# Simulating Fire With Texture Splats

Xiaoming Wei[1], Wei Li[1], Klaus Mueller[1] and Arie Kaufman[1]
Center For Visual Computing (CVC)
And Department Of Computer Science
State University Of New York At Stony Brook
Stony Brook, NY 11794-4400

## Abstract

We propose the use of textured splats as the basic display primitives for an open surface fire model. The high-detail textures help to achieve a smooth boundary of the fire and gain the small-scale turbulence appearance. We utilize the Lattice Boltzmann Model (LBM) to simulate physically-based equations describing the fire evolution and its interaction with the environment (e.g., obstacles, wind and temperature). The property of fuel and non-burning objects are defined on the lattice of the computation domain. A temperature field is also incorporated to model the generation of smoke from the fire due to incomplete combustion. The linear and local characteristics of the LBM enable us to accelerate the computation with graphics hardware to reach real-time simulation speed, while the texture splat primitives enable interactive rendering frame rates.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

**Keywords:** Fire Modeling, Textured Splatting, Lattice Boltzmann Model, Graphics Hardware

## 1 Introduction

Amorphous phenomena, such as smoke, fire, and gas are essential in many applications, such as in virtual environments, flight simulation, landscape design and film making, to name just a few. However, the animation and visualization of such phenomena is a difficult task. Among the recent and impressive works to model fluid behaviors, the Navier-Stokes (NS) equations have been solved in different ways. To simulate the hot and turbulent behavior of gas, Foster and Metaxas [8] presented a full 3D finite difference solution of the NS equations. Stam [21] further proposed an unconditionally stable implicit solution. Foster et al. [7] implemented a modified semi-Lagrangian scheme to solve the animation of liquid. Fedkiw et al. [6] improved the semi-Lagrangian method by introducing the concept of vortex confinement to the graphics field.

Among all these natural phenomena, fire is believed to be the most difficult to describe, since we do not have a complete understanding of fire and its complexity. Numerous studies have been conducted to describe different aspects of the behavior of fire [1, 3, 12, 17, 18, 19, 20]. Nielsen [16] has given a good survey of the visualization and modeling of fire. Among the early works, Perlin and Hoffert [17, 18] have presented a noise-based method to model 2D and 3D fire, where the turbulence movement is achieved by a fractal perturbation. However, this procedural method constrains the viewpoint position and it cannot describe external effects, such as wind and spread over terrain. Reeves [20] was the first to use particle systems to model fire. In the film *Star Trek II: The Wrath of Khan*, the wall-of-fire element was generated using a two-level hierarchy of particle systems. The top-level system was centered at the impact point of the genesis bomb, generating particles which were themselves particle systems. These second-level particle systems were modeled to resemble explosions where each such particle system acted like a volcano exploding upwardly, eventually falling back to the planet surface due to the pull of gravity, forming a parabolic curve rather than a straight line. Due to the discrete nature of particles, a huge amount of them were required to achieve good results.

To avoid the computational complexity of large particle systems, King et al. [11] have used textured splats to achieve fire animation. These splat primitives are based on simple and local dynamics, and only a small number of them are required for an animation with a sufficient amount of complexity. The high-detail textures not only supply the turbulence appearance of the fire, but also generate a smooth boundary. A drawback of their model is, however, that it lacks the interaction of the fluid with environmental influences, such as wind and temperature, isolating the fire from the rest of the scene.

Chiba et al. [3] have proposed the use of a vortex-based velocity field and a 2D fuel map to describe the spread of fire. Their 2D velocity field is generated randomly. Stam and Fiume [22] have represented the flammable objects with a map indicating the fuel density and temperature at every point on the surface. They then used a numerical finite difference method to simulate the resulting fire. Their method is computationally somewhat expensive. Perry and Picard [19], and recently Beaudoin et al. [1], have simulated the spreading of fire on polygon meshes. They used several connected sample points to represent the boundary of the fire. Lee et al. [12] have extended their work to model both the evolution of fire on complex geometrical structures and the merging of multiple fires. After we completed the current work, we came across Nguyen et al.'s [15] paper, which was concurrently developed. However, while they implemented a full physically-based simulation of fire, our main concern has been to achieve real-time modeling speed.

In this paper, we propose the use of textured splats, instead of particles, as the display primitives in order to achieve real-time performance. However, in contrast to King et al. [11], we also model the interaction of the display primitives with the environment, such as wind and boundary objects. Since our goal is to preserve real-time performance, we cannot use the standard finite difference methods. However, at the same time, we do not want to give up the physically-based behavior either. For this reason, we propose to model fire using a technique known in the fluid dynamics literature as the Lattice Boltzmann Model (LBM), which has not been used before in the fields of computer graphics and visualization. Based on a simple cellular automata framework, the LBM has great

[1]Email:{wxiaomin, liwei, mueller, ari}@cs.sunysb.edu

potential to achieve our goals of real-time and physically-based interaction with the environment. In contrast to the traditional differential equations, the LBM approach considers the problem from the microscopic perspective, using simple, linear and local calculations. These properties make the LBM method very amenable to acceleration on commodity graphics hardware.

