Flow Simulation with Locally-Refined LBM

Ye Zhao *
Kent State University

Feng Qiu
Stony Brook University

Zhe Fan
Stony Brook University

Arie Kaufman
Stony Brook University

Abstract

We simulate 3D fluid flow by a locally-refined lattice Boltzmann method (LBM) on graphics hardware. A low resolution LBM simulation running on a coarse grid models global flow behavior of the entire domain with low consumption of computational resources. For regions of interest where small visual details are desired, LBM simulations are performed on fine grids, which are separate grids superposed on the coarse one. The flow properties on boundaries of the fine grids are determined by the global simulation on the coarse grid. Thus, the locally refined fine-grid simulations follow the global fluid behavior, and model the desired small-scale and turbulent flow motion with their denser numerical discretization. A fine grid can be initiated and terminated at any time while the global simulation is running. It can also move inside the domain with a moving object to capture small-scale vortices caused by the object. Besides the performance improvement due to the adaptive simulation, the locally-refined LBM is suitable for acceleration on contemporary graphics hardware (GPU), since it involves only local and linear computations. Therefore, our approach achieves fast and adaptive 3D flow simulation for computer games and other interactive applications.

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

Keywords: Lattice Boltzmann model, Flow simulation, Local refinement, Embedded grids, Interactive fluid simulation, GPU

1 Introduction

The application of CFD (Computational Fluid Dynamics) methods for solving the Navier-Stokes equations has led to significant developments in the visual simulation of gases and fluids. However, the unpredictability and complexity of the fluid dynamics in nature, combined with a large-scale 3D modeling environment, make it extremely difficult to simultaneously model correct fluid behaviors through numerical simulation, create abundant visual details and satisfy the critical performance requirement. Researchers have presented several approaches to overcome this problem from different directions [Klingner et al. 2006; Losasso et al. 2004; Mueller et al. 2003; Premoze et al. 2003; Rasmussen et al. 2003; Shah et al. 2004]. These methods are often ingenious, but they are rarely generalizable, and it is typically infeasible to compose more complex simulations by their combination. The booming development of modern graphics hardware has also encouraged researchers to utilize the graphics processing unit (GPU) for accelerating flow simulations [Bolz et al. 2003; Goodnight et al. 2003; Harris et al. 2003; Harris et al. 2002; Krueger and Westermann 2003]. While achiev-

ing improved performance for 2D flows or simple 3D flows, these methods have not solved complicated 3D flow simulations with arbitrarily-shaped internal objects.

In this paper, we develop a visual flow simulation system that models sufficient flow details in dynamically-changing regions of interest, runs fast, operates on contemporary consumer hardware, and at the same time is easy to build, use and maintain, in comparison to other types of GPU-accelerated adaptive solvers. We achieve these with a level-of-detail scheme and by utilizing a hardware-accelerated multiple-grid lattice Boltzmann method (LBM). Consequently, it can be easily included in applications that require adaptive simulation and fast performance, such as computer games and virtual environments.

The LBM has developed into a viable CFD method and a promising numerical scheme for simulating fluid flow and modeling physics in fluids [Succi 2001]. It has the advantage of being easy to implement and especially suitable for GPU acceleration. The scheme is particularly successful in fluid flow applications involving complex boundaries such as arbitrarily-shaped objects and moving objects. The fundamental idea of the LBM is to construct simplified kinetic models that incorporate the essential physics of microscopic processes so that the macroscopic averaged properties obey the desired macroscopic Navier-Stokes equations. Therefore, a natural advantage of the LBM is that no discretization of the macroscopic continuum equations has to be provided, as opposed to conventional numerical schemes. As a result, LBM does not need to explicitly consider the distribution of pressure on interfaces of refined grids because it is implicitly included in the computational scheme. We provide an overview of the LBM in the Appendix.

