Visual Simulation of Heat Shimmering and Mirage

Ye Zhao, *Member*, *IEEE*, Yiping Han, Zhe Fan, Feng Qiu, Yu-Chuan Kuo, Arie E. Kaufman, *Fellow*, *IEEE*, and Klaus Mueller, *Member*, *IEEE*

Abstract—We provide a physically-based framework for simulating the natural phenomena related to heat interaction between objects and the surrounding air. We introduce a heat transfer model between the heat source objects and the ambient flow environment, which includes conduction, convection, and radiation. The heat distribution of the objects is represented by a novel temperature texture. We simulate the thermal flow dynamics that models the air flow interacting with the heat by a hybrid thermal lattice Boltzmann model (HTLBM). The computational approach couples a multiple-relaxation-time LBM (MRTLBM) with a finite difference discretization of a standard advection-diffusion equation for temperature. In heat shimmering and mirage, the changes in the index of refraction of the surrounding air are attributed to temperature variation. A nonlinear ray tracing method is used for rendering. Interactive performance is achieved by accelerating the computation of both the MRTLBM and the heat transfer, as well as the rendering on contemporary graphics hardware (GPU).

Index Terms—Heat transfer, lattice Boltzmann model, thermal flow dynamics, heat shimmering, mirage, GPU acceleration, nonlinear ray tracing.

1 Introduction

Various natural phenomena involve hot objects, dynamic flows, and heat transfers, such as melting, dissolving, shimmering, and mirage, which are of great interest to researchers in computer graphics and scientific simulations. For simulating these phenomena, it is imperative to provide a correct and efficient modeling of the heat transfer as well as the interaction between the objects and the flow. This paper presents a physically-based method that provides a basic framework for modeling these thermal phenomena.

The shimmering effect can be achieved by ad hoc data-driven methods, such as noise-based approaches. However, these require a substantial amount of manual work to adjust the parameters. In contrast, our method simulates the phenomena physically with thermal air flow effects. The parameters in our system have real physical meanings and are easily employed to control the resulting behavior, especially for realistic 3D scenarios. Our method is particularly valuable as it is not easy for a noise-based model to implement the shimmering effect once air flows are interacting with internal objects. Finally, although the focus of this paper is on heat shimmering and mirage, our method also works well with other thermal flow phenomena. For example, it can be used to model air flow and contaminant transport in urban environments with the inclusion of thermal effects due to surface heating by the sun.

- *Y. Zhao is with the Department of Computer Science, Kent State University, Kent, OH 44242. E-mail: zhao@cs.kent.edu.*
- *Y. Han, Z. Fan, F. Qiu, Y.-C. Kuo, A.E. Kaufman, and K. Mueller are with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794.*
  E-mail: {yhan, fzhe, qfeng, yukuo, ari, mueller}@cs.sunysb.edu.

Manuscript received 19 Dec. 2005; revised 4 May 2006; accepted 17 May 2006; published online 8 Nov. 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0219-1205.

Our method includes conduction, convection, and radiation, which are the three basic types of heat transfer in the real world. Heat sources are defined as any arbitrarily shaped objects interacting with the surrounding air. The temperature distribution on the objects can be calculated from radiators (e.g., the sun) or defined by the user with other physical or nonphysical methods. Such temperature distribution is applied to the surface geometry by a novel mechanism, termed *temperature texture*. We model the heat transfer from the heat sources to the ambient flow. Although heat transfer modeling has been used before in computer graphics, to the best of our knowledge, this is the first physically-based implementation for heat exchange between arbitrarily-shaped, heated objects and the surrounding air. The different heat exchange behaviors are determined by material and flow properties, which are controlled by physically meaningful parameters, such as thermal conductivity, Prandtl number, and flow velocity. In the air region, a hybrid thermal lattice Boltzmann model (HTLBM), which couples the multiple-relaxation-time LBM (MRTLBM) with a finite difference discretization of an advection-diffusion equation for temperature, is used for modeling the thermal flow dynamics.

Heat shimmering and mirage appear when the heated air has a different refractive index than that of the cooler surrounding air, resulting in an altered light direction through the hot air compared to that of the cooler air. Here, the changes in the index of refraction are attributed to temperature variation. Once the dynamic temperature distribution is computed by our physically-based modeling framework, we apply a nonlinear ray tracing method to render the resulting visual effects. The local and explicit operations of the HTLBM make it possible to accelerate both the physical simulation and the rendering on a contemporary graphics processing unit (GPU) for interactive performance.

In summary, the main contributions of this paper are: 1) proposing the first approach that physically models the heat transfer from heat sources to the surrounding air, 2) introducing the concept of *temperature texture* to represent temperature distributions on heat sources, and 3) implementing thermal flow modeling in an HTLBM framework with GPU acceleration. Our method is applicable for the visual simulation of heat shimmering and mirage, and it can be used to model various other thermal flow phenomena.

2 Background

Modeling the temperature evolution in a flow is an essential step for heat-related phenomena. In earlier work concerned with the modeling of shimmering and mirage [1], [22], the temperature or index of refraction is manually established for particular situations. In another work [25], the temperature fields are determined by superinterpolated weighted blobs, which are created at user-specified heat sources with an initial temperature. The temporal evolution of the blobs is achieved by advecting their centers through a turbulent wind field, which is generated by a physically-based simulation [24].

The temperature evolution in fluid has also been modeled by others, but not specifically for shimmering or mirage effects. Over the past decade, a variety of physics-based approaches have been applied to model fluid dynamics. In particular, numerical methods for solving the Navier-Stokes (NS) equations have led to significant advances in the visualization of gas, fire, and fluids [3], [6], [8], [9], [23], in which the temperature is considered to affect the flow dynamics. Usually, an advection-diffusion equation governs the temperature evolution. By applying Boussinesq approximations (the density variation only appears in the force term), a buoyancy force determined by the temperature is applied to the fluid equations.

Unlike previous work that manually arranges the heat distribution or calculates it by heated blobs advected in a wind field, our framework simulates the temperature evolution by physically modeling the thermal flow dynamics. Our method is also different from the fluid modeling methods that include temperature evolution. These existing approaches only consider the heat evolution inside the fluid or implement the heat transfer between the fluid and the suspended particles by an empirical equation [7], while ours models the heat transfer between the arbitrarily shaped heat sources with different material properties and the surrounding air.

LBM [26] is a relatively new approach in computational fluid dynamics (CFD) with a simple and parallelizable grid-based numerical scheme. The fundamental idea of the LBM is to construct simplified kinetic models that incorporate the essential physics of microscopic processes such that the macroscopic averaged properties obey the desired macroscopic NS equations. The LBM does not need to iteratively solve the large linear system produced by the Poisson equation of the pressure. Therefore, it has the advantage of being easy to implement and, in addition, it is especially suitable for GPU acceleration. Furthermore, the LBM scheme handles complex and moving boundaries very well. However, the LBM is an explicit solver and requires relatively small time steps. Multiresolution schemes and adaptive time step schemes have been applied to the LBM to address this issue [17]. Generally, the LBM has achieved success in the world of physics both from the analytical and practical points of view. It has also been used in graphics for simulating a variety of fluid phenomena with complex boundary conditions [4], [21], [27], [28], [30]. Comparing our current work with the former LBM work, we are the first to implement the thermal LBM in graphics, where the air flow is affected by the heat and also contributes to the heat evolution.