In our implementation, we distinguish between macroscopic and microscopic particles. The microscopic particles form the "packets" that propagate across the LBM grid links, while the macroscopic particles form the "display primitives" that move freely about in space. The LBM consists of a discrete lattice, where on every lattice cell, a number of microscopic packet distribution values, representing the density of the microscopic particles, are defined. At each time step, these local microscopic packets relax toward certain values in order to conserve the mass and momentum locally, and their new values propagate to the neighbors along predefined directions. After the LBM packet propagation phase, macroscopic velocity vectors are calculated at each grid node and the resulting velocity field is used to drive the macroscopic particles across space. A physically-based flow simulation results, but at a coarse scale. To add the fine-scale detail for visual plausibility, we associate that macroscopic particle with a texture splat that bears some detail of fire or smoke. An advantage of this framework is that both simulation and rendering of the textured splats can be accelerated on commodity graphics hardware. All put together, we obtain a visual simulation of fire with smoke that is both physically and visually realistic and can occur in real-time.

The remainder of the paper is organized as follows. In the next section, we introduce the basic elements of fire. The textured splats are discussed in Section 3. In Sections 4, 5, 6 and 7, we present the ideas of the LBM, the initial and boundary conditions and the hardware acceleration and its implementation. In Section 8, we demonstrate the generation of smoke from fire by incorporating a temperature field into the model. Finally, we describe several examples in Section 9.

## 2 Fire Theory

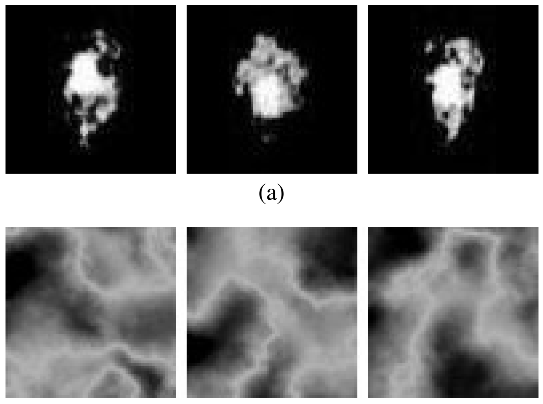
Fire requires three elements to be present: fuel, heat and oxygen.

- Fuel can be in the form of a solid, liquid or gaseous substance. However, in order to burn, all fuels must be chemically decomposed into gases or vapors. This process takes place through the action of heat.

- Heat can be understood as a measure of the molecular activity occurring within an object. The higher the temperature, the faster the action of the molecules. If sufficient heat is applied to an object, the molecules may move so fast that they break away from the bulk of the surface. This is how the fuel is transformed to gas. Heat can be transferred in three ways: convection, conduction and radiation. An intensely burning fire may be sufficient to ignite fuel even from a distance.

- Oxygen contained in air is essential to the combustion. At a certain stage, fire particles may change to smoke particles because of the lack of oxygen in air.

Based on the sum of these conditions, the fire will spread in some places and extinguish in others, due to the existence and consumption of fuel, heat and oxygen. Fire can exist in different forms under different circumstances, ranging from the quietly burning flame of a candle to the roaring fire of a burning oil well. Generating an accurate model of physics that covers all types of fire is too complicated at least for now. In this work, we will concentrate on the modeling of the freely fed fire, for instance, the flickering torch which has no lack of fuel and receives plenty of air. This kind of fire may be advected by all kinds of external forces, which requires the modeling of the interaction between the burning objects and non-burning objects.


(a)

(b)

Figure 1: Two sets of example texture splats generated from real images, used in our work (courtesy of Scott King).

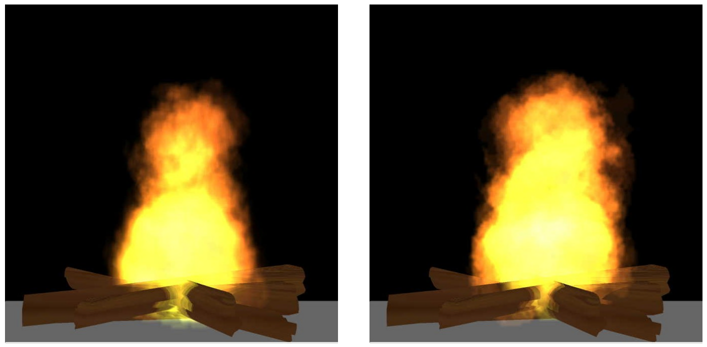

(a) (b)

Figure 2: A simple camp fire using texture splats.

## 3 Texture Splats

In this paper, we associate texture splats [4] with the display primitives for visualizing the fire. A texture splat in this context is a small image of turbulence detail, that is multiplied with a smoothing function, for example a Gaussian, to provide good blending in areas where splats partially overlap. The textures can be generated from real images (see Figure 1 for two sets of example textures), from a noise function [18], or from a high-resolution detailed simulation of fire. The high-detail textures can achieve the appearance of turbulent behavior. In our experiments, around 100 display primitives are sufficient to obtain a visually pleasing result. The image size of the textured splats is $32 \times 32$ pixels. Besides 2D textures, 3D hypertextures [18] could also be employed.

