Eurographics Workshop on Natural Phenomena (2007)
D. Ebert, S. Mérillou (Editors)

Convective Clouds

Robert Geist, Jay Steele, James Westall

School of Computing, Clemson University, Clemson, SC, USA

Abstract

A new technique for rendering convective clouds is suggested. The technique uses two lattice-Boltzmann (LB) models, one for generating the spatial and temporal distribution of water density and the other for photon transport, that is, lighting the water density with correct anisotropic scattering. The common LB structure is easily mapped to parallel execution environments such as a GPU or multiple CPUs connected via the Message Passing Interface (MPI), thereby providing sub-minute execution times on commodity hardware.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Rendering clouds from three-dimensional models has become an integral task for visual productions ranging from feature films to the evening weather report. When those models are physically-based, they offer the potential for integration into larger, weather prediction systems such as the commonly used Weather Research and Forecasting model (WRF) [MDG*05].

A complete model naturally comprises two components: a water density model, used to generate the spatial and temporal distribution of both vapor and condensate, and a photon transport model, used to capture the anisotropic (principally forward) scattering of photon streams through the medium. Most of these models are expressed as coupled systems of partial differential equations (PDEs) which are solved with variations on conventional (finite-element, finite-difference) techniques. An exception is the lighting model proposed by [GRWS04], which is based on the lattice-Boltzmann (LB) technique.

Lattice-Boltzmann methods are computational alternatives to finite-element methods, and they have provided significant successes in modeling fluid flows and associated transport phenomena [AR93, CD98, SD95]. The methods simulate transport by tracing the evolution of a single particle distribution through synchronous updates on a discrete grid. They provide stability, accuracy, and computational efficiency comparable to finite-element methods, but they realize significant advantages in ease of implementation, parallelization, and an ability to handle inter-facial dynamics and complex boundaries.

The principal drawback to the methods is the counter-intuitive direction of the derivation they require. Differential equations describing the macroscopic system behavior are derived from a postulated computational update, rather than the reverse. Thus a relatively simple computational method must be justified by a relatively intricate derivation.

The purpose of this paper is to demonstrate that the same LB technique used for lighting clouds [GRWS04] can be used to generate their vapor density and temperature distributions. To this end, we draw upon the multi-component, LB model due to Shan and Doolen [SD95], Shan's simulation of Rayleigh-Bénard convection [Sha97], and the fundamental structure used in the lighting model [GRWS04].

In this three-dimensional, two-component model, the two components are water vapor density, sometimes called absolute humidity, and temperature. The key quantity of interest is the per-component *directional density*, $f_{\sigma,i}(\vec{r},t)$, which is the density of component σ arriving at lattice site $\vec{r} \in \Re^3$ at time t in direction $\vec{c}_i$. The directions $\vec{c}_i$, i = 0,1,...,18, are all the non-corner lattice points of a cube of unit radius, $\{-1,0,1\}^3$. If λ is the lattice spacing and τ is the time step of our simulation, then the entire lattice Boltzmann computation is just an iterated, synchronous update of the directional

densities according to:

$$f_{\sigma,i}(\vec{r}+\lambda\vec{c}_i, t+\tau) = f_{\sigma,i}(\vec{r},t) + [\Omega_\sigma(f_\sigma)]_i \qquad (1)$$

where $\Omega_\sigma : \Re^{19} \rightarrow \Re^{19}$ is a linear (affine) collision operator. The simplicity of the required computation (1) somewhat belies the underlying complexity of the limiting case. We can show that the limiting case is, in fact, a multi-component, Navier-Stokes equation that includes both conventional external forces (e.g. gravity) and those arising from component interactions.

Although our approach does not provide the real-time rendering of the best techniques currently available [HL01, HISL03], it does provide a stronger physical basis, images of high quality, and reasonable execution times for grids of relatively high resolution (128^3 nodes or larger).

2. Related Work

Both the generation of cloud densities and the lighting of participating media have been studied extensively in the literature. [Kv84] addressed both in their foundational work. In generating cloud densities they used a coupled system of PDEs that represented air velocity, temperature, and water mixing ratios, i.e., the mass of vapor/liquid per unit mass of air. The governing equation was the Euler equation of hydrodynamics, which is just the Navier-Stokes equation without the dissipative effects of viscosity. The velocity field was assumed to be divergence-free, and pressure was not explicitly modeled. In the related problem of modeling flows of hot gases, [FM97] make a convincing case for use of the full Navier-Stokes equations. They employ a finite difference scheme for solution. Restricting their simulations to smoke, [FSJ01] suggest that the Euler equations will suffice for visual accuracy and are computationally less intensive. A principal contribution of this work is the vorticity confinement correction, which adds back fine-scale flow detail that is damped out by coarse-grid solvers. [WLMK04] use a lattice-Boltzmann model to simulate the dynamics of steam. They model air flow and then attach textures, constructed from photographs of steam, to particles in the flow. Attaching textures in this way can be regarded as a simple alternative to the vorticity confinement correction. [MYDN01] use a coupled map lattice simulation of the Navier-Stokes equation for air flow to generate some high quality cloud images. The coupled map lattice bears some similarity to the lattice-Boltzmann approach in that both may be regarded as cellular automata. A drawback is their approximation of the substantial derivative, $D\vec{u}/dt = \partial\vec{u}/\partial t + \vec{u}\cdot\nabla\vec{u}$, by the ordinary partial, $\partial\vec{u}/\partial t$, and an unusual approximation for the pressure term. The effects of each approximation on physical accuracy are not addressed. [NR92] generate cloud densities using a percolation model. Their results offer excellent agreement with real clouds in 2D (view from below) fractal analysis, but the model does not generate macroscopic cloud structure. The approach of [HISL03] provides both high quality images and excellent execution time. Their model is similar to that of [Kv84], but they draw on models of [Hou93] in including both positive and negative buoyancy effects. They solve using fragment programs on GPUs.