Some previous research has reported on the rendering of heat shimmering and mirage. Berger et al. [1] first implemented ray tracing for mirages. They generated mirage images by sending rays through multiple air layers with different refractive indices. Musgrave [19] pointed out that the primary reason for mirage creation is total reflection. Groeller [10] developed algorithms to trace nonlinear rays through a class of force fields. Berger et al. [1] calculated the trajectory of a light ray by changing the refractive index gradient parameters. Stam and Languenou [25] presented a method to trace the rays by integrating the basic equations from geometrical optics with perturbation. The equations govern the propagation of rays in a medium with a continuously varying index of refraction. Recently, Seron et al. [22] solved the problem of light propagation through an inhomogeneous medium using a general equation based on Fermat's principle. They applied the method to the distortions of the spherical shape of the sun during sunsets. Unlike these methods, our approach computes the index of refraction physically as a function of the pressure and temperature at a particular position. Such computation is performed at discrete steps when a light ray traverses the medium and the ray direction is changed at each step by applying Snell's Law to model the refraction of the ray.

In the next section, we describe our heat transfer model for heat sources, as well as the temperature texture. We then describe our thermal flow modeling in Section 4. In Section 5, the nonlinear ray tracing rendering scheme is discussed. In Section 6, we describe our GPU acceleration. Finally, several examples demonstrate the visual results of our method.

3 Heat Transfer

In the real world, there are three ways in which heat may be transferred between substances that are at different temperatures: conduction, convection, and radiation [12]. The flow of heat by conduction occurs when a temperature gradient exists in a body. Different materials transfer heat between them by conduction at different rates, which is measured by the material's thermal conductivity k. Fourier's law describes the heat conduction as

$$q = -k\nabla T, \tag{1}$$

where q is the heat-transfer rate per unit area and ∇T is the temperature gradient.

Convection involves the energy exchange between a surface and an adjacent fluid. The fluid immediately adjacent to the surface forms a thin boundary layer. Heat is conducted into this layer, where the fluid carries the heat away. A high velocity produces a large temperature gradient. Thus, the temperature gradient at the wall

depends on the flow field. Newton's law of cooling expresses the effect of convection as

$$q = h(T_{body} - T_\infty), \tag{2}$$

where h is the convection heat-transfer coefficient which is a function of the geometry and fluid properties, T_{body} is the temperature of the surface, and T_∞ is the temperature of the oncoming fluid.

Radiation is the heat transfer by emission and absorbtion via electromagnetic waves. It may occur through regions where no material medium is involved; for example, from the sun to the earth through mostly empty space. An ideal thermal radiator (black body) emits energy at a rate proportional to the fourth power of the temperature T of the body and directly proportional to its surface area A (σ is the Stefan-Boltzmann constant) as

$$q_{emitted} = \sigma A T^4. \tag{3}$$

In our method, heat transfer occurs in three different situations: 1) in the air (conduction and convection), 2) from the heat source objects to the ambient air (convection), and 3) on the heat source objects (user defined or radiation from the sun). The first phenomenon is modeled with a standard advection-diffusion equation of temperature, which is part of the HTLBM to simulate thermal flow dynamics and is discussed in Section 4. Here, we describe our heat transfer implementation in the other two situations. To the best of our knowledge, physically-based modeling of heat transfer from arbitrarily shaped objects to the air with different material properties, as well as radiation from the sun to the objects, have not appeared before in computer graphics.

3.1 Heat Transfer from Heat Sources to the Air

Heat exchange between the object surface and the air is modeled by (2), where h is the key coefficient to be computed according to flow velocity, material, and geometry. Considering a thin thermal boundary layer, with a characteristic length L, a local Nusselt number (a ratio of conductive to convective thermal resistance of the fluid) is defined as

$$Nu = \frac{hL}{k}. \tag{4}$$

For a plate interface, Nu relates to the flow properties as

$$Nu = 0.332 Re^{\frac{1}{2}} Pr^{\frac{1}{3}}, \tag{5}$$

where Pr is the Prandtl number that approximates the ratio of momentum diffusivity and thermal diffusivity. The value of Pr for different materials can be found in physics handbooks. Re is the local Reynolds number at position P in the thermal boundary layer, which is the ratio of inertial to viscous forces and is defined as

$$Re = \frac{\mathbf{u}L}{\nu}, \tag{6}$$

where ν is the kinematic fluid viscosity, $\mathbf{u}$ is the flow velocity, and L is the distance from P to the plate interface.

For an arbitrary object, the interface between the air and a small region of itself can be locally simplified as a plate interface. Therefore, we can combine (4) and (5) to compute the local heat-transfer coefficient h at a position P, and then calculate the temperature change by (2). Note that the radiation from the object to the air is usually very small and, thus, is neglected.

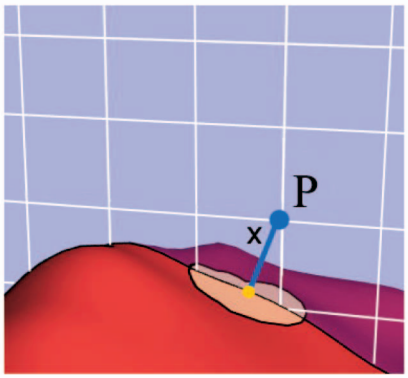


Fig. 1. Heat transfer surrounding a heat source boundary.

As shown in Fig. 1, on a blue cutting plane of the LBM grid (in white lines), a grid point P has a closest distance x to the red object surface. P has a flow velocity $\mathbf{u}$ and viscosity ν. In our simulation, we modify the temperature values of the LBM grid points that are close to the heat source surface (i.e., in the thin thermal boundary layer) as follows:

1. Find the Prandtl number, Pr, of air and the thermal conductivity, k, of the heat source material in physics handbooks.
2. Calculate the local Reynolds number, Re, from (6); let $L = x$.
3. Combine (4) and (5) to compute the convection heat-transfer coefficient, h.
4. Use (2) to modify the temperature T_∞ of point P from the heat source temperature T_{body}.

Following this algorithm, the temperatures of the LBM lattice points inside the thermal boundary layer are modified at every computational step. This procedure sets up the thermal boundary condition of the heat transfer computation in the air.

In the algorithm above, T_{body} of the object surface is the heat source temperature. Given the temperature distribution of the object surface, we calculate the average T_{body} in a surface area surrounding the closest point to P. In Fig. 1, this area is shown as the yellow region on the red object surface. This mechanism is used because the surface temperature distribution typically has a higher resolution than that of the LBM lattice. The temperature distribution of the heat source is discussed in Section 3.2 and Section 3.3.

3.2 Heat Sources

Our model considers two kinds of heat sources. The first kind of heat sources are objects that generate heat themselves, such as room heaters, hot food, grills, etc. We define the temperature distribution on such object surfaces using temperature textures, described in Section 3.3. On the other hand, objects may absorb energies coming from outer radiators. For outdoor scenes, the sun is the most important

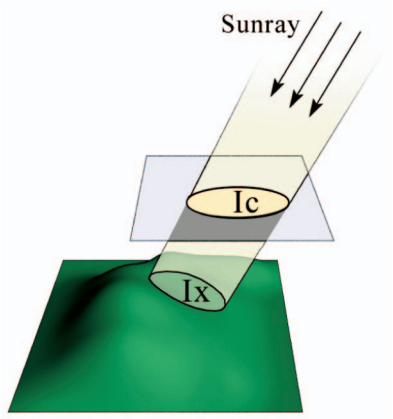


Fig. 2. Sun irradiation reaches a nonhorizontal surface.

radiator, which is located at infinity and sun-rays that arrive at the objects are all parallel to each other.