The color and transparency of the display primitives are decided by their location in the fire. Since the temperature rises towards the center of the fire, the radiation from different layers of the fire has a different wavelength distribution. In our work, we use a black body color table to define the colors at different layers. The display primitives at the center of the fire are assigned a brighter color than those located on the periphery of the fire. We further assume that light scattering inside the fire is insignificant and thus implement a single scattering of light toward the viewer [9] in form of the back-

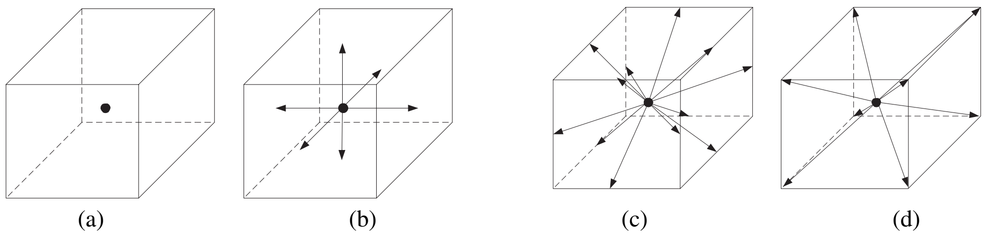


Figure 3: The four sub-lattices defined on a 3D grid. The velocity vectors $e_{qi}$ are shown as arrows in each sub-lattice.

to-front rendering of the $N$ primitives to the screen:

$$E_i = C_i + T_i \cdot E_{i-1} \quad 1 \le i \le N \tag{1}$$

Here, $E_i$ is the light received at the current position, $C_i$ is the amount of light emitted toward the viewer by the display primitive $i$, and $T_i$ is the transparency of the display primitive $i$. This function can be implemented by the blending function of OpenGL, if we consider $C_i$ as the incoming fragment and $E_{i-1}$ as the content of the frame buffer. Figure 2a and 2b show the results of a simple camp fire generated using the two sets of texture splats shown in Figure 1a and 1b respectively. In this example, around 100 display primitives were used. Both results are quite satisfying as long as we keep the turbulent detailed part at the center of the texture. The following is the process of our basic campfire model:

1. Set glBlendFunc(GL_ONE,GL_ONE_MINUS_SRC_ALPHA);
2. Clear frame buffer to black;
3. Inject new display primitives into the system with an initial assigned texture splat;
4. Update the position of the display primitives based on a simple upward velocity;
5. Assign color and opacity for each display primitive based on its current location;
6. Render all textured splats in back-to-front order using splatting [23];
7. Go back to step 2.

As mentioned in the introduction, this simple model works quite well for an individual open surface fire, however, it cannot address the issues of the generation of smoke and the interaction with surrounding environment, such as wind and boundary objects. By defining the conditions of temperature, pressure and wind velocity around a burning object, one could solve the governing differential equations to model the exact movement of the fire. However, this would require a huge amount of computation time. We avoid this expensive computation by considering the fire as a form of hot gas, so the behavior of the fire is mainly decided by the wind direction and the location of the fuel and non-burning boundary objects. In the next section, we describe the use of LBM as the underlying physical model to account for the effect of wind and boundary objects.

## 4 Lattice Boltzmann Model

The LBM [10, 13, 14] is based on Cellular Automata (CA), proposed by John Von Neumann as formal models of self-reproducing organisms. They are discrete dynamic systems, where the space, time, and the state of the cells are all discrete. Each cell in the regular spatial lattice can be in any one of a finite number of states. The states are synchronously updated according to local rules. The CA can produce extremely complex structures from the evolution of rather simple local rules. Dobashi et al. [5] used a special CA model to implement a realistic animation of clouds.

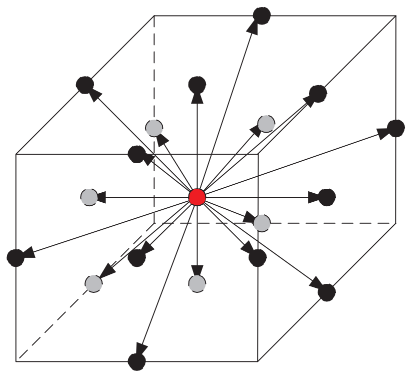


Figure 4: The D3Q19 lattice geometry

Like CA, the LBM is calculated on a 2D or 3D discrete grid, where both the time and the state of each cell are also discrete. Several variables are defined at each cell to indicate whether there are microscopic packets moving in a certain direction. In CA, boolean values are used as these variables, while in the LBM, the averaged packet distributions $f_{qi}$ are used at each cell. They can be of any positive real value, representing the density of the microscopic particle packet. In this way, the simulation result of the LBM does not need to be averaged over a certain grid size, while the averaging has to be done in a CA with boolean variables. The index $qi$ describes the D-dimensional sub-lattice defined by the permutations of a D-tuple of (±1,...,±1,0,...0), where $q$ is the number of non-zero components and $i$ counts the sub-lattice vectors. For a 3D grid, there are four sub-lattices, as shown in Figure 3:

(a) $q$=0, the cell (0,0,0) has particle packets with zero velocities;

(b) $q$=1, the six nearest neighbors $(\pm1,0,0)$,$(0,\pm1,0)$, $(0,0,\pm1)$, where the particle packets move with unit velocity;

(c) $q$=2, the twelve second-nearest neighbors $(\pm1,\pm1,0)$, $(0,\pm1,\pm1)$, $(\pm1,0,\pm1)$, where the particle packets move with a velocity of $\sqrt{2}$; and

(d) $q$=3, the eight third-nearest neighbors $(\pm1,\pm1,\pm1)$ where the particle packets move with a velocity of $\sqrt{3}$.