Radiative transfer in scattering volumes is also well-studied topic. Early approaches assumed that propagating rays encountered at most one scattering event [EP90,Sak90]. Later techniques capture multiple scattering. Analytic models of multiple scattering have invariably lead to diffusion processes. [Kv84] used spherical harmonics to expand both the light intensity field and the scattering phase function. They obtained a coupled set of partial differential equations in the spherical harmonic coefficients whose solution would yield intensity at each spatial coordinate. [Sta95] observed that an approximate solution was a diffusion process and provided substantial detail regarding this process, including a suggested multi-grid solution technique. [JMLH01] showed that a simple, two-term approximation of radiance naturally leads to a diffusion approximation that is appropriate for a highly scattering medium. As noted earlier, we will use the lighting model of [GRWS04], where it is shown that the limiting process $(\lambda,\tau \rightarrow 0)$ for the update given by (1) can be described by the standard diffusion equation

$$\frac{\partial\rho}{\partial t} = D\nabla^2\rho \qquad (2)$$

with diffusion coefficient

$$D = \left(\frac{\lambda^2}{\tau}\right)\left[\frac{(2/\sigma_t)-1}{4(1+\sigma_a)}\right]$$

which is completely determined by the absorption and scattering coefficients, σ_a and σ_t.

3. A Multi-Component, Water Density Model

We now describe our water density model and sketch the derivation of the associated Navier-Stokes equation for the limiting $(\lambda,\tau \rightarrow 0)$ case. Although we will ultimately restrict to two components, vapor density and temperature, much of the derivation will apply to an arbitrary number of components. The full details of the derivation may be found in the online technical report [Gei06].

As noted earlier, the key quantity of interest will be the per-component *directional density*, $f_{\sigma,i}(\vec{r},t)$, which is the density of component σ arriving at lattice site $\vec{r} \in \Re^3$ at time t in direction $\vec{c}_i$, and the directions $\vec{c}_i$, i = 0,1,...,18, are the non-corner lattice points of a cube of unit radius, $\{-1,0,1\}^3$. We take $\vec{c}_0 = (0,0,0)$, and $\vec{c}_1$ through $\vec{c}_6$ to be the axis directions. Note that these directions are really projections from 4D space of 24 lattice points that are equidistant from the 4D origin,

$$\begin{array}{ll} (\pm1,0,0,\pm1) & (0,\pm1,\pm1,0) \\ (0,\pm1,0,\pm1) & (\pm1,0,\pm1,0) \\ (0,0,\pm1,\pm1) & (\pm1,\pm1,0,0) \end{array}$$

where the projection is truncation of the fourth component. The equidistant points allow for isotropic flow, but the projection to 3D means that the axial directions will carry double weight in the discussions below.

Component density per site is $\rho_\sigma(\vec{r},t) = \sum_{i=0}^{18} f_{\sigma,i}(\vec{r},t)$, and total density per site is $\rho(\vec{r},t) = \sum_\sigma \rho_\sigma(\vec{r},t)$. If we take unit velocity $v = \lambda/\tau$, and $\vec{v}_i = v\vec{c}_i$, then component velocity per site is $\vec{u_\sigma}(\vec{r},t) = (\sum_{i=0}^{18} f_{\sigma,i}(\vec{r},t)\vec{v}_i)/\rho_\sigma(\vec{r},t)$.

The fundamental, synchronous update (1) obviously depends upon the structure of the affine operator, Ω. For lighting, [GRWS04] used a simple, 19×19 matrix whose entries were determined by the absorption and scattering coefficients, the scattering phase function, and the cloud density at the site. Any operator must satisfy:

$$\sum_{i=0}^{18} [\Omega_\sigma(f_\sigma)]_i = 0 \qquad \text{conservation of mass} \tag{3}$$

$$\sum_\sigma \sum_{i=0}^{18} [\Omega_\sigma(f_\sigma)]_i \vec{v}_i = 0 \quad \text{conservation of momentum} \tag{4}$$

If external force $\vec{F}_\sigma(\vec{r},t)$ is applied to component σ, then instead of (4) we must have:

$$\sum_\sigma \sum_{i=0}^{18} [\Omega_\sigma(f_\sigma)]_i \vec{v}_i = \tau \sum_\sigma \vec{F}_\sigma(\vec{r},t) \tag{5}$$

Note that if there is no net momentum flux at the boundaries, then momentum of the entire system is still conserved.