The intensity of solar radiation heavily depends on atmospheric conditions, time of the year, and the incident angle for the sun-ray on the surface of the objects. At the outer limit of the atmosphere, the total solar irradiation is $E_{bo} = 1,395W/m^2$. The actual solar irradiation that can reach a planar surface on the ground can be calculated as

$$\frac{I_c}{I_o} = e^{-na_{ms}m}, \tag{7}$$

where $I_o = E_{bo}sin\alpha$ (α is the angle between the sun-ray and the surface), $m = \csc\alpha$ is the relative thickness of the air mass, $a_{ms} = 0.128 - 0.054\log m$ is the molecular scattering coefficients and n is the turbidity factor [12].

The surface of the heat source objects may not always be flat. We further project the solar irradiation I_c from a planar surface onto surface areas at an arbitrary orientation, called I_x, as shown in Fig. 2. At a radiation equilibrium, the temperature of the object surface can then be calculated from the following equation:

$$\left(\frac{I_x}{A}\right)_{sun} \alpha_{sun} = \alpha_{lowtemp}\sigma(T^4 - T_{surr}^4), \tag{8}$$

where A is the area of the surface, α_{sun} is the absorptivity between the object surface and the sun, and $\alpha_{lowtemp}$ is the absorptivity between the object surface and surrounding air. $\sigma = 5.669 \times 10^{-8}W/m^2$ is the Stefan-Boltzmann constant. α_{sun} and $\alpha_{lowtemp}$ vary between different materials and can be found in physics handbooks. These two parameters determine the surface temperature. Therefore, different materials show different temperature distributions under the same radiation. Using heat shimmering as an example, Fig. 3 shows the different shimmering results created by three materials exposed to the sun: white paint with weak shimmering, asphalt with moderate shimmering, and copper with strong shimmering. The absorptivity parameters of the white paint is set as $\alpha_{sun} = 0.12$ and $\alpha_{lowtemp} = 0.90$ [12]. For asphalt, $\alpha_{sun} = 0.90$, $\alpha_{lowtemp} = 0.90$ and, for copper, $\alpha_{sun} = 0.18$, $\alpha_{lowtemp} = 0.03$. The sun-ray has an incident angle of 75 degrees and n is set to 2.0.

In the simulation, we compute the temperature distribution of an outdoor object as follows:

1. Choose the appropriate parameters for the scene: environment temperature, incident direction of the sun-ray, material of the object, etc.
2. Calculate the solar radiation energy on a planar surface by (7). This planar surface approximates a large region of the object and is actually formed by small regions with different orientations.
3. For every small region of this planar surface:
   - Calculate the actual heat on the region by projecting the radiation from the surface.
   - Calculate the equilibrium temperature by (8).

3.3 Temperature Texture

Heat sources are modeled as geometric objects inside the computational volume. Usually, the computational lattice of the LBM has a lower resolution when considering the temperature distribution. When we compute the temperature of one LBM grid point, we cannot use only the closest point on the object surface. The heat variation between two neighboring grid points will be lost. We therefore utilize the idea of texture mapping and propose a temperature texture to overcome this problem. Temperature textures are associated with the heat source objects. Instead of mapping colors to the objects, temperatures are mapped to the objects. The temperature texture can be computed either by physical methods (as in Section 3.2) or by user definitions. During simulation, the temperature of an LBM grid point is

Fig. 3. A sun-heated surface composed of different materials, resulting in different shimmering effects on a stone wall in the background. (a) White paint, (b) asphalt, and (c) copper.

Fig. 4. A snapshot of the shimmering effects from a simplified heated terrain with its temperature texture rendered in color over the terrain.

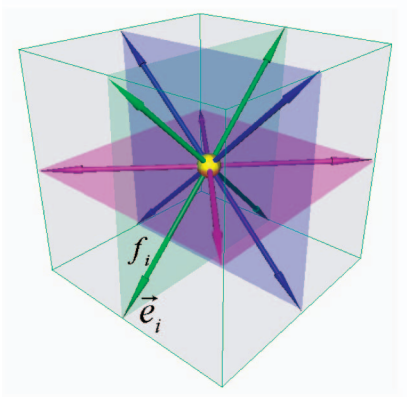


Fig. 5. The D3Q13 lattice geometry. The particle distribution f_i is associated with the link corresponding to the $\mathbf{e}_i$ velocity vector.

computed by interpolating in a region around its closest heat source point in the temperature texture. Therefore, the temperature texture provides a good data structure to store heat distribution for thermal boundary conditions, and it is straightforward in a GPU implementation.

Fig. 4 illustrates a snapshot of the shimmering effects over a brick wall from a simplified heated terrain with its temperature texture rendered over the surface. Different colors illustrate temperature difference: Red indicates higher temperatures, yellow indicates medium temperatures, and green indicates lower temperatures.

4 Thermal Flow Modeling

We model and simulate thermal flow dynamics using a hybrid thermal lattice Boltzmann model (HTLBM). In this approach, the heat transfer in the air is modeled with an advection-diffusion equation coupled to the LBM.

Most of the previous LBM works in graphics [27], [28], [30] are based on a single-relaxation-time LBM (SRTLBM), where one constant related to the viscosity, the relaxation time, is used to control the behavior of the fluid. In SRTLBM, the thermal effects cannot be easily incorporated in the flow dynamics. In this paper, a hybrid TLBM (HTLBM) [14] that couples temperature, modeled by an advection-diffusion equation, to the multiple-relaxation-time LBM (MRTLBM) [5] is used for modeling thermal flows. We have introduced this scheme to the visualization community [21], where MRTLBM was applied to dispersion simulation in an urban environment. In this paper, we implement the HTLBM with graphics hardware acceleration to model the thermal flow dynamics and, specifically, those related to heat shimmering and mirage.

4.1 Multiple-Relaxation-Time Lattice Boltzmann Method

LBM models Boltzmann particle dynamics on a 3D lattice. The Boltzmann equation expresses how the average number of flow elements or "particles" with a given velocity changes between neighboring sites due to interparticle interactions and ballistic motion. The variables associated with each lattice site are the particle distributions f_i that represent the probability of the presence of a fluid particle with a given velocity direction $\mathbf{e}_i$. Particles stream synchronously along the lattice links in discrete time steps. Between streaming steps, they undergo collision.