Three different 3D LBM grids can be formed by combining these sub-lattices. In our current work, we implement the D3Q19 model, which represents a good compromise in terms of computational efficiency and reliability. The D3Q19 lattice consists of three sub-lattices: sub-lattices 0, 1 and 2. Figure 4 shows the D3Q19 model lattice geometry. The velocity directions of the 18 moving packet distributions are shown as arrows. The red center is the packet distribution with zero velocity.

The macroscopic density (mass) $\rho$ and velocity $u$ are calculated from the respective velocity moments of the packet distribution values as follows:

$$\rho = \sum_{qi} f_{qi} \tag{2}$$

$$u = \frac{1}{\rho} \sum_{qi} f_{qi} e_{qi} \tag{3}$$

where $f_{qi}$ is the packet distribution value defined at each cell and $e_{qi}$ is the velocity vector, representing the packet velocity along the lattice link $qi$. Its definition is shown in Figure 3. The physical rules of the LBM grid are designed such that they satisfy the incompressible Navier-Stokes equations globally. That is, the LBM must guarantee the conservation of mass and momentum. Based on this, at each time step, every cell updates its packet distribution values according to two simple and local rules: collision and propagation. Collision describes the redistribution of microscopic packets at each local node. It is decided by the collision operator. No matter how the collision operator is defined, the local conservation of mass and momentum must be satisfied. Propagation means the packet distribution values move to the nearest neighbor along their velocity directions. These two rules of the LBM can be described by the following equations:

$$collision : f_{qi}^{new}(x,t) - f_{qi}(x,t) = \Omega_{qi} \tag{4}$$

$$propagation : f_{qi}(x + e_{qi}, t+1) = f_{qi}^{new}(x,t) \tag{5}$$

where $\Omega$ is a general collision operator. The collisions are completely local, making the LBM efficiently parallelizable. In one time step $t$, each packet distribution value at every node is updated based on the collision operator $\Omega$. Then, in time step $t$+1, the new packet distribution value propagates to the nearest node along the velocity vector $e_{qi}$.

Combining Equations 4 and 5, we get $f_{qi}(x + e_{qi}, t+1) - f_{qi}(x,t) = \Omega_{qi}$. It is critical to select $\Omega_{qi}$ in such a way that the mass and momentum are conserved locally. Based on the work of Chen and Doolean [2], we assume that for each individual packet distribution $f_{qi}$ at each cell, there is always a local equilibrium packet distribution $f_{qi}^{eq}$. Its value only depends on the conserved quantities $\rho$ and $u$ at that cell. In this way, we get a new equation, also called the kinetic equation:

$$f_{qi}(x + e_{qi}, t+1) - f_{qi}(x,t) = -\frac{1}{\tau}(f_{qi}(x,t) - f_{qi}^{eq}(\rho, u)) \tag{6}$$

Table 1: Coefficients of the three sub-lattices in the D3Q19 model

| | Sub-lattice 0 | Sub-lattice 1 | Sub-lattice 2 |
|---|---|---|---|
| $A_q$ | $\frac{1}{3}$ | $\frac{1}{18}$ | $\frac{1}{36}$ |
| $B_q$ | 0 | $\frac{1}{6}$ | $\frac{1}{12}$ |
| $C_q$ | 0 | $\frac{1}{4}$ | $\frac{1}{8}$ |
| $D_q$ | $-\frac{1}{2}$ | $-\frac{1}{12}$ | $-\frac{1}{24}$ |

where $\tau$ is the relaxation time scale and $f_{qi}^{eq}(\rho, u)$ is the equilibrium packet distribution. According to Muders's work [14], the equilibrium packet distribution can be represented by a linear formula:

$$f_{qi}^{eq} = \rho(A_q + B_q(e_{qi} \cdot u) + C_q(e_{qi} \cdot u)^2 + D_q(u)^2) \tag{7}$$

where the coefficients $A_q$ through $D_q$ are dependent on the employed lattice geometry. They are constant values for the specific model. In Table 1, we list the coefficients used in the D3Q19 model. Using these coefficients in conjunction with Equations 6 and 7 ensures local conservation of mass and momentum. Based on the work [10, 13, 14], it can be demonstrated that by observing the above propagation and collision rules, the following macroscopic incompressible Navier-Stokes equations without external forces can be recovered.

$$\nabla \cdot u = 0 \tag{8}$$

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \nu \nabla^2 u \tag{9}$$

The *viscosity* of a fluid is a measure of the fluid resistance to change of shape. For example, water has a higher viscosity value than gas. In LBM, the viscosity is decided by the relaxation time scale $\tau$ with equation $\nu = \frac{1}{3}(\tau - \frac{1}{2})$. Since viscosity is always greater than zero, $\tau$ must be greater than $\frac{1}{2}$.

We now describe our algorithm for calculating the LBM:

1. Set the initial conditions for all grid cells, choose proper density and velocity for inlet cells, and select a relaxation time scale $\tau$. In our current work, $\tau$ is empirically set to be 0.5128 to have a stable simulation;

2. Calculate the macroscopic variables of density and velocity for each cell using Equations 2 and 3;

3. Compute the equilibrium packet distribution for each packet distribution by Equation 7;

4. Plug the packet distribution and equilibrium values into the kinetic equation, Equation 6;

5. Propagate the packet distribution to all neighboring cells;

6. Modify the packet distribution locally to satisfy the boundary conditions;

7. Back to Step 2.

## 5 Initial Conditions and Boundary Objects

Initial conditions are usually specified in terms of macroscopic variables, such as densities and velocities. For the LBM, these macroscopic values are translated into the corresponding microscopic packet distribution values for each node. This is done by solving the equilibrium Equation 7. Hence, the initial values of the density and velocity of each node are plugged into Equation 7, and the equilibrium packet distribution values are set as the initial packet distribution values at each node.