For a large-scale 3D simulation with physically-based methods, it is inefficient to maintain a uniform high-resolution grid spanning the entire domain. In this paper, we use an adaptive computational structure consisting of a coarse grid and one or more fine grids. The global flow behavior in the whole simulation space is roughly modeled by the LBM on the coarse grid. For regions of interest, separate LBM simulations run on the corresponding fine grids superposed on the coarse one. The fine-grid simulations are passively driven by the coarse grid, which has two benefits: (1) a fine grid is easily initiated and terminated at any time while the global simulation is running; (2) a fine grid is able to move inside the domain along with a moving object, to model small-scale turbulence caused by the object-fluid interactions. Moreover, such embedded grids retain the inherent parallelizability of the LBM computational scheme, so that we can accelerate the multiple-grid LBM on the GPU. In recapitulation, our main contribution in this paper is to achieve fast and interactive 3D fluid simulation by an adaptive modeling scheme and straightforward graphics hardware acceleration based on an explicit fluid solver (LBM). It has great potential for entertainment, educational and scientific applications.

2 Previous Work

Over the past decades, a variety of approaches have been published for physically-based modeling of fluid phenomena with turbulent motions and complex visual effects [Enright et al. 2002; Fedkiw et al. 2001; Foster and Metaxas 1996; Goktekin et al. 2004; Nguyen et al. 2002; Selle et al. 2005; Stam 1999; Stam 2003]. The fluid solvers usually consume a large amount of memory and computation time due to the discretization of the large-scale simulation

---

domain. Researchers have presented several approaches to tackle this problem, such as representing fluids with particles [Premoze et al. 2003], generating a 3D flow field from simpler 2D simulations [Rasmussen et al. 2003], refining the simulation domain with an octree-based multi-resolution scheme [Losasso et al. 2004], adaptive simulation cells culling [Shah et al. 2004], and dynamic tetrahedral simulation meshes [Klingner et al. 2006].

In computer graphics, LBM has been used in simulating a variety of fluid phenomena, such as ink dispersion [Chu and Tai 2005], free-surface fluid [Thurey and Rude 2004], floating objects [Wei et al. 2004], flows on the surface [Fan et al. 2005], and heat shimmering and mirage [Zhao et al. 2007]. In physics, the LBM can handle adaptive mesh refinement (AMR) with locally embedded grids [Dupuis and Chopard 2003; Filippova and Hänel 1998]. The methods with embedded grids can handle varying levels of resolution over a simulation volume using multiple LBM grids with different resolutions, instead of tree-based recursive structures (e.g., octree [Losasso et al. 2004]). In this paper, we adopt this approach and develop it for computer graphics applications. It maintains the locally-refined lattice structure with multiple grids, while changing their density, so that the computational parallelizability is still retained.

3   Multiple Grid Structure

An overview of the LBM computation on a uniform grid has been provided in the Appendix. Instead of applying one uniform grid with a high resolution in the whole space, our approach optimizes the use of computational resources with multiple LBM grids. A coarse grid covering the entire simulation domain models the global flow behavior. For a region of interest (e.g., near an internal obstacle or user specified), a fine grid is superposed. Such local refinement strategy models small-scale flow details in the crucial regions by using denser computational grids.

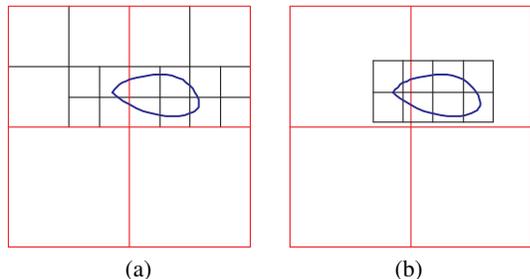(a)   (b)

Figure 1: 2D local grid refinement enclosing a blue object: The coarse grid is marked in red, while the high resolution fine grid is marked in black: (a) Quadtree; (b) Embedded grid.

In Figure 1, a special 2D region surrounds a blue object immersed in a fluid and interacts with the flow, where local refinement is desired. Figure 1a shows how to refine the grid with a quadtree subdivision of the space. In comparison, Figure 1b uses a high-resolution fine grid superposed on the coarse grid. Each of these grids performs a separate LBM computation with interactions on their interfaces. The particle distributions, f_i, of the grid sites sitting on the interfaces are defined by the values computed by the coarse grid. Thus, the flow simulation on the fine grid follows global fluid behavior. The fine grid can be superposed on the coarse grid at any arbitrary position, and with a resolution that is a reasonable multiple of the coarse one. For the communication between different grids, we use the global position of their rectangular bounding boxes to transform positions between their local coordinates and the global coordinates. Furthermore, a fine grid can even move inside the simulation domain together with a moving object to capture its

turbulent vortices. Unlike the tree-based structure, such a dynamic grid refinement does not require complex operations on updating or rebuilding the recursive data structure. Therefore, the computational overhead of grid communication is minimal and feasible in GPU-based simulations for computer games and other interactive applications.