Many collision operators satisfy these constraints. When direct control over individual collision events is unimportant, a convenient operator is the LBGK operator [BEK54]:

$$[\Omega_\sigma(f_\sigma)]_i = -\frac{1}{\xi_\sigma}\left[f_{\sigma,i}(\vec{r},t) - f_{\sigma,i}^{(eq)}(\vec{r},t)\right] \tag{6}$$

where ξ_σ is the *relaxation time* of the σ^{th} component (a parameter), and $f_{\sigma,i}^{(eq)}(\vec{r},t)$ is the *local equilibrium density*. It is defined, per direction per component, by $f_{\sigma,i}^{(eq)}(\vec{r},t) =$

$$\begin{array}{ll}
\rho_\sigma(d - [\vec{u_\sigma}^{(eq)}]^2/(2v^2)) & i = 0 \\
\rho_\sigma(\frac{1-d}{12} + \frac{\vec{v}_i\cdot\vec{u_\sigma}^{(eq)}}{6v^2} + \frac{\vec{v}_i\vec{v}_i:\vec{u_\sigma}^{(eq)}\vec{u_\sigma}^{(eq)}}{4v^4} - \frac{[\vec{u_\sigma}^{(eq)}]^2}{12v^2}) & i = 1...6 \\
\rho_\sigma(\frac{1-d}{24} + \frac{\vec{v}_i\cdot\vec{u_\sigma}^{(eq)}}{12v^2} + \frac{\vec{v}_i\vec{v}_i:\vec{u_\sigma}^{(eq)}\vec{u_\sigma}^{(eq)}}{8v^4} - \frac{[\vec{u_\sigma}^{(eq)}]^2}{24v^2}) & i = 7...18
\end{array}$$

where $d \in [0,1]$ is a parameter denoting the desired fraction of density with zero speed at equilibrium, ':' denotes the scalar product of tensors, and $\vec{u_\sigma}^{(eq)}$ is defined so that (4) or (5) holds. Specifically, if we use elementary identities on the weighted direction sums, e.g., $\sum_{i=1}^{6} 2v_{i\alpha} + \sum_{i=7}^{18} v_{i\alpha} = 0$ $\alpha \in \{x,y,z\}$, then it is easy to verify that

$$\sum_i f_{\sigma,i}^{(eq)} = \rho_\sigma \tag{7}$$

and

$$\sum_i \vec{v}_i f_{\sigma,i}^{(eq)} = \rho_\sigma \vec{u_\sigma}^{(eq)} \tag{8}$$

regardless of the definition of $\vec{u_\sigma}^{(eq)}$. To enforce constraint (4) we would then need

$$\begin{aligned} 0 &= \sum_\sigma \sum_i -\frac{\vec{v}_i}{\xi_\sigma}\left[f_{\sigma,i} - f_{\sigma,i}^{(eq)}\right] \\ &= -\sum_\sigma \frac{\rho_\sigma \vec{u_\sigma}}{\xi_\sigma} + \sum_\sigma \frac{\rho_\sigma \vec{u_\sigma}^{(eq)}}{\xi_\sigma} \end{aligned}$$

In the absence of external forces, we choose to make all $\vec{u_\sigma}^{(eq)}$s equal, i.e., independent of σ. Thus we are led to the definition:

$$\vec{u_\sigma}^{(eq)} = \vec{u}^{(eq)} = \left(\sum_\sigma \frac{\rho_\sigma \vec{u_\sigma}}{\xi_\sigma}\right) / \left(\sum_\sigma \frac{\rho_\sigma}{\xi_\sigma}\right) \tag{9}$$

In the presence of external forces, we instead define

$$\vec{u_\sigma}^{(eq)} = \vec{u}^{(eq)} + \frac{\xi_\sigma \tau}{\rho_\sigma}\vec{F}_\sigma \tag{10}$$

which guarantees that (5) holds. The principal motivation for the choice (6) is that it is computationally fast and will lead to the Navier-Stokes equation at the macroscopic level.

We have yet to define an overall, component-independent velocity, $\vec{u}$. This is a matter of choice (within reason), since there is no apriori-correct weighting for the components. We observe that total momentum at a site before a collision is $\sum_\sigma \rho_\sigma \vec{u_\sigma}$ and total momentum after the collision is $\sum_\sigma \rho_\sigma \vec{u_\sigma} + \tau \sum_\sigma \vec{F}_\sigma$. If we want $\rho\vec{u}$ to match the cross-collisional average, we must have

$$\vec{u} = \left(\sum_\sigma \rho_\sigma \vec{u_\sigma} + \frac{\tau}{2}\sum_\sigma \vec{F}_\sigma\right)/\rho \tag{11}$$

Although we have yet to specify the parameter values, external forces, and initial conditions used in our computation, it is worth noting that the procedure is otherwise complete: we synchronously update all nodes in the lattice using (1) where Ω is given by (6).