Obstacles in the scene are modeled as boundary conditions. For a grid node near the boundary, some of its neighboring nodes lie outside the computation domain. Therefore, the packet distribution values at these nodes are not uniquely defined. In Figure 5, we show a concave edge boundary node of the D3Q19 model with nine unknown packet distribution values. Different types of boundary conditions have been introduced in the field of Hydrodynamics

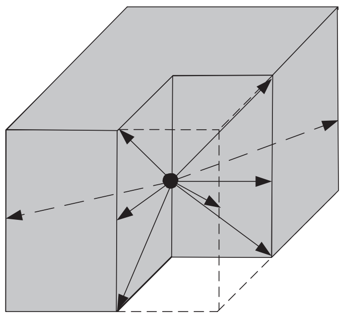


Figure 5: Concave edge boundary node with nine unknown values (some invisible links shown by dashed arrow lines).

for the LBM. In this paper, we implement the simple bounce-back method, where boundary nodes are placed half-way between the grid points. When packets propagate to the boundary nodes, they just bounce back along the link. The propagation step is changed to $f_{-qi}(x,t+1) = f_{qi}(x,t)$. This approach is fast, easy to implement and it can handle complicated boundaries.

For the boundary condition of an outlet, one way to do in the LBM is to assign a density value for the outflow. However, we have found that it is difficult to define the outflow density before the simulation. We instead impose a zero derivative condition after the collision step, which works very well. Suppose the surface Z=$N_z$ is an outlet (where $N_z$ is the number of lattice cells in the Z-direction), for each of the outlet nodes, we execute the equation: $f_{qi}(i,j,N_z - 1) = f_{qi}(i,j,N_z)$

## 6 Hardware Acceleration

The local nature of the LBM calculations lends itself well to an implementation on commodity graphics hardware for additional speedup. The LBM Equations 2, 3, 6 and 7 are mainly composed of simple operations such as addition, subtraction and multiplication, that are naturally available on the rasterization stage of the graphics hardware. Division and other more complicated operations are implemented with lookup tables.

First, we divide the Lattice Boltzmann model and group the packet distributions $f_{qi}$ into arrays according to their velocity directions. All the packet distributions with the same velocity direction are grouped into the same array, while keeping the neighboring relationship of the original model. We then store the arrays as 2D textures. For a 2D model, all such arrays are naturally 2D, while for a 3D model, each array forms a volume and is stored as a stack of 2D textures. The idea of 2D textures stacks is from 2D texture-based volume rendering, but note that we don't need three replicated copies of the dataset (one for each major direction). The density $\rho$ and velocity $u$ are then computed by summing the $f_{qi}$ from all directions. Next, the $f_{qi}$ are updated according to Equations 6 and 7 with similar hardware configurations. The new distribution textures are generated and replace the old ones. Finally, the newly created $f_{qi}$ propagates to the neighboring grid at every time step. We decompose the velocity into two parts, the velocity component within the slice and the velocity component orthogonal to the slice. The propagation is done for the two velocity components independently. The in-slice propagation is achieved simply by translating the distribution textures and the propagation along the direction of the orthogonal velocity is done by renaming the textures.

To reduce the overhead of switching between textures, multiple textures representing packet distributions with the same velocity direction are stitched into one larger texture. For the initial and boundary conditions, the packet distribution values should be handled differently. A general approach is to compute the new distribution values for these nodes, then set the new values into the distribution textures. The computation of these new packet distributions can be done with either CPU or graphics hardware.

A major concern about using graphics hardware for general computation is the accuracy. Most graphics hardware supports only 8 bits per color channel. There have been a few limited supports of 16-bit textures but these are too restrictive to be capable of relatively complicated application such as the LBM simulation. Fortunately, the variables of our current LBM simulation fall into a small numerical range which makes the use of range scaling quite effective. The hardware acceleration gives us a speedup factor of over 50. (See [13] for detailed description of the hardware implementation.)

## 7 Implementation

Our simulation process consists of two parts. First, considering that the fire can only burn in the form of gas, we initialize the LBM grid with the velocity of the fire and the boundary conditions, and generate a 3D velocity volume. Then, the display primitives, later rendered as texture splats, are injected into the scene and are advected by this 3D velocity volume. We assume that each display primitive has a certain amount of fuel. After a few time steps, the display primitives are removed from the system as the fuel they carry is consumed by the combustion. Initially, the inlets are defined explicitly. During the simulation, the flammable objects can catch fire and new inlets are defined. The power of our method is that we distinguish between the microscopic packets in the LBM and the macroscopic display primitives that we can observe. Thus, we do not require a huge amount of display primitives, which allows the rendering to be fast. Our fire simulation system works in the following way:

1. Inject new fire particles (display primitives) into the system with an initial texture (the number of particles characterizes the density of the fire);

2. Update the velocity vector on the grid points according to the LBM algorithm in Section 4. We implement this part in texture hardware to attain real-time interaction speed;

3. Compute the velocity vector at the current position by trilinear interpolation for each fire particle, and move it to its new location. If a fire particle moves out of the grid, remove it from the system. If a fire particle consumes all of the fuel it carries, it is also removed from the system;