A fine grid is currently positioned at the object center with user-defined box size to enclose the object. For a moving object, the box moves together with the center position of the object. The size of a fine grid represents the size of a region of interest. We plan to further improve our system by exploring automatic refinement criteria defined by flow properties, such as obstacle boundaries and vorticity concentration [Losasso et al. 2004], local Reynolds number, strain tensor magnitude, etc.

4   Computational Procedure

4.1   Temporal Interpolation

The time step size of an LBM simulation is partially defined by and directly proportional to the grid spacing, which is the spacing between two neighboring grid sites. For LBM simulation, the computations on the coarse grid and on the fine grid have to be synchronized. When the simulation proceeds with one large time step on the coarse grid, the fine grid has to perform the computation with several smaller steps to advance its simulation to the same time.

Figure 2 illustrates the computational procedure of our LBM simulation on a coarse grid and a fine grid. At first, the global simulation on the coarse grid is running with a large time step t_c from starting time T_1 to time T_3. When the fine grid simulation starts at time T_2, it runs with a smaller time step t_f (t_c is a multiple of t_f). Figure 2 depicts the procedure with $t_c = 2t_f$. The coarse grid computation does not provide the values of the particle distributions (f_i), on the interfaces at time T_2 due to its large time step size. Therefore, we perform a temporal interpolation to compute f_i at time T_2 from the global computation results on the coarse grid at times T_1 and T_3 (see the red arrows). Such temporal interpolation is also computed at time T_4 and so on, which can be implemented by a linear scheme as

$$\hat{f}_i^c(\vec{x}, T_2) = (1 - \frac{t_f}{t_c})f_i^c(\vec{x}, T_1) + \frac{t_f}{t_c}f_i^c(\vec{x}, T_3), \qquad (1)$$

where the hat represents the interpolating results, $\vec{x}$ is the spatial position, and the superscript c means that the value is computed on the coarse grid. When $t_c \neq 2t_f$, the temporal interpolation can be implemented in a similar manner.

4.2   Spatial Interpolation

After the temporal interpolation, we have computed the particle distributions of grid sites on the coarse grid at time T_2. As illustrated in Figure 3, now we have $\hat{f}_i^c(O, T_2)$, $\hat{f}_i^c(P, T_2)$, $\hat{f}_i^c(Q, T_2)$ and $\hat{f}_i^c(R, T_2)$ at the grid sites O, P, Q and R, respectively. Next, we execute a spatial interpolation to compute particle distributions of the non-overlapped grid sites on the fine grid, such as A and B, from the known values. For example, for the site A, $\hat{f}_i^c(A, T_2)$ is computed by

$$\hat{f}_i^c(A, T_2) = Interp(\hat{f}_i^c(O, T_2), \hat{f}_i^c(P, T_2), \hat{f}_i^c(Q, T_2), \hat{f}_i^c(R, T_2)). \ (2)$$