For our work, we use the 13-velocity 3D lattice denoted as D3Q13. As illustrated in Fig. 5, this lattice is a structured grid whose unit cell includes the center cell with zero velocity and the 12 minor-diagonal neighbor links (the six axial and eight major-diagonal links are not used). For a node $\mathbf{r}$ at time t, the macroscopic fluid density, $\rho(\mathbf{r}, t)$, and velocity, $\mathbf{u}(\mathbf{r}, t)$, are computed from the velocity distributions as follows:

$$\rho = \sum_i f_i \qquad \mathbf{u} = \frac{1}{\rho}\sum_i f_i \mathbf{e}_i. \tag{9}$$

In SRTLBM, the Bhatnager, Gross, Krook (BGK) model is usually used to represent the particle collisions [26]. The BGK model represents collisions as a statistical redistribution of momentum, which locally drives the system toward equilibrium while conserving mass and momentum. In terms of this model, the Boltzmann dynamics can be represented as a two-step process of collision and ballistic streaming:

$$f_i(\mathbf{r}, t^+) = f_i(\mathbf{r}, t) - \frac{1}{\tau}(f_i(\mathbf{r}, t) - f_i^{eq}(\rho, \mathbf{u})), \tag{10}$$

$$f_i(\mathbf{r} + \mathbf{e}_i, t + 1) = f_i(\mathbf{r}, t^+). \tag{11}$$

In these equations, $\mathbf{r}$ locates a node of the lattice and $f_i(\mathbf{r}, t^+)$ denotes the postcollision distribution. The distribution denoted as f_i^{eq} represents a local equilibrium distribution whose value depends only on conserved quantities—mass ρ and momentum $\rho\mathbf{u}$. Finally, the constant τ represents the relaxation time scale that determines the viscosity of the flow.

MRTLBM is a newer version of LBM developed by d'Humiéres et al. [5]. This collision model abandons SRTLBM to achieve better numerical stability and greater flexibility in selecting the transport coefficients. The essential idea is to make a change of basis from phase space (i.e., the space of the distributions f_i) to the space of hydrodynamic moments (i.e., density, momentum, energy, etc.) and to perform the collision step in the latter space. As in the BGK model, collisions are implemented via a relaxation, but in the moment space, each moment is

allowed to relax individually. Although the relaxation rates are not all independent, the additional flexibility allows one to maneuver the model into regions of higher stability while decoupling some of the transport coefficients. After relaxation, the inverse transformation is applied to return to phase space where streaming, boundary update rules, and additional microphysics are implemented as before.

Mathematically, the change of basis from the space of distributions to the space of moments is given by:

$$|m\rangle = M|f\rangle, \quad |f\rangle = M^{-1}|m\rangle, \tag{12}$$

$$|f\rangle = (f_0, f_1, \ldots, f_{12})^T, \tag{13}$$

$$|m\rangle = (m_0, m_1, \ldots, m_{12})^T, \tag{14}$$

where T denotes the transpose. Each of the 13 moments $\{m_i|(i = 0, 1, \ldots, 12)\}$ has a physical meaning. For example, m_0 is the mass density ρ, $m_{1,2,3}$ are the components of the momentum vector $\mathbf{j}$, m_4 is the energy, and the other higher order moments are components of the stress tensor and other high order tensors. The rows of the matrix M relate the distributions to the moments. For example, since $\rho = \sum_i f_i$, the first row of M consists of all ones. Although the values of the distributions and the moments vary over the nodes of the lattice, the matrix M is simply constant for a given lattice.

In MRTLBM, the two step process of collision and streaming becomes:

$$|f(\mathbf{r}, t^+)\rangle = |f(\mathbf{r}, t)\rangle - M^{-1}S[|m(\mathbf{r}, t)\rangle - |m^{eq}(\mathbf{r}, t)\rangle], \tag{15}$$

$$|f(\mathbf{r} + \mathbf{e_i}, t + 1)\rangle = |f(\mathbf{r}, t^+)\rangle. \tag{16}$$

The components of the vector $|m^{eq}\rangle$ are the local equilibrium values of the moments. Among them, the mass density and the momentum (m_0 to m_4) are conserved. Expressions for the nonconserved moments depend only on local values of the conserved moments [14]. The matrix S in the collision equation is a diagonal matrix whose elements are the relaxation rates, $\{s_i|(i = 0, 1, \ldots, 12)\}$. Their values are directly related to the kinematic shear and bulk viscosities, ν and ξ, respectively:

$$\nu = \frac{1}{2}\left(\frac{1}{s_6} - \frac{1}{2}\right), \tag{17}$$

$$\xi = \left(\frac{2}{3} - \gamma c_{s0}^2\right)\left(\frac{1}{s_5} - \frac{1}{2}\right), \tag{18}$$

where γ is the specific heat and c_{s0} is the isothermal speed of sound. The user has the freedom to choose the flow parameters to define characteristics of the fluid being modeled. This choice then determines the relaxation rates.

MRTLBM can also accommodate a body force due to gravity or some other external field. This is implemented by adding the force $\mathbf{F}$ to the momentum, $\mathbf{j}' = \mathbf{j} + \mathbf{F}\delta t$ (typically, $\delta t = 1$). In practice, for stability, the force term is executed in two steps, one-half before the relaxation step and one-half after.

Note that SRTLBM can be seen as a special case of MRTLBM associated with a specific choice of parameter values in the equilibria of the moments so that only one single relaxation rate, $1/\tau$, remains free. Although MRTLBM requires somewhat more computation compared to SRTLBM, much of the computational cost can be ameliorated by adopting a simpler lattice such as the D3Q13, which is made useable by the improved stability of the MRTLBM. Yet, MRTLBM retains the parallelizability of SRTLBM.

4.2 Coupling Advection-Diffusion Temperature

To capture thermal effects, temperature is coupled to MRTLBM through the energy moment that the model exposes. For the D3Q13 lattice, the energy equilibrium is modified as

$$m_4^{eq} = n_1(c_{s0}^2 - n_2)\rho + n_3(n_4 - \gamma)\mathbf{j}\cdot\mathbf{j} + q_1 T. \tag{19}$$

The new variables in (19) are the temperature T and its constant coupling coefficient q_1. The parameters, n_1 to n_4, are constants and their values are determined by linear stability analysis [14]. The heat transfer here is modeled separately with a standard advection-diffusion equation, giving rise to our HTLBM as

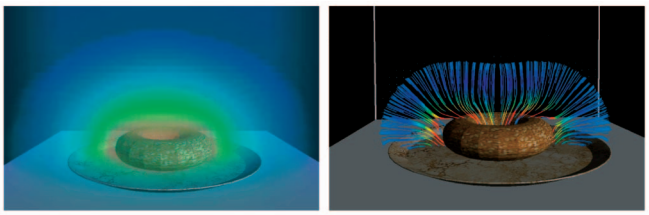
$$\partial_t T + \mathbf{u}\cdot\nabla T = \kappa\Delta T + q_2(\gamma - 1)c_{s0}^2\nabla\cdot\mathbf{u}, \tag{20}$$

where κ is the thermal diffusivity of the fluid and q_2 is another constant coupling coefficient. This equation is solved with the following finite-difference equation:

$$\begin{aligned} T(\mathbf{r}, t+1) - T(\mathbf{r}, t) = &-\mathbf{j}\cdot\nabla^* T + \kappa\Delta^* T \\ &+ q_2(\gamma - 1)\cdot c_{s0}^2\nabla^*\cdot\mathbf{j}, \end{aligned} \tag{21}$$

where $*$ denotes the corresponding finite-difference operators. The density is conserved and can be treated as a constant. By setting $\rho = 1$ in HTLBM, the velocity $\mathbf{u}$ can be replaced by the momentum $\mathbf{j}$. Since (20) has no direct relation with the computation of HTLBM, the thermal model can be replaced by any plausible thermal model. However, for stability, the stencils of the finite difference operators must respect the symmetry of the lattice. Because the LBM is an explicit fluid solver, here, we adopted the simple and explicit solver for the temperature evolution (21), which uses the same small time step and lattice spacing as in the LBM. Thus, it can be easily accelerated on parallel machines with the LBM to achieve fast performance. In our simulation, both the LBM and the temperature evolution have the limitation that they can only model flows with low Mach number. This means that the velocity of the flow should be small compared to the speed of sound. However, the air flows in the shimmering and mirage generally satisfy this requirement.