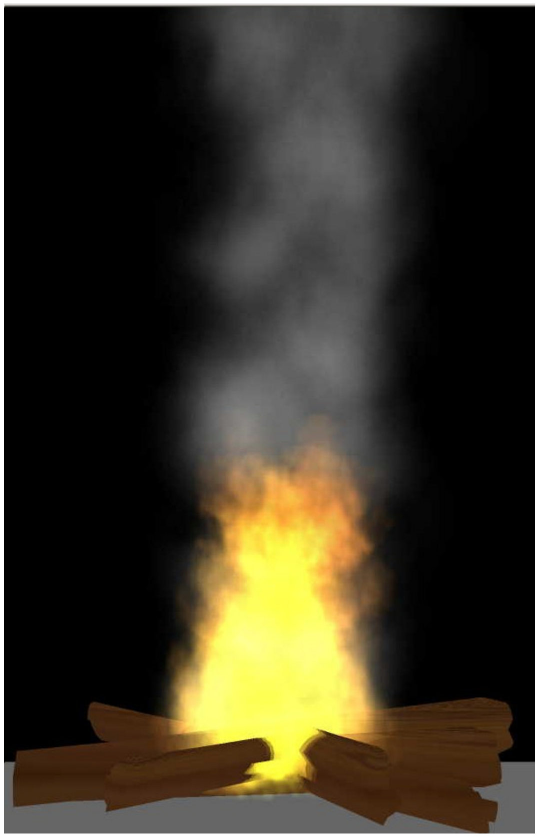
4. Assign the color and transparency for each fire particle;

5. Render all fire particles in a back-to-front order using texture splatting. Since fire emits light, to account for the effect of fire on other objects, we first render the fire particles separately to construct a light image, then use projective textures to map the image onto surrounding objects.

6. Go back to step 1.

## 8 Smoke

Smoke generally refers to a visible mixture of products given off by an incomplete combustion of a substance such as wood, coal and fuel oil. This airborne mixture usually contains small particles of carbon, ash and the like, as well as vapor, such as carbon dioxide and water vapor. The generation of smoke from fire can be modeled by ways of a temperature field. The exchange of fire energy in air


(a)

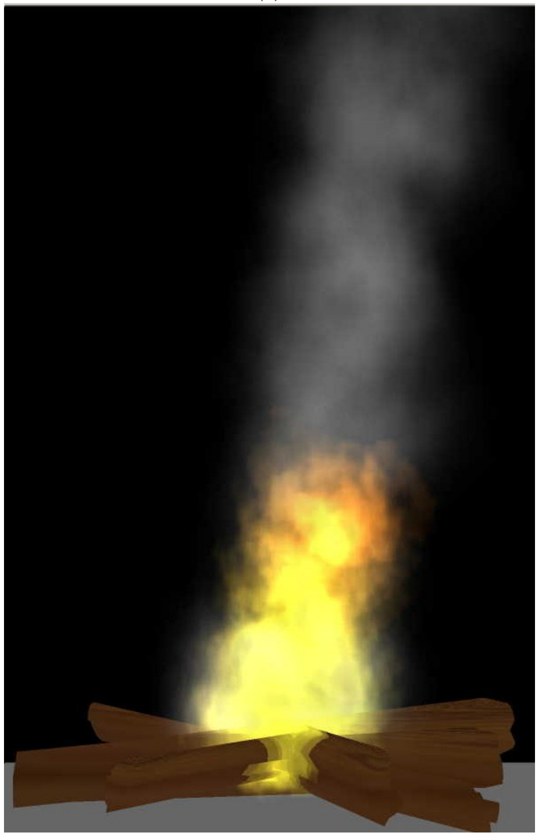

(b)

Figure 6: A camp fire with smoke

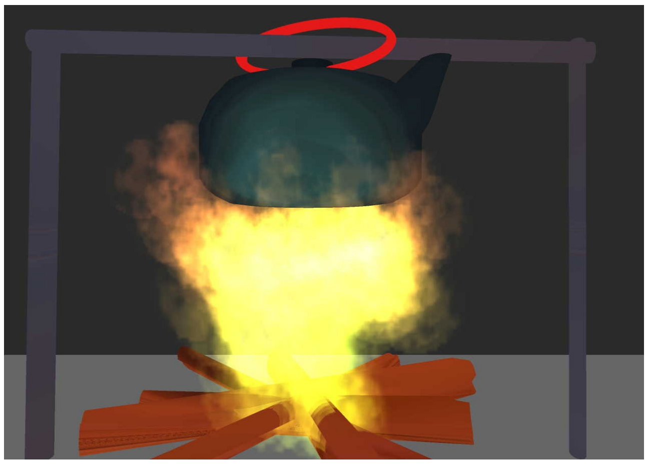


Figure 7: A kettle on a campfire

can be characterized as a combination of the convection and diffusion of heat in neighboring cells. To achieve fast speed, we use a linear equation, similar to Chiba et al.'s [3], instead of a more accurate differential equation, to approximate the change of temperature for the display primitives. Our model is governed by the following heat formula:

$$T_k(t) = \alpha T_k(t-1) + \beta \sum_{j \neq k} G(d_{kj}) T_j(t-1) \qquad (10)$$

where $\alpha$ is the conservation coefficient; $\beta$ is the transferability coefficient; $d_{kj}$ is the distance between the display primitives $k$ and $j$; and $G()$ is a function describing the thermal diffusion. We use a Gaussian filter as the function $G()$ to approximate the effect of diffusion. At $t$=0, $T_k(0)$ is a predefined initial value, indicating the initial temperature. As the temperature of a display primitive decreases to a certain point, the fire particle changes to a smoke particle. The texture on it is kept unchanged. A separate color table is used to assign the color for smoke. The computation for smoke is done on CPU.