The $Interp()$ is a linear interpolation function (trilinear in 3D case). The $\hat{f}_i^c(A, T_2)$ is obtained by the LBM computation of the coarse grid and the value cannot be directly used in the LBM computation on the fine grid. In the next section, we describe how to transform this value $\hat{f}_i^c(A, T_2)$ to a proper fine grid value $f_i^f(A, T_2)$, to define the boundary condition of the fine grid.
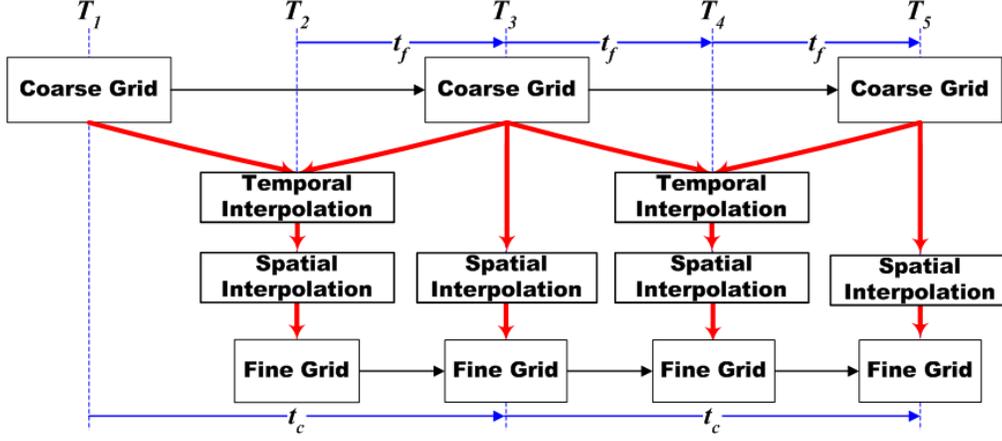
Figure 2: Computational procedure of a coarse grid and a fine grid. $t_c = 2t_f$, where t_c is the time step of the coarse grid and t_f is that of the fine grid. T_1 to T_5 represent five fine grid time steps.
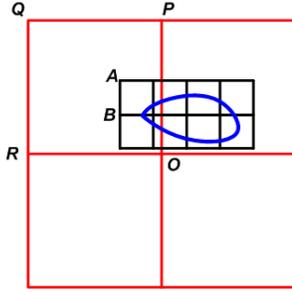

Figure 3: Spatial interpolation at time T_2. The particle distribution values at coarse grid sites O, P, Q and R are used to compute the particle distributions at the boundary sites A on the fine grid by interpolation (bilinear interpolation in this 2D case).

4.3   Boundary Value Transformation

The relaxation time and the packet distributions of the fine grid have to be recomputed from the values of the coarse grid simulation, because the two LBM computations have different configurations, such as grid spacing and time step size. A high resolution grid has a smaller grid spacing, δr_f, than that of the coarse grid, δr_c. In order to keep the global velocity $\vec{u}$ consistent on the different grids, $\frac{\delta r}{\delta t}$ has to be kept as a constant for both of them. At the same time, the viscosity of the fluid has to be constant between the grids. Then, the relaxation time (see Equation 9), τ_f, of the fine grid is computed as

$$\tau_f = \frac{\delta r_c}{\delta r_f}(\tau_c - \frac{1}{2}) + \frac{1}{2}, \qquad (3)$$

where τ_c is the relaxation time of the coarse grid.

By decomposing the particle distribution f_i into equilibrium and nonequilibrium parts and considering the change of τ for the coarse and fine grids, the transformation between the f_is are derived (see [Filippova and Hänel 1998] for theoretical details). The f_i on the interface sites of the fine grid are calculated from the coarse grid by

$$f_i^f = \hat{f}_i^{eq,c} + (\hat{f}_i^c - \hat{f}_i^{eq,c})\frac{\delta r_f \tau_f}{\delta r_c \tau_c}. \qquad (4)$$

The $\hat{f}_i^c$ and $\hat{f}_i^{eq,c}$ with a hat here are the coarse grid simulation results after temporal and spatial interpolations. The temporal and spatial interpolations, as well as the fine grid boundary computations, are all linear and local operations, and therefore, convenient for GPU acceleration.

4.4   Discussion

The LBM grid refinement algorithm [Dupuis and Chopard 2003] with two-way coupling has been validated for its physical correctness, if we connect the two grid scales by passing on simulation results from the fine to the coarse and vice versa. In our current implementation, there is only one-way coupling from the coarse grid to the fine grids, while the fine simulation does not feedback to the global simulation. Adopting this strategy makes our algorithm not physically correct in regions of interest. However, it simplifies the computation and GPU implementation with visually pleasing results. Moreover, the initiation, termination and movement of the fine grids can be performed at any time step during the simulation, since these operations do not affect the global simulation. Such detail-enhancing method is similar to the advected textures [Neyret 2003], which generates very high resolution fluid appearance based on correct low resolution CFD with the use of Kolmogorov's energy cascade theory. In comparison, we use LBM based fine grid simulation instead of the noise-based textures to provide small details in regions of interest. In the future, we will implement two-way coupling simulations with GPU accelerations to make our method totally physically-correct. The main task would be keeping the adaptive simulation benefits, while the hardware performance is not affected too much.

