particle system - a technique for modeling a class of fuzzy objects. *Proceedings of SIGGRAPH 1983 17*, 3, 359–376.

RUUTH, S., MERRIMAN, B., AND OSHER, S. 2000. A fixed grid method for capturing the motion of self-intersecting wavefronts and related PDEs. *Journal of Computational Physics 163*, 1–21.

STAM, J., AND FIUME, E. 1995. Depicting of fire and other gaseous phenomena using diffusion processes. *Proceedings of SIGGRAPH 1995*, 129–136.

TURK, G. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. *ACM SIGGRAPH Computer Graphics 25*, 4, 289–298.

WEI, X., LI, W., MUELLER, K., AND KAUFMAN, A. 2002. Simulating fire with texture splats. *Proceedings of IEEE Visualization*, 227–237.

WEI, X., LI, W., MUELLER, K., AND KAUFMAN, A. 2003. The lattice boltzmann method for gaseous phenomena. *IEEE Transaction on Visualization and Computer Graphics (to appear)*.

WEI, X., ZHAO, Y., FAN, Z., LI, W., YOAKUM-STOVER, S., AND KAUFMAN, A. 2003. Blowing in the wind. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 75–85.

WESTOVER, L. 1990. Footprint evaluation for volume rendering. *ACM SIGGRAPH Computer Graphics 24*, 4, 367–376.

ZHOU, Y., AND TOGA, A. W. 1999. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics 5*, 3, 196–209.

ZHOU, Y., KAUFMAN, A., AND TOGA, A. W. 1998. Three-dimensional skeleton and ceterline generation based on an approximate minimum distance field. *The Visual Computer 14*, 303–314.