## 9 Examples

In this section, we show a few examples of fire generated using the LBM and the textured splats. Our work has been implemented on a P4 1.6GHz PC with a Nvidia GeForce3 Ti 200 card that has 64MB of memory. Figure 6 shows two images of the same camp fire model in Figure 2, with the addition of smoke. In Figure 2b, we set a wind boundary condition on the left side of the grid. Figure 7 indicates the interaction of fire with the surrounding boundary objects. A kettle is placed above a campfire. We initialize a $4 \times 4$ patch as the inlet of the LBM grid with a velocity of 0.2. The kettle is placed as a boundary box into the grid.

Figure 8 is an image sequence showing the change of the fire boundary as we move a torch. Initially, the torch is stationary, as shown in Figure 8a. As we move it to the left, the computation LBM grid moves together with the base of the torch and we set the left side of the grid to a velocity of 0.1, simulating the moving speed. This boundary condition is kept unchanged throughout the moving process. As the torch stops, the speed of the boundary condition is gradually reduced to 0. The shape of the fire becomes stable again, as shown in Figure 8d. In all these examples, our grid size is $32 \times 32 \times 32$. The time used for one step simulation without using graphics hardware is about 180ms. With graphics hardware, the computation time for one simulation step is about 4.6ms. For

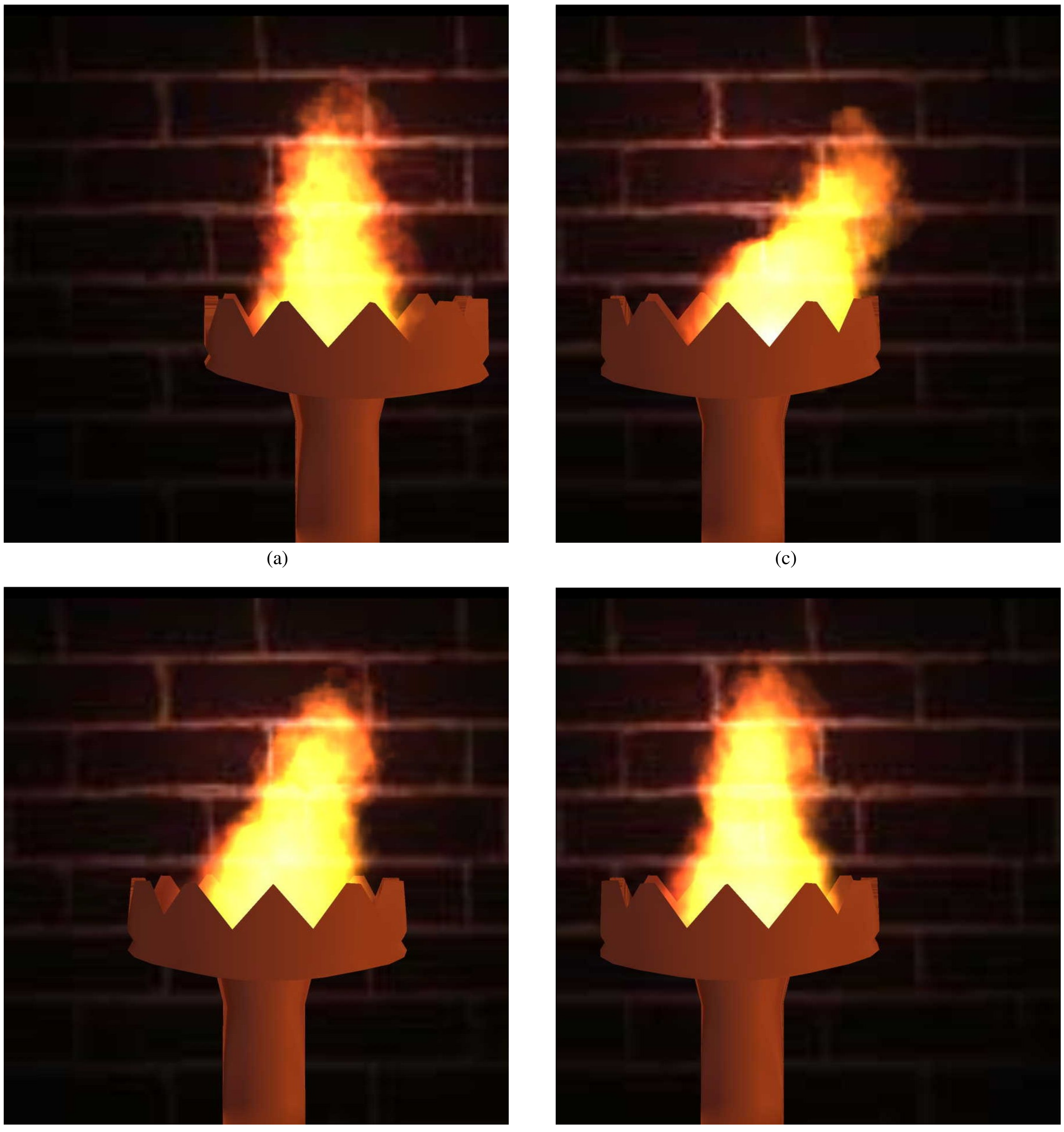


Figure 8: An image sequence (a) to (d) showing the dynamic change of fire behavior as a torch is moving to the left side of the scene.

each example, around 100 display primitives are used to generate the images. The rendering time for the entire image is 15ms.