3.1. Macroscopic System Behavior

The standard flow equations (continuity, Euler, Navier-Stokes) can now be derived directly from the specified computational update (1), although the full derivation is long and arduous. Here we provide only a sketch. For the full details, the reader is referred to [Gei06].

We use the so-called *Chapman-Enskog expansion*, standard in lattice-Boltzmann modeling. (See [CD98].) We assume that $f_{\sigma,i}$ can be written as a small perturbation about some local equilibrium, $f_{\sigma,i}^{(0)}$:

$$f_{\sigma,i} = f_{\sigma,i}^{(0)} + \varepsilon f_{\sigma,i}^{(1)} \tag{12}$$

where ε is the *Knudsen number*, which represents the mean free path between successive collisions.

The choice of $f^{(0)}$ is not necessarily unique. The constraints are that it carries the density and the momentum, specifically:

$$\sum_i f^{(0)}_{\sigma,i} = \rho_\sigma \tag{13}$$

$$\sum_i \vec{v}_i f^{(0)}_{\sigma,i} = \rho_\sigma \vec{u} \tag{14}$$

From (7) and (8), it is easy to find a suitable choice for $f^{(0)}_{\sigma,i}$: use $f^{(eq)}_{\sigma,i}$ and replace every instance of $\vec{u_\sigma}^{(eq)}$ with $\vec{u}$.

We also consider system behavior at multiple time scales as both lattice spacing and time step approach 0. We partition the time scale

$$t = K\frac{t_0}{\varepsilon} + (1-K)\frac{t_1}{\varepsilon^2} \tag{15}$$

where $t_0 = o(\varepsilon)$, $t_1 = o(\varepsilon^2)$, and $K \in [0,1]$. Similarly, we write distance

$$\vec{r} = \frac{\vec{r_0}}{\varepsilon} \tag{16}$$

where $\vec{r_0} = o(\varepsilon)$. Note that the relationship among the partials is given by:

$$\begin{aligned} \frac{\partial}{\partial t} &= \varepsilon\frac{\partial}{\partial t_0} + \varepsilon^2\frac{\partial}{\partial t_1} \\ \frac{\partial}{\partial r_\alpha} &= \varepsilon\frac{\partial}{\partial r_{0\alpha}} \qquad \text{for} \quad \alpha \in \{x,y,z\} \end{aligned} \tag{17}$$

All of the standard flow equations now result from inserting (12) and (17) into a two-variable Taylor expansion of (1) and equating coefficients of like powers of ε.

In particular, we obtain the standard continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\vec{u}) = 0 \tag{18}$$

The Navier-Stokes equation includes a pressure term, and so we need an appropriate definition. We assume that, for those external forces, $\vec{F_\sigma}$, that contribute to pressure (typically, all component interactions but not gravity), we can find a *potential*, i.e., a function V with the property that $\nabla V = -\sum_\sigma \vec{F_\sigma}$. We then define pressure as

$$p = v^2\left(\frac{1-d}{2}\right)\rho + V \tag{19}$$

so that

$$\nabla p = v^2\left(\frac{1-d}{2}\right)\nabla\rho - \sum_\sigma \vec{F_\sigma} \tag{20}$$

With this definition, we can then derive the Navier-Stokes equation:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u} = -(1/\rho)\nabla p + \sum_\sigma(\rho_\sigma/\rho)\vec{g}_\sigma - \sum_\sigma \mu_\sigma\nabla^2(\rho_\sigma\vec{u}) \tag{21}$$

where component viscosity, $\mu_\sigma = \frac{\tau v^2}{6\rho}(1-2\xi_\sigma)$. Thus, unlike other treatments, we are not limited to the Euler equation as a macroscopic description of system behavior.

Nevertheless, the most interesting behavior arises from an attempt to establish a per-component version of the continuity equation. It turns out that individual components do not satisfy the standard continuity equation with respect to the composite velocity. Instead, we can derive

$$\begin{aligned} \frac{\partial\rho_\sigma}{\partial t} &= -\nabla\cdot(\rho_\sigma\vec{u}) - \xi_\sigma\tau\nabla\cdot\vec{F_\sigma} \\ &+ (\tau\xi_\sigma - \tau/2)\nabla\cdot\left[-(\rho_\sigma/\rho)\nabla p + v^2\left(\frac{1-d}{2}\right)\nabla\rho_\sigma\right] \\ &+ \tau\nabla\cdot\frac{\rho_\sigma}{\rho}\left[\frac{1}{2}\sum_\sigma\vec{F_\sigma} + \sum_\sigma\xi_\sigma\vec{F_\sigma}\right. \\ &+ \left.\frac{\nabla p}{\rho}\sum_\sigma\xi_\sigma\rho_\sigma - v^2\left(\frac{1-d}{2}\right)\sum_\sigma\xi_\sigma\nabla\rho_\sigma\right] \end{aligned} \tag{22}$$

and from this other important model properties will follow, as described below.