5   GPU Acceleration

The key for our method to achieve fast speed is to combine the efficient adaptive LBM method and the acceleration technique of non-graphics computation on modern GPU [Owens et al. 2005], a powerful, low cost, highly data-parallel processor. In recent years, graphics hardware has seen a booming development. The graphics pipeline has become programmable with high-level programming language support. This has propelled researchers to adventure in using the GPU for accelerating flow simulations [Bolz et al. 2003; Goodnight et al. 2003; Harris et al. 2003; Krueger and Westermann 2003]. They 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 cannot efficiently simulate complicated 3D flows interacting with arbitrarily-shaped internal obstacles. In contrast, the LBM is superior for GPU acceleration in such a case. Single-resolution LBM acceleration has been presented on a single GPU [Li et al. 2005] or a GPU cluster [Fan et al. 2004]. However, these implementations do not include grid refinement, which is the focus of this paper.

The challenge in implementing the multiple-grid LBM on the GPU is that there are several LBM grids with different grid densities. This motivates us to use an object-oriented technique in the framework of our implementation on the GPU. We implement Li et al.'s [2005] method for each grid and encapsulate the texture data and computation kernels in a class. For each LBM grid, the grid data are stored in 32 bit floating point textures by arranging slices of the 3D grid into a 2D atlas. The computation kernels are written in Cg fragment programs and executed by OpenGL commands of rendering rectangles into the textures, which are renderable when we attach them to Framebuffer Objects (FBO). Then, we design the interface kernels as a member function for the finer grids to access the data of the coarse grid and compute its own boundary values using interpolations. With this abstraction, we can easily deploy as many fine grids as necessary at any position in the simulation domain.

Data communications are performed on the boundaries between a coarse and a fine grid, which uses the simulation results of the coarse grid to compute the boundary values of the fine grid. The particle distributions, f_i^f, on the boundaries are computed in three steps: (1) temporal interpolation between different large time steps of the coarse grid (Equation 1); (2) spatial interpolation for the boundary nodes (Equation 2); (3) refinement transformation from the coarse grid interpolation results to the values used in the fine grid simulation (Equation 4).
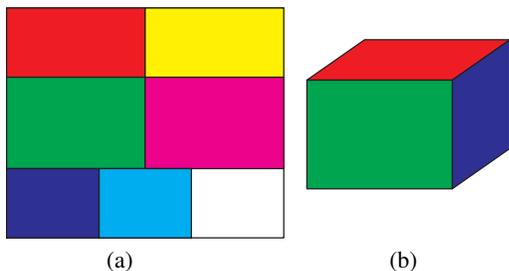

(a)   (b)

Figure 4: The boundary texture. Boundaries of the fine grid (b) are flattened and packed into a 2D texture (a). Each color block in the 2D texture represents a face of the boundary box painted with the same color.

One optimization approach is to perform all the computations in the three steps only on the boundary nodes, since in most cases, these nodes are just a small portion of all the grid nodes. Therefore, the spatial interpolation is first applied on a box of six rectangular faces representing the boundaries of the fine grid at a particular time. As shown in Figure 4, these boundary faces are flattened and packed into a 2D boundary texture. In the GPU pipeline, each of the six regions in the boundary texture is updated by drawing a rectangle of the corresponding region. The vertices of these rectangles store their positions in the coordinate of the coarse grid. The fragment program uses them to locate the position of each fragment in the coarse grid LBM textures, fetch corresponding neighboring values, and perform trilinear interpolation to compute the particle distribution values of particular positions, such as A and B in Figure 3. For a moving fine grid, the positions stored in its boundary texture have to be updated at each step, which will slightly increase the overhead compared with static refinement.

After performing spatial interpolation separately on two consecutive steps of the coarse grid (e.g., at the time T_3 and T_5 in Figure 2), we compute by temporal interpolation the values of fine grid boundary nodes (e.g., A, B) at in between times (e.g., T_4) to a new boundary texture, required for the fine grid simulation. Now, the computation is performed only between the boundary textures. Thus, we avoid the temporal interpolation for the whole coarse grid by moving the spatial interpolation ahead. This is possible because the temporal and spatial interpolations are commutative in the adaptive LBM computational procedure.

Finally, the results of these interpolations in the new boundary texture are transformed by the refinement computation, and then used to modify the corresponding LBM texture nodes of the fine grid, enabling its ordinary LBM simulation. In general, the overhead of the grid communications compared with the conventional LBM computation is minimal.