In some numerical methods [3], [6], [9], the temperature evolution is modeled and applied to the flow dynamics as a buoyancy force, which is based on the Boussinesq approximation and the coupling parameters are usually chosen manually. Our HTLBM can also easily include the body forces, however, we instead couple the temperature effect as an energy term, and the coupling parameters are computed by the physically-based linear analysis [14]. Therefore, our method can be extended to situations where the Boussinesq approximation is not satisfied, including the case when temperature-dependent transport coefficients are used. Second, unlike previous methods, ours has a viscous dissipation term in the temperature evolution equation (the



(a) (b)

Fig. 6. Hot bagel. (a) Temperature distribution surrounding the bagel. (b) Streamlines of the flow on a cutting plane.

last term in (20)). This term is set to zero for incompressible flows. The HTLBM we used simulates weakly compressible NS equations. Therefore, we include this term which, in practice, plays an important role in generating the shimmering effects.

Fig. 6 illustrates the temperature propagation effects from a hot bagel. Fig. 6a shows the temperature distribution surrounding the bagel. The color varies from red to green to blue as the temperature varies from high to medium to low. Fig. 6b shows the streamlines of the flow generated by the temperature variation on a cutting plane.

4.3 Boundary Conditions

Interactions between an LBM flow field and an immersed object result from the exchange of momentum at their shared boundaries. The treatment of boundary conditions in LBM has been discussed in our previous work. In HTLBM, boundary conditions are also handled in the discrete velocity space. We implemented periodic, out-flow and bounce-back conditions [26] for the surrounding walls of the simulation space, as well as improved bounce-back rules for curved, moving, no-slip boundaries [18] for the inside objects.

For modeling thermal flow dynamics, heat exchange at boundaries is considered as thermal boundary conditions. In solving the temperature evolution equation (20), the walls of the computational volume are treated via the adiabatical thermal boundary condition $\partial_{\hat{\mathbf{n}}} T = 0$, where $\hat{\mathbf{n}}$ defines the unit normal outward. To implement this, the temperature is computed on the same lattice in two steps. First, the advection-diffusion equation is solved for interior nodes of the lattice, then the adiabatical condition is applied for the walls. For heat source objects inside the volume, we apply the algorithm described in Section 3 to set the temperature of the nodes close to the object surfaces. This represents another type of the thermal boundary condition in (20).

4.4 Computational Procedure

To recapitulate, the computation procedure for our HTLBM simulations involves the following series of steps:

0. Initialize HTLBM with correct initial conditions and boundary conditions.
1. Perform streaming (16) of f_i for $i = 0 \ldots 12$.
2. Transform to moment space m_i, for $i = 0 \ldots 12$.
3. Add half of the body forces, $\mathbf{j}' = \mathbf{j} + \frac{1}{2}\mathbf{F}$, to $m_{1,2,3}$.
4. Compute the heat transfer and solve (21) for one step.
5. Perform collision (15), incorporating the value of T to m_4 (19).
6. Add another half of the body forces, $\mathbf{j}'' = \mathbf{j}' + \frac{1}{2}\mathbf{F}$, to $m_{1,2,3}$.
7. Transform back from m_i to f_i, for $i = 0 \ldots 12$.
8. Apply air flow boundary conditions and thermal boundary conditions.
9. Return to Step 1.

The momentum $\mathbf{j}'$ provides the flow velocity that is the output at each step. The heat transfer computation in Step 4 is computed in parallel with the LBM, and is coupled at Step 5.

5 Nonlinear Ray Tracing

The temperature variation resulting from the interaction between the heat sources and the surrounding air is computed from the method described above. The changes in the index of refraction of the air are attributed to such temperature variation. Refraction, which produces the shimmering phenomena, occurs when light rays cross the interface between regions that have different indices of refraction. The relation between the angle of incidence θ_1 and the angle of refraction θ_2 is described by Snell's Law:

$$\frac{n_1}{n_2} = \frac{\sin\theta_2}{\sin\theta_1}, \tag{22}$$

where n_1 and n_2 are the corresponding indices of refraction of the two materials, and the incident ray and the refracted ray stay in the same plane.

For air, the dependence of the index of refraction on temperature and pressure can be empirically described by the following equation [16]:

$$n = \frac{c_1 * Pa * (1.0 + Pa * (60.1 - 0.972 * T) * 10^{-10})}{1.0 + c_2 * T}, \tag{23}$$

where $c_1 = 0.0000104$, $c_2 = 0.00366$, and n is the index of refraction of air. The constant pressure of the air, Pa, is measured in Pascal and the temperature, T, in Celsius.

A light ray traverses the temperature volume with a small step size. At each step, we calculate the gradient of the temperature field by trilinear interpolation at the hit point, which defines the normal N of the interface. Then, the index of refraction is determined by (23). By bending the light ray using Snell's Law (22), the new resulting ray direction is obtained, and the ray is traversed to the next hit point. When bending the ray, total reflection may occur, which causes a mirage. Our algorithm includes this situation: When calculating θ_2 in (22), if $|\sin\theta_2| > 1$, total reflection occurs. As a consequence, the ray direction is changed to the total reflection direction at the point. Therefore, the effects of a mirage are naturally included in our model.

A heat source object is voxelized and each voxel is assigned a segmentation flag: inside or outside. For each ray, if a sampling point is inside the object, the nonlinear ray tracing stops and returns the color of the object texture.

6 Hardware Acceleration

An attractive feature of our model is that the computation is inherently local and explicitly parallel. This feature allows us to accelerate our simulation on a low-cost SIMD processor (GPU) and achieve a performance of several frames per second. Using the GPU for general-purpose computation (GPGPU) has become an active field [20]. For

TABLE 1
Per Step Modeling and Rendering Time (in Milliseconds) and Frames per Second (FPS) for the CPU, GPU, and the GPU/CPU Speedup Factor

	Modeling		Rendering	Total	FPS
	HTLBM	Heat Transfer			
CPU	439	130	3034	3603	0.28
GPU	62	10	104	176	5.7
Speedup	7.1	13.0	29.1	20.5	20.5

fluid simulation, researchers [2], [11], [13] have used the GPU to implicitly solve the NS equations by ways of finite difference methods, which iteratively solve linear systems with Jacobi, Gaussian-Seidel, conjugate gradient, or multigrid methods. While achieving interactive performance for 2D flows or simple 3D flows, these methods have not solved complicated 3D flow simulations with complex inside objects. Our model, which is based on the local lattice operations, is naturally suitable for GPU acceleration with its data parallelism and locality in memory access.