3.2. Interaction Forces

We assume the interaction potential, V, is of the form:

$$V = (1/2)\sum_{\sigma_1}\sum_{\sigma_2} G_{\sigma_1,\sigma_2}\Psi_{\sigma_1}(\rho_{\sigma_1})\Psi_{\sigma_2}(\rho_{\sigma_2}) \tag{23}$$

where $G_{\sigma_1,\sigma_2} = G_{\sigma_2,\sigma_1}$ is a symmetric *strength of interaction* and Ψ_{σ_i} is an *effective density*. We can then take

$$\vec{F_{\sigma_1}} = -\Psi_{\sigma_1}(\rho_{\sigma_1})\sum_{\sigma_2} G_{\sigma_1,\sigma_2}\Psi'_{\sigma_2}(\rho_{\sigma_2})\nabla\rho_{\sigma_2} \tag{24}$$

so that $\nabla V = -\sum_\sigma \vec{F_\sigma}$, as required.

To include external forces that are not interactions, e.g., gravity, we write instead

$$\vec{F_{\sigma_i}} = -\Psi_{\sigma_i}(\rho_{\sigma_i})\sum_{\sigma_j} G_{\sigma_i,\sigma_j}\Psi'_{\sigma_j}(\rho_{\sigma_j})\nabla\rho_{\sigma_j} + \rho_{\sigma_i}\vec{g_{\sigma_i}} \tag{25}$$

where $\vec{g_{\sigma_i}}$ carries the non-interactive external force on component σ_i. Now $\nabla V = -\sum_\sigma \vec{F_\sigma} + \sum_\sigma \rho_\sigma\vec{g_\sigma}$, and so we need to correct (20) and subsequent expressions involving ∇p, in particular (22), by replacing ∇p with $\nabla p - \sum_\sigma \rho_\sigma\vec{g_\sigma}$ wherever it occurs.

3.3. Other Model Properties

In addition to satisfying a multi-component Navier-Stokes equation, the model has other important properties that distinguish it from previous treatments.

3.3.1. Diffusion of Thermal Energy

In his simulation of Rayleigh-Bénard convection, [Sha97] argues that when viscous and compressive heating effects can be neglected, temperature can be modeled as a separate component whose molecular mass is (relatively) 0. Assume

we have only two components where the first is vapor density and the second is temperature. To simplify notation, assume $\xi_1 = \xi_2 = \xi$. Then from (22) and (20) we have

$$\frac{\partial \rho_2}{\partial t} + \nabla \cdot (\rho_2 \vec{u}) = \nabla \cdot \left[D \left[\frac{\rho_1}{\rho} \nabla \rho_2 - \frac{\rho_2}{\rho} \nabla \rho_1 \right] + \tau \xi \left[\frac{\rho_2}{\rho} \vec{F}_1 - \frac{\rho_1}{\rho} \vec{F}_2 \right] \right] \quad (26)$$

where $D = \tau(\xi - 1/2)v^2 \left(\frac{1-d}{2} \right)$.

If we now use (24) and assume that $G_{i,j} = 0$ for $i \neq j$, we can collect coefficients of density gradients to obtain

$$\frac{\partial \rho_2}{\partial t} + \nabla \cdot (\rho_2 \vec{u}) = \nabla \cdot \left[\frac{\rho_1}{\rho} \left(D + \tau \xi \Psi_2 G_{2,2} \Psi_2' \right) \nabla \rho_2 - \frac{\rho_2}{\rho} \left(D + \tau \xi \Psi_1 G_{1,1} \Psi_1' \right) \nabla \rho_1 \right] \quad (27)$$

If the relative densities approach limits, $\rho_1/\rho \to 1$ and $\rho_2/\rho \to 0$, and we also have $G_{2,2} = 0$, we get

$$\frac{\partial \rho_2}{\partial t} + \nabla \cdot (\rho_2 \vec{u}) = D \nabla^2 \rho_2 \quad (28)$$

Thus temperature is both advected and diffused.

3.3.2. Phase Transition

For the same two components, assume $G_{\sigma_1,\sigma_2} = 0$, except for $G_{1,1}$, so $V = G_{1,1} \Psi_1^2(\rho_1)$. From (19) we have

$$p = v^2 \left(\frac{1-d}{2} \right) (\rho_1 + \rho_2) + G_{1,1} \Psi_1^2(\rho_1) \quad (29)$$

If $G_{1,1}$ is negative and Ψ_1 is increasing and bounded, there can be a range (of ρ_1) over which $dp/d\rho_1$ is negative, which signals a phase transition.