6 Rendering

Our LBM-based simulation computes the flow field in a 3D simulation domain. For example, when a fluid source (e.g., a smoke inlet) is placed in the domain and begins to release smoke, the smoke densities of all the grid nodes construct a scalar volumetric dataset. The evolution of this density volume is modeled by an advection-diffusion equation and computed by a back-tracing algorithm (see Appendix A in [Stam 1999]). We further use the monotonic cubic interpolation [Fedkiw et al. 2001] for computing the back-tracing density values at positions off the regular grid sites. This high-order interpolation scheme slows the generation of the density volume, however, it fixes the over-shooting problem of the trilinear interpolation method and provides clearer visual details. The velocities used in the back-tracing algorithm are chosen from multiple grids. Obviously, the velocities from the fine grids have higher priority than those from the coarse grid. Consequently, the smoke density values in regions of interest are computed and stored in a high-resolution volume, which provides more details of the smoke dispersion. However, using this approach, the rendering results may show some unnatural edge effects on the grid boundaries, observed from some angles. Such interface artifacts arise from the varying levels of resolution both in the multiple grid modeling and rendering. We have minimized such artifacts by interpolating fluid density around boundaries to achieve visually satisfied results. Better smoothing methods or adding some small noise may further improve the results. Introducing backward coupling from fine grids to the coarse grid will further tackle this problem in the modeling stage.