To implement the HTLBM computation on the GPU, we encode the lattice data as colors and pack them into texture atlases. The lattice operations described in Section 4.4, which update the lattice data based on neighboring attributes, are implemented in fragment programs. For each data element, the fragment programs retrieve appropriate information from the local region, then compute and update with the new values. This procedure is similar to what Li et al. [15] have proposed for the GPU-mapping of the SRTLBM computations. A similar technique also applies for the GPU implementation of the computations related to the temperature distribution on heat source objects as well as for the heat transfer from the heat sources to the air. However, because these computations are only necessary for the regions surrounding the objects, we can save storage and computation time by defining bounding boxes around the objects and only execute heat transfer computations within these regions.

For rendering, we have also implemented on the GPU the procedures required for the nonlinear ray tracing through the temperature volume. By executing the whole simulation cycle (including both computation and rendering) on the GPU, we do not need to read data from the GPU, which could be a major bottleneck. For every image pixel, a ray is shot from the eye to its position. The information of all rays is stored in a 2D texture (each texel corresponds to one ray) and is processed by a fragment program. On current GPUs, the Shader Model 3.0 allows loops, dynamic branching, and program lengths of up to 65,535 instructions. Using these facilities, we are able to use only a single pass of fragment processing to iteratively forward the rays and compute their refractions until they terminate. This allows for a much easier GPU implementation than the previous GPU-based nonlinear raycaster [29] which required multiple rendering passes.

In Table 1, we report the performance of our GPU-accelerated simulation for the desert scene in Fig. 7, timed on an nVidia GeForce 6800 Ultra. The 3D simulation lattice size is $50 \times 50 \times 50$, which is moderate for GPU implementation of both simulation and rendering. To reduce the possible low frequency artifacts, a higher resolution lattice may be adopted for the simulation. However, it will consume a great amount of computational resources (sometimes unbearable for the texture memory on the GPU) to create more visual details and, hence, cannot fulfill the performance requirement. A multiresolution LBM simulation method may be adopted to optimize the use of resources, which will be implemented in our future work. The numerical model requires $62 + 10$ ms for each time-step. For rendering an image of 400×400 pixels, with the step size set to 1.0, our GPU-based nonlinear ray-casting takes 104 ms for each frame. The total time for one combined step of modeling and rendering is 176 ms, resulting in a 5.7 frames per second animation. For comparison, we list in Table 1 the time for the same computation implemented on one 3.0GHz Pentium Xeon CPU with 1GB Memory.

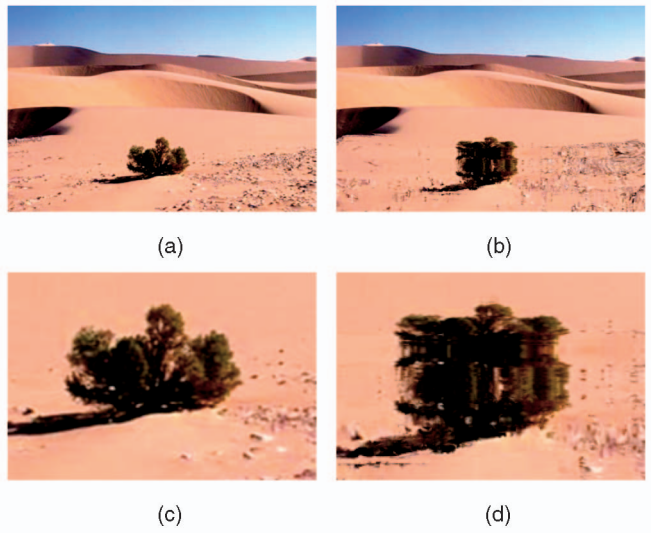


Fig. 7. Desert shimmering: (a) original scene, (b) heat shimmering, (c) zoom-in view of (a), and (d) zoom-in view of (b).

7 Results

We have applied our heat evolution computation and nonlinear ray tracing methods in several example scenarios. To better illustrate our methods, we have generated the visual results from the beginning of the heat transfer from the heat sources to the air. Therefore, the shimmering has a startup stage with a dramatic heat spreading trace and strong trembling effect before reaching a steady state with only a moderate trembling effect. The startup stage is not easily observed in the real world due to the more typical gradual heat build-up. Our animation speeds up this build-up procedure and allows the visualization of the interesting behavior of shimmering at both stages.

Fig. 7 illustrates the shimmering effects easily observed in a desert on a sunny day. The ground is heated up rapidly by the sun and the heat rises to the air. Due to the nonuniform and dynamic distribution of the air temperature, the background landscape appears distorted to the observer. In Fig. 7b, and in the corresponding zoom-in view in Fig. 7d, heat comes up from the ground and shimmering is clearly visible on the bush at the center of the scene. Shimmering phenomena can also be observed above a truck hood due to the engine heat. Such an effect is illustrated in Figs. 8a, 8b, and 8c, where one can see the distorted road and hill in the background, especially in the zoom-in view of Fig. 8d.

Fig. 8. Shimmering from a truck hood due to the engine heat. (a) Original scene, (b) heat starts to emanate from the hood, resulting in a distorted background, (c) heat shimmering rises, and (d) zoom-in view of (b).

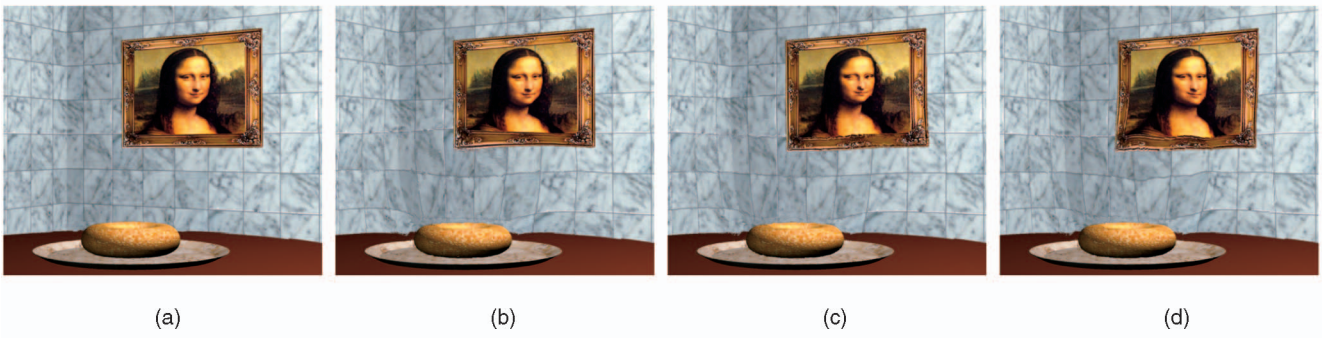


Fig. 9. Shimmering from a hot bagel. (a) Original scene, (b) heat starts to emanate from the bagel, resulting in a distorted wall behind it, (c) heat shimmering rises, and (d) wind blowing from the right.