[Sha97] suggests $\Psi_1(\rho_1) = \rho_{1_0}(1 - e^{-\rho_1/\rho_{1_0}})$. With this choice of Ψ_1, we can solve $dp/d\rho_1 = 0$ for interval bounds:

$$\rho_1 = -\rho_{1_0} ln \left[\frac{1 \pm \sqrt{1 + \frac{v^2(1-d)}{G_{1,1}\rho_{1_0}}}}{2} \right] \quad (30)$$

To normalize this, we take

$$G_{1,1} = -v^2 \left(\frac{1-d}{\rho_{1_0}} \right) (1 + \rho_{2_0} - \rho_2) \quad (31)$$

where $\rho_2 \leq \rho_{2_0}$. The value ρ_{1_0} represents maximum effective water vapor density (bound on Ψ_1), and ρ_{2_0} represents maximum temperature at which a phase transition can take place. We then have $dp/d\rho_2 > 0$, and the interval over which $dp/d\rho_1$ is negative increases as temperature (ρ_2) decreases.

3.3.3. Latent Heat

Positive values of $G_{1,2} = G_{2,1}$ can be shown to effect an increase/decrease in temperature during the phase transitions of condensation/evaporation that typically accompany entrainment (mixing). Consider a local region wherein the velocity field is divergence-free ($\nabla \cdot \vec{u} = 0$), there is negligible spatial thermal gradient ($\nabla \rho_2 = 0$), and yet there is a sharp spatial transition in water density from vapor to condensed water. Again assuming $G_{2,2} = 0$, we could reduce (26) to

$$\frac{\partial \rho_2}{\partial t} = \tau \xi \nabla \cdot \left[\Psi_2(\rho_2) G_{1,2} \Psi_1'(\rho_1) \nabla \rho_1 \right] \quad (32)$$

Thus, if $G_{1,2}$ is positive, we should see an accompanying local increase in temperature.

4. Rendering

Rendering is a four-stage pipeline: water density generation, percolation, lighting, and ray-casting. We use a lattice with 128^3 nodes, which offers a relatively high resolution without exorbitant memory requirements. For this lattice, each stage of the pipeline can be executed in less than one minute on commodity hardware.

4.1. Water Density Generation

We use the transient solution of the model of section 3 to generate macroscopic cloud structure. For the parameters, initial conditions, and boundary conditions we have chosen (below), interesting cloud structures begin to emerge after approximately 10 iterations of (1).

Initial conditions are designed to force the rapid emergence of a convective cloud. The lattice cube is filled with vapor density and a linear temperature gradient is applied in the vertical direction. The initial distribution of vapor density is centered on *silos*, which are cylinders with hemi-spherical caps that are placed inside the cube with bases on the cube floor. The intent is to simulate a snapshot of a convective bubble [RY89]. Each silo is filled with vapor density that is uniformly distributed over $[0.99\rho_{1_0}, 1.01\rho_{1_0}]$. Outside the silo, but inside the cube, the density falls off linearly with distance from the central axis or the center of the hemisphere.

The effective density, $\Psi_1(\rho_1)$, and component $G_{1,1}$ are as specified in section (3.3.2) with model parameters $\rho_{1_0} = 1.0$, $\rho_{2_0} = 0.505$, $d = 0.1$, $\lambda = 1.0$, $\tau = 1.0$, and (hence) $v = 1.0$. The fixed temperature at the top of the cube is 0.495. Relaxation times are $\xi_1 = 1.01$, $\xi_2 = 2.00$. Smaller values than these lead to instability, and larger values delay emergence of interesting structure.

We also take $\Psi_2(\rho_2) = \rho_2$, $G_{1,2} = G_{2,1} = 0.5$, $G_{2,2} = 0.0$, and $g_1(\rho_2) = 0.01(\rho_2 - 0.5)$, which represents the net effect of gravity and buoyancy. This expression follows from the so-called Boussinesq approximation of density, common in studying natural convection [Sha97]. It is given by $\rho/\rho_0 = 1 + \beta(T - T_0)$, where ρ_0 and T_0 are density and temperature at a reference point, and β is the constant thermal expansion coefficient.

Boundary conditions are toroidal, except in the vertical direction, where we fix temperature at the top and bottom of the cube to the initial values. We also fix the component velocities at the top and bottom, $\vec{u_1} = \vec{u_2} = 0$, by equalizing directional densities. This adds significantly to the stability of the system.

4.2. Percolation

Although we use a relatively high resolution and simulate a full Navier-Stokes equation, our model is not immune to the damping effects of finite grid simulation as described by [FSJ01], for which the vorticity confinement correction was devised. We prefer an alternative approach to restoring fine detail to the flow. We regard the output of the water density model as macroscopic cloud structure and use it as a collection of masks for humidity seeding the input to the percolation model of [NR92]. Because the Nagel-Rascke model is bit-oriented, we first expand the original 128^3 grid by a factor of K in each dimension. The binary output is then averaged over each cube of edge dimension K to provide a real density with resolution $1/K^3$ at each of the original 128^3 sites. In all the examples shown, we used $K = 5$.