Since in our current work, we solve the LBM on a relatively low-resolution grid, one may ask why not solve the NS equations on the same grid. Although it is true that combining the NS equations with the high-resolution texture splats has the same effect, however, the computation of the NS equations is not as simple as that of the LBM. Also, the fact that the calculation of the LBM only consists of simple operations such as addition, subtraction and multiplication, and it is conducted locally, allows us to further improve the calculation speed of the LBM by employing commodity texture hardware to achieve real-time speed.

## 10 Conclusions

In this paper, we have proposed the use of textured splats as the basic primitives to model fire. The interaction of fire and the surrounding environments, such as boundary objects and blowing wind, is described by the simple and linear LBM. This method can be used in many applications, such as virtual environment, flight simulation and landscape design. Our approach has the following advantages:

- The calculation of the LBM can be accelerated by texture hardware, due to the LBM's linear and local computations. In all of our experimental results, we were able to achieve real-time frame rates. This can be of great help when dealing with dynamic changes of the boundary conditions during 3D simulations.

- The turbulence behavior of the fire can be incorporated via texture splats. These can be rendered quickly as well on commodity texture hardware.

There still remain a few things to be investigated. First, we plan to extend our model to handle the propagation of fire in complex polygonal models. We believe that combining our work with the fire spreading models [1, 12, 19] will be a promising direction. Second, alternating the textures of the display primitives during the simulation will also help to get visually more pleasing results. Finally, we would like to incorporate more complex illumination effects in future implementations, such as the heat-based refractive effects that occur when objects are viewed across the hot fire.

## Acknowledgments

This work has been partially supported by ONR grant N000140110034 and NSF CAREER grant ACI0093157.

## References

[1] P. Beaudoin, S. Paquet, and P. Poulin. Realistic and controllable fire simulation. *Proc. Graphics Interface*, pages 159–166, June 2001.

[2] S. Chen and G. D. Doolean. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329–364, 1998.

[3] 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.

[4] R. A. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. *Proceedings of IEEE Visualization*, pages 91–98, October 1993.

[5] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita. A simple, efficient method for realistic animation of clouds. *Proceedings of SIGGRAPH*, pages 121–128, August 2000.

[6] R. Fedkiw, J. Stam, and H. Jensen. Visual simulation of smoke. *Proceedings of SIGGRAPH*, pages 129–136, August 2001.

[7] N. Foster and R. Fedkiw. Pratical animation of liquids. *Proceedings of SIGGRAPH*, pages 15–22, August 2001.

[8] N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. *Proceedings of SIGGRAPH*, pages 181–188, August 1997.

[9] M. J. Harris and A. Lastra. Real-time cloud rendering. *Computer Graphics Forum (Eurographics Proceedings)*, 20(3):76–84, September 2001.

[10] B. D. Kandhai. Large scale lattice-boltzmann simulations. PhD thesis, University of Amsterdam, December 1999.

[11] S. A. King, R. A. Crawfis, and W. Reid. Fast volume rendering and animation of amorphous phenomena. *Volume Graphics*, pages 229–242, 2000.

[12] H. Lee, L. Kim, M. Meyer, and M. Desbrun. Meshes on fire. *EG Workshop on Computer Animation and Simulation*, pages 75–84, September 2001.

[13] W. Li, X. Wei, and A. Kaufman. Accelerating lattice Boltzmann method on graphics hardware. Technical Report 010416, Computer Science Department, SUNY at Stony Brook, April 2001 (revised version submitted for publication).

[14] D. Muders. Three-dimensional parallel lattice boltzmann hydrodynamics simulations of turbulent flows in interstellar dark clouds. PhD thesis, University at Bonn, August 1995.

[15] D. Nguyen, R. Fedkiw, and H. Jensen. Physically based modeling and animation of fire. *Proceedings of SIGGRAPH*, to appear 2002.

[16] T. E. Nielsen. Modelling, animation, and visualization of fire. Master's thesis, University of Copenhagen, Denmark, April 1999.

[17] K. Perlin. An image synthesizer. *Proceedings of SIGGRAPH*, 19(3):287–296, July 1985.

[18] K. Perlin and E. M. Hoffert. Hypertexture. *Proceedings of SIGGRAPH*, 20(3):253–262, July 1989.

[19] C. H. Perry and R. W. Picard. Synthesizing flames and their spreading. *Proceedings of the Fifth Eurographics Workshop on Animation and Simulation*, pages 1–14, September 1994.

[20] W. T. Reeves. Particle system-a technique for modeling a class of fuzzy objects. *Proceedings of SIGGRAPH*, 17(3):359–376, July 1983.

[21] J. Stam. Stable fluids. *Proceedings of SIGGRAPH*, pages 121–128, August 1999.

[22] J. Stam and E. Fiume. Depiction of fire and other gaseous phenomena using diffusion processes. *Proceedings of SIGGRAPH*, pages 129–136, August 1995.

[23] L. Westover. Footprint evaluation for volume rendering. *Proceedings of SIGGRAPH*, pages 367–376, August 1990.