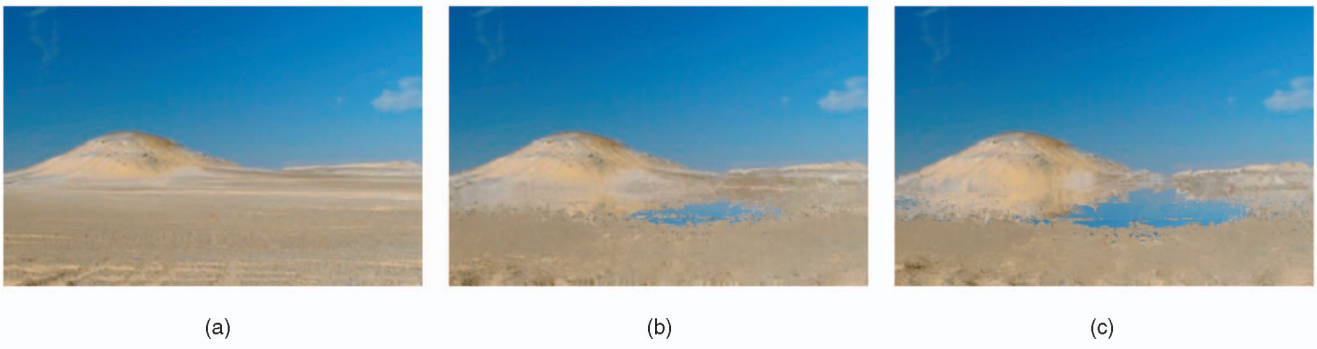


Fig. 10. Mirage in a desert. (a) Original scene, (b) a phantom body of water appears, and (c) water area becomes larger.

One of the benefits of our method is that an arbitrarily-shaped heat source can be embedded inside the thermal flow modeling volume. In Fig. 9, we model a hot bagel just unloaded from an oven. It transfers the heat to the ambient air and generates a special shimmering effect. In Fig. 9b, the bagel begins to spread heat upwards and distortion of the wall behind it is visible. The distortion rises with the heat, as shown in Fig. 9c. Our method uses the HTLBM to model flow dynamics, thus, wind can be incorporated into our example. In Fig. 9d, the shimmering is affected by a wind blowing from the right.

Mirage occurs when some rays are bent by total reflection, a situation which is handled naturally in our algorithm. In Fig. 10, comparing with a static desert scene (Fig. 10a), a phantom body of water appears in the desert with shimmering (Fig. 10b) and may become larger (Fig. 10c). In Fig. 11, we compare a real photo with our mirage effect. Fig. 11a is a real photo taken in Finland. Fig. 11b shows an original synthetic scene with no mirage. Using it as the background and starting our simulation, the mirage effect similar to the real photo (of Fig. 11a) appears, as shown in Fig. 11c.

As a lattice-based method, the LBM simulation suffers from the same problem as other Eulerian methods—the accuracy of the simulation depends on the size of the simulation grid. Low frequency noise may appear in a low-resolution simulation which is required in order to achieve interactive simulation speed. This problem can be overcome by using a high-resolution simulation lattice at the expense of performance. Alternatively, without compromising the simulation performance, high frequency small-scale details can be added to the temperature volume that is used for rendering, thereby, reducing the low frequency artifacts.

8 Conclusions

We have described a physically-based solution for simulating and animating heat shimmering and mirage phenomena. It allowed us to physically model one of the most important heat exchange scenarios: between arbitrarily shaped heat objects and the surrounding air. The temperature distribution on the heat source is defined by a temperature texture. We have further implemented a hybrid thermal LBM method to

Fig. 11. Mirage over water. (a) A real photo pictured in Finland, (b) original synthetic scene, and (c) mirage effect of the scene in (b), similar to the photo in (a) generated by our simulation.

simulate the thermal flow dynamics. By these means, we have modeled and animated the heat evolution that occurs in the real world using a physically-based approach. The temperature variation affects the trajectory of light rays, which generates the shimmering and mirage phenomena. We have introduced a nonlinear ray tracing method to render the visual results of these phenomena. Interactive performance has been obtained by implementing both the simulation and rendering on the GPU.

Our method provides a framework for modelling the thermal interaction between heat sources, objects, and the thermal flow. In the future, we will extend our framework to simulate other thermal flow phenomena (melting, dissolving, boiling, etc.) by incorporating object deformation, morphing, and phase-changing.

Acknowledgments

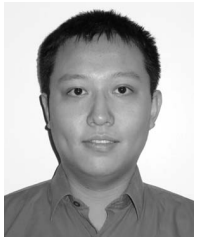
The animation movies of the examples can be downloaded from the authors Web site http://www.cs.sunysb.edu/~vislab/projects/amorphous/ShimmeringMirage. This work has been partially supported by US National Science Foundation grants CCR-0306438 and ACI-0093157. The authors thank Autodesk/Alias and Pixar for providing them with Maya and Renderman software, which they have used to render the images. Fig. 11a is courtesy of Virtual Finland, the Ministry for Foreign Affairs of Finland. The truck model in Fig. 8 is courtesy of the Turbo Squid Inc. Ye Zhao was a PhD candidate at Stony Brook University at the time of this study.

References

[1] M. Berger, T. Trout, and N. Levit, "Ray Tracing Mirages," *IEEE Computer Graphics and Applications,* vol. 10, no. 3, pp. 36-41, 1990.
[2] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder, "Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 917-924, 2003.
[3] M. Carlson, P. Mucha, R. Horn, and G. Turk, "Melting and Flowing," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* pp. 167-174, 2002.
[4] N. Chu and C. Tai, "Moxi: Real-Time Ink Dispersion in Absorbent Paper," *Proc. ACM SIGGRAPH,* pp. 504-511, 2005.
[5] D. d'Humiéres, I. Ginzburg, M. Krafczyk, P. Lallemand, and L. Luo, "Multiple-Relaxation-Time Lattice Boltzmann Models in Three-Dimensions," *Philosophical Trans. Royal Soc. of London,* vol. 360, no. 1792, pp. 437-451, 2002.
[6] R. Fedkiw, J. Stam, and H. Jensen, "Visual Simulation of Smoke," *Proc. ACM SIGGRAPH,* pp. 15-22, 2001.
[7] B. Feldman, J. O'Brien, and O. Arikan, "Animating Suspended Particle Explosions," *Proc. ACM SIGGRAPH,* pp. 708-715, 2003.
[8] N. Foster and D. Metaxas, "Realistic Animation of Liquids," *Graphical Models and Image Processing,* vol. 58, no. 5, pp. 471-483, 1996.
[9] N. Foster and D. Metaxas, "Modeling the Motion of a Hot, Turbulent Gas," *Proc. ACM SIGGRAPH,* pp. 181-188, 1997.
[10] E. Groeller, "Nonlinear Ray Tracing: Visualizing Strange Worlds," *The Visual Computer,* vol. 11, no. 5, pp. 263-374, 1995.
[11] M. Harris, "Fast Fluid Dynamics Simulation on the GPU," *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics,* R. Fernando, ed., chapter 38, pp. 637-665, Addison-Wesley, 2004.
[12] J. Holman, *Heat Transfer,* sixth ed. McGraw-Hill, Inc., 1986.
[13] J. Krüger and R. Westermann, "Linear Algebra Operators for GPU Implementation of Numerical Algorithms," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 908-916, 2003.
[14] P. Lallemand and L. Luo, "Theory of the Lattice Boltzmann Method: Acoustic and Thermal Properties in Two and Three Dimensions," *Physical Rev. E,* vol. 68 p. 036706, 2003.
[15] W. Li, Z. Fan, X. Wei, and A. Kaufman, "Flow Simulation with Complex Boundaries," *GPU Gems II: Programming Techniques for High-Performance Graphics and General-Purpose Computation,* M. Pharr, ed., chapter 47, pp. 747-764, Addison-Wesley, 2005.
[16] D. Lide, *Handbook of Chemistry and Physics,* 84th ed. CRC Press LLC, 2003.
[17] C. Lin and Y. Lai, "Lattice Boltzmann Method for on Composite Grids," *Physical Rev. E,* vol. 62, no. 2, pp. 2219-2225, 2000.
[18] R. Mei, S. Luo, and W. Shyy, "An Accurate Curved Boundary Treatment in the Lattice Boltzmann Method," *J. Computational Physics,* vol. 155, pp. 307-330, June 1999.
[19] F. Musgrave, "A Note on Ray Tracing Mirages," *IEEE Computer Graphics and Applications,* vol. 10, no. 6, pp. 10-12, 1990.
[20] J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, and T. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware," *Proc. Eurographics State-of-the-Art Reports,* pp. 21-51, Aug. 2005.
[21] F. Qiu, Y. Zhao, Z. Fan, X. Wei, H. Lorenz, J. Wang, S. Yoakum-Stover, A. Kaufman, and K. Mueller, "Dispersion Simulation and Visualization for Urban Security," *Proc. IEEE Visualization Conf.,* pp. 553-560, Oct. 2004.
[22] F. Seron, D. Gutierrez, G. Gutierrez, and E. Cerezo, "Visualizing Sunsets through Inhomogeneous Atmospheres," *Proc. Computer Graphics Int'l Conf.,* pp. 349-356, 2004.
[23] J. Stam, "Stable Fluids," *Proc. ACM SIGGRAPH,* pp. 121-128, 1999.
[24] J. Stam and E. Fiume, "Turbulent Wind Fields for Gaseous Phenomena," *Proc. ACM SIGGRAPH,* pp. 369-376, 1993.
[25] J. Stam and E. Languenou, "Ray Tracing in Not-Constant Media," *Proc. Conf. Rendering Techniques,* pp. 225-234, 1996.
[26] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond,* Numerical Math. and Scientific Computation. Oxford Univ. Press, 2001.