The water density model carries a condensate threshold of ρ_{1_0}, and so sites with densities less than $\rho_{1_0} - \varepsilon_1$ are not seeded, i.e., all corresponding K^3 sites are marked "off". For sites with densities greater than $\rho_{1_0} + \varepsilon_2$, all corresponding K^3 sites are marked "on". A linear scaling is used for sites with densities in between. With the parameters of section 4.1, we use $\varepsilon_1 = 0.00025$ and $\varepsilon_2 = 0.0002$.

4.3. Lighting

We use the model of [GRWS04], but we observe that the 4D density arrays used there (3 spatial indices, i, j, k, and 1 directional, m) can be folded into a 2D, RGBA float texture

$$(i, j, k, m) \rightarrow ((k\%(E/4)) * E + i, j * M + m)[k/(E/4)]$$

where edge size $E = 128$ and $M = 19$. The model can then be executed as a collection of fragment programs on a GPU, for which we witness more than a 60-fold reduction in execution time compared to the results reported in [GRWS04]. We can also use NVIDIA's Compute Unified Device Architecture (CUDA) library and encode the model directly.

4.4. Ray-casting

Here we use a conventional, volumetric compositing approach to visualization of lighted volume densities, which amounts to integration along rays cast from the viewer's eye through image pixels into the volume density.

5. Results

The cloud images shown in Figure 1 were generated from stochastically identical initial conditions, described in section 4.1, with four silos of varying radii. The only difference between the two images was the single random seed supplied to the model. Obviously, the initial conditions exert a strong influence on the final cloud structure. In Figure 2 we show images obtained from a grid of dimension 256×128^2 with eight silos. Again, the only difference between the two images of Figure 2 is the initial random seed supplied to the model. Note that the effects are approximately additive. The four silos used in Figure 1 appear on the left in Figure 2.

Similarly, all of the clouds in Figure 3 are synthetic, as is the cloud of Figure 4. Cloud shadows in Figure 3 were generated by using backward (ground to sun) orthographic projections of the lighted clouds to create 2D textures and then compositing these into the scene photo.

Execution times are reasonable for each stage of the pipeline. The data structures are ideal for parallel updates. The water density generation, percolation, lighting, and ray-casting stages can all be mapped to fragment programs for execution on a GPU. We used CUDA on an NVIDIA G80 Quadro for all stages. For the clouds of Figure 1, the lighting stage averaged 18 seconds. The percolation stage averaged 40 seconds. Unlike lighting and percolation, which require steady-state solution, the water density generation is a transient solution, for which the number of iterations is somewhat arbitrary. The images of Figure 1 required 0.26 seconds per iteration. We find interesting structure emerges within 10 - 20 iterations. Ray-casting (640×480) required 3 seconds.

A short, animated fly-through of one our clouds may be seen at `http://www.fx.clemson.edu/~jesteel/cloud_circle.mov`.

6. Conclusions

We have provided a new technique for rendering convective clouds. We use two models, one for generating the distribution of water density and the other for photon transport. The two models share a common structure in that they are both based on a lattice-Boltzmann formulation, and this structure allows convenient parallel execution. The principal contribution here is the water density model, which yields high quality images in reasonable execution time. The model also provides a stronger physical basis than other techniques in that its limiting case satisfies a full, multi-component Navier-Stokes equation, it offers both advection and diffusion of thermal energy, and it naturally models the phase transitions of condensation and evaporation.

We have focused exclusively on convective clouds. Whether our water model will be of value in rendering clouds formed by other processes, e.g., stratus clouds, remains an open question.

7. Acknowledgments

This work was supported in part by the CISE Directorate of the U.S. National Science Foundation under award EIA-

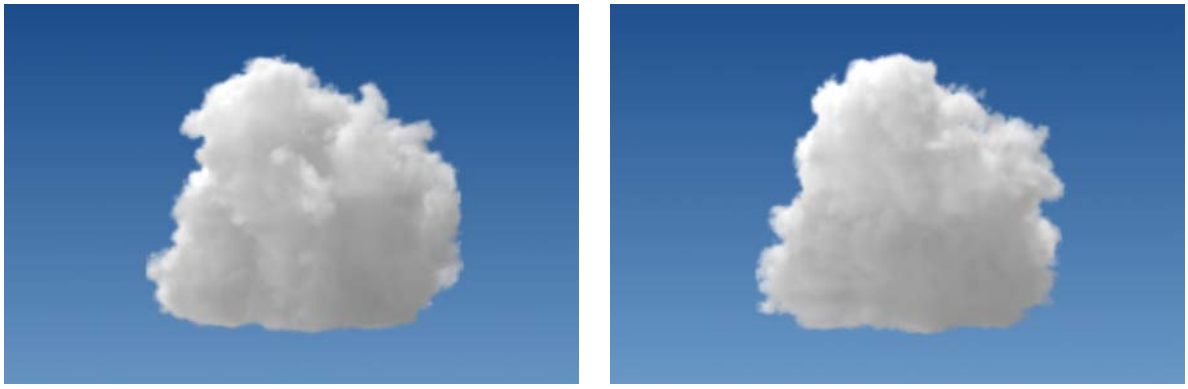

Figure 1: *Synthetic clouds generated from a* 128^3 *grid with 4 silos of varying radii.*