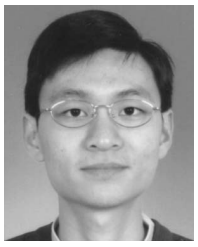
[27] N. Thurey and U. Rude, "Free Surface Lattice-Boltzmann Fluid Simulations with and without Level Sets," *Proc. Workshop Vision, Modelling, and Visualization,* pp. 199-208, 2004.
[28] X. Wei, Y. Zhao, Z. Fan, W. Li, F. Qiu, S. Yoakum-Stover, and A. Kaufman, "Lattice-Based Flow Field Modeling," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 6, 719-729, Nov./Dec. 2004.
[29] D. Weiskopf, T. Schafhitzel, and T. Ertl, "GPU-Based Nonlinear Ray Tracing," *Proc. Computer Graphics Forum,* vol. 23, no. 3, pp. 625-633, 2004.
[30] Y. Zhao, L. Wang, F. Qiu, A. Kaufman, and K. Mueller, "Melting and Flowing in Multiphase Environment," *Computers & Graphics,* vol. 30, no. 4, 2006.

Ye Zhao received the BE and MS degrees in computer science from the Tsinghua University of China in 1997 and 2000. He further received the MS and PhD degrees in computer science at the Stony Brook University in 2002 and 2006. He is currently an assistant professor in the Department of Computer Science at Kent State University. His research interests include natural phenomena modeling and visualization, general purpose computing on graphics hardware, and volume graphics. He is a member of the IEEE. For more information, see http://www.cs.kent.edu/~zhao.



Yiping Han graduated from the Computer Science Department at Stony Brook University. He received the BS degree in computer science from Zhejiang University of China in 2001 the MS degree in computer science from Stony Brook University in 2005.

Zhe Fan received the BS degree in computer science from the University of Sciences and Technology of China in 1998 and the MS degree in computer science from Chinese Academy of Sciences in 2001. He is a PhD candidate in computer science at Stony Brook University. His research interests include computer graphics, visualization, physically-based modeling, and parallel computing. For more information, see http://www.cs.sunysb.edu/~fzhe.



Feng Qiu received the BS degree in computer science from the Peking University of China in 1997 and the ME degree in computer engineering from the Chinese Academy of Sciences in 2000. He is a PhD candidate in the Department of Computer Science at Stony Brook University. His current research interests are focused on hardware accelerated rendering. For more information, see http://www.cs.sunysb.edu/~qfeng.

Yu-Chuan Kuo received the BS degree in computer science and information engineering from the National Taiwan University in 2002. He is an MS student in the Department of Computer Science and the Department of Applied Mathematics and Statistics at Stony Brook University. For more information, see http://www.cs.sunysb.edu/~yukuo.

Arie E. Kaufman received the BS degree (1969) in mathematics and physics from the Hebrew University of Jerusalem, Israel, the MS degree (1973) in computer science from the Weizmann Institute of Science, Rehovot, Israel, and the PhD degree (1977) in computer science from the Ben-Gurion University, Israel. He is a distinguished professor and chair of the Computer Science Department and the director of the Center for Visual Computing (CVC) at the State University of New York at Stony Brook (SBU). He is an IEEE fellow, a member of the IEEE Computer Society, and the recipient of IEEE Visualization Career Award (2005). He further received the IEEE Outstanding Contribution Award (1995), the ACM Service Award (1998), the IEEE CS Meritorious Service Award (1999), was a member of the European Academy of Sciences (2002), the State of New York Entrepreneur Award (2002), the IEEE Harold Wheeler Award (2004), and the State of New York Innovative Research Award (2005). Dr. Kaufman was the founding Editor-in-Chief of *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 1995-1998. He has been the cofounder, papers/program cochair, and member of the steering committee of IEEE Visualization Conferences; cofounder/chair of Volume Graphics Workshops; cochair for Proceedings of Eurographics/SIGGRAPH Graphics Hardware Workshops, the papers/program cochair for Proceedings of the ACM Volume Visualization Symposia. He previously chaired and is currently a director of the IEEE CS Technical Committee on Visualization and Graphics. He has conducted research and consulted for more than 35 years specializing in volume visualization, graphics architectures, algorithms, and languages, virtual reality, user interfaces, multimedia, and their applications. For more information, see http://www.cs.sunysb.edu/~ari.

Klaus Mueller received the MS degree in biomedical engineering in 1991 and the PhD degree in computer science in 1998, both from Ohio State University. He is currently an associate professor in the Computer Science Department at Stony Brook University, where he also holds coappointments in the Biomedical Engineering and Radiology Departments. His current research interests are computer and volume graphics, visualization, medical imaging, and computer vision. He won the US National Science Foundation CAREER award in 2001 and has served as a program cochair at various conferences, such the Volume Graphics Workshop, IEEE Visualization, and the Symposium on Volume Visualization and Graphics. He has authored and coauthored more than 70 journal and conference papers. He is a member of the IEEE and the IEEE Computer Society.