Figure 2: *Synthetic clouds generated from a* 256×128^2 *grid with 8 silos of varying radii.*

Figure 3: *View from Rich Mountain, Brevard, NC*

Figure 4: *Echo Canyon, north of Abiquiu, NM*

0305318 and by a Fellowship grant from NVIDIA Corp. Thanks to Xiaowen Shan for many useful discussions.

References

[AR93] AHARONOV E., ROTHMAN D.: Non-newtonian flow through porous media: A lattice-boltzmann method. *Geophysical Res. Letters 20* (April 1993), 679–682. 1

[BEK54] BHATNAGAR P., E.GROSS, KROOK M.: A model for collision processes in gases. *Phys. Rev. 94* (1954), 511–525. 3

[CD98] CHOPARD B., DROZ M.: *Cellular Automata Modeling of Physical Systems*. Cambridge Univ. Press, Cambridge, UK, 1998. 1, 3

[EP90] EBERT D., PARENT R.: Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. *ACM Computer Graphics (SIGGRAPH '90) 24*, 4 (August 1990), 357–366. 2

[FM97] FOSTER N., METAXAS D.: Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 181–188. 2

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 15–22. 2, 6

[Gei06] GEIST R.: A three-dimensional, multi-component, lattice-boltzmann model with component interaction. http://www.cs.clemson.edu/~rmg/clouds_tr.pdf, 2006. 2, 3

[GRWS04] GEIST R., RASCHE K., WESTALL J., SCHALKOFF R.: Lattice-boltzmann lighting. In *Rendering Techniques 2004 (Proc. Eurographics Symposium on Rendering)* (Norrkőping, Sweden, June 2004), pp. 355 – 362,423. 1, 2, 3, 6

[HISL03] HARRIS M., III W. B., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *Graphics Hardware* (2003). 2

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud rendering. In *EG 2001 Proceedings* (2001), Chalmers A., Rhyne T.-M., (Eds.), vol. 20(3), Blackwell Publishing, pp. 76–84. 2

[Hou93] HOUZE R.: *Cloud Dynamics*. Academic Press, San Diego, CA, 1993. 2

[JMLH01] JENSEN H., MARSCHNER S., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001* (August 2001), pp. 511–518. 2

[Kv84] KAJIYA J., VON HERZEN B.: Ray tracing volume densities. *ACM Computer Graphics (SIGGRAPH '84) 18*, 3 (July 1984), 165–174. 2

[MDG*05] MICHALAKES J., DUDHIA J., GILL D., HENDERSON T., KLEMP J., SKAMAROCK W., WANG W.: The weather reseach and forecast model: Software architecture and performance. In *Proceedings of the* 11^{th} *ECMWF Workshop on the Use of High Performance Computing In Meteorology* (2005), Zwieflhofer W., Mozdzynski G., (Eds.), World Scientific, pp. 156 – 168. 1

[MYDN01] MIYAZAKI R., YOSHIDA S., DOBASHI Y., NISHITA T.: A method for modeling clouds based on atmospheric fluid dynamics. In *Proc. Ninth Pacific Conference on Computer Graphics and Applications (PG'01)* (Tokyo, Japan, October 2001), pp. 363 – 372. 2

[NR92] NAGEL K., RASCHKE E.: Self-organizing criticality in cloud formation? *Physica A 182* (1992), 519–531. 2, 6

[RY89] ROGERS R., YAU M.: *A Short Course in Cloud Physics*. Pergamon Press, Oxford, England, 1989. 5

[Sak90] SAKAS G.: Fast rendering of arbitrary distributed volume densities. In *Proc. Eurographics '90* (September 1990), pp. 519–530. 2

[SD95] SHAN X., DOOLEN G.: Multicomponent lattice-boltzmann model with interparticle interaction. *J. of Statistical Physics 81*, 1/2 (1995), 379–393. 1

[Sha97] SHAN X.: Simulation of rayleigh-bénard convection using a lattice boltzmann method. *Physical Review E 55*, 3 (1997), 2780–2788. 1, 4, 5

[Sta95] STAM J.: Multiple scattering as a diffusion process. In *Proc.* 6^{th} *Eurographics Workshop on Rendering* (Dublin, Ireland, June 1995), pp. 51–58. 2

[WLMK04] WEI X., LI W., MUELLER K., KAUFMAN A.: The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004), pp. 164-176. 2