The density volume generated in the simulation is rendered using a volumetric ray tracing method with single scattering. The algorithm has two steps: (1) A lighting volume is generated that stores the light intensity of each voxel; (2) One ray is cast from each pixel on the image plane to compute the color of that pixel. In order to achieve interactive speed, it is imperative to accelerate the rendering process with the GPU. The lighting volume is constructed facing the light source so that the direction of the light is perpendicular to every slice of the lighting volume. The first slice is initialized with the intensity of the light source. Every other slice is calculated by attenuating the intensities of the previous slice with opacities defined by the density values interpolated from the density volume. In the ray-casting step, each ray starts from a front face of the volume bounding box and stops at either a back face or the surface of the polygonal objects inside the volume. A face of the bounding box is a front (back) face if and only if the dot product of its normal and the view direction is less (greater) than 0. To calculate the termination points, the polygonal objects and the back faces are projected onto the image plane with depth test. This method computes the depth information of the possible ray termination points. When a ray is cast into the volume, color and opacity
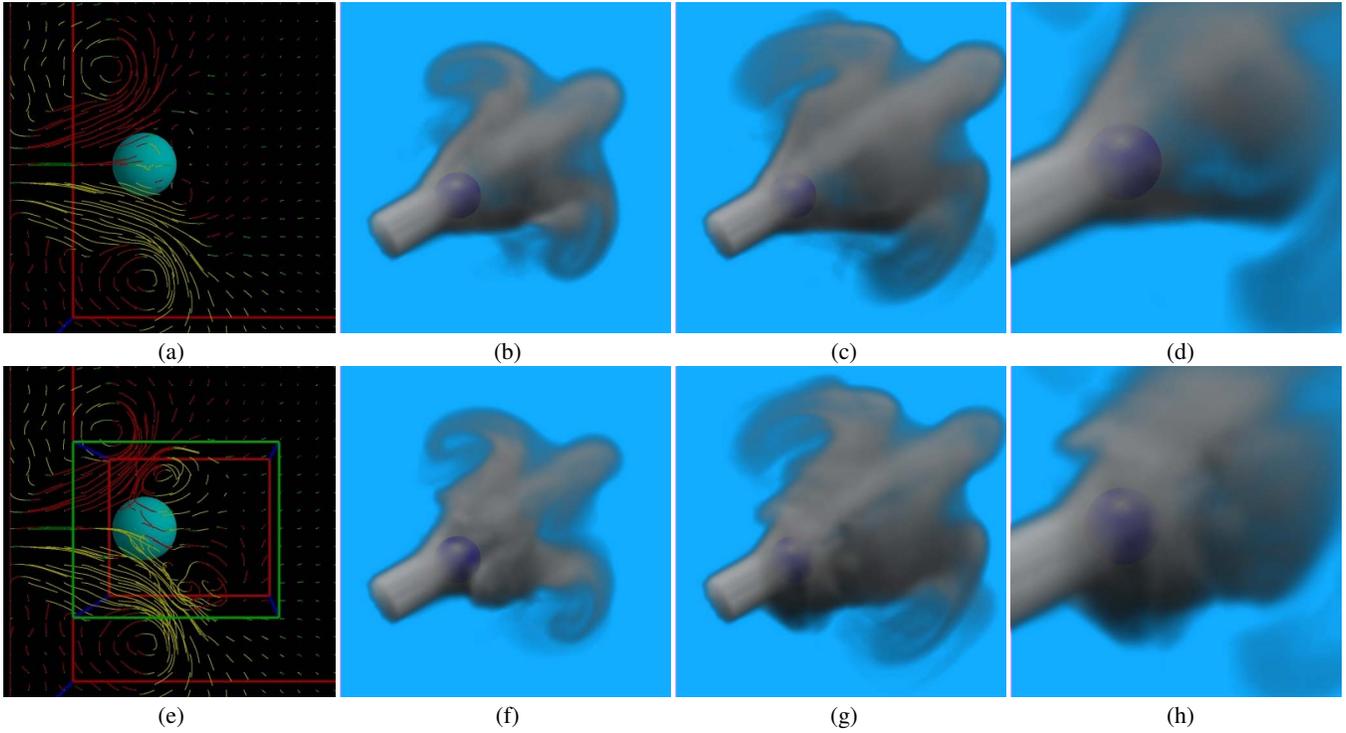
Figure 5: (a)-(d) Smoke passing a static sphere; (d) A zoom-in view of (c); (e)-(h) Simulation of smoke with a fine grid surrounding the sphere, superposed on the coarse grid; (h) A zoom-in view of (g).
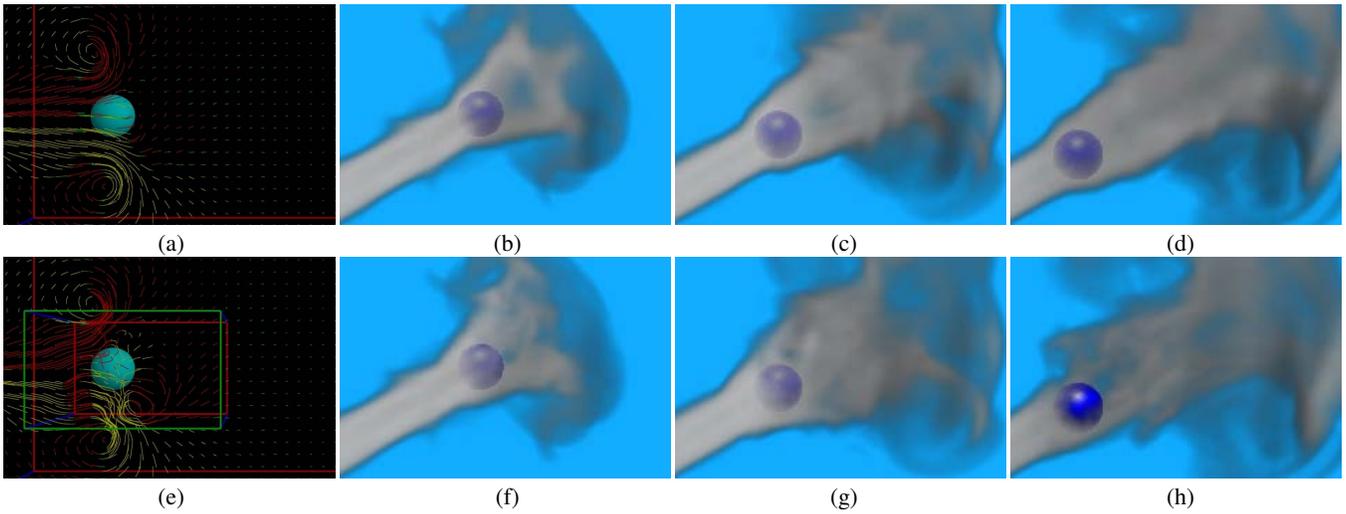

Figure 6: (a)-(d) Smoke passing a sphere moving towards the smoke inlet; (e)-(h) Smoke with a fine grid moving along with the sphere, superposed on the coarse grid, .

Table 1: Performance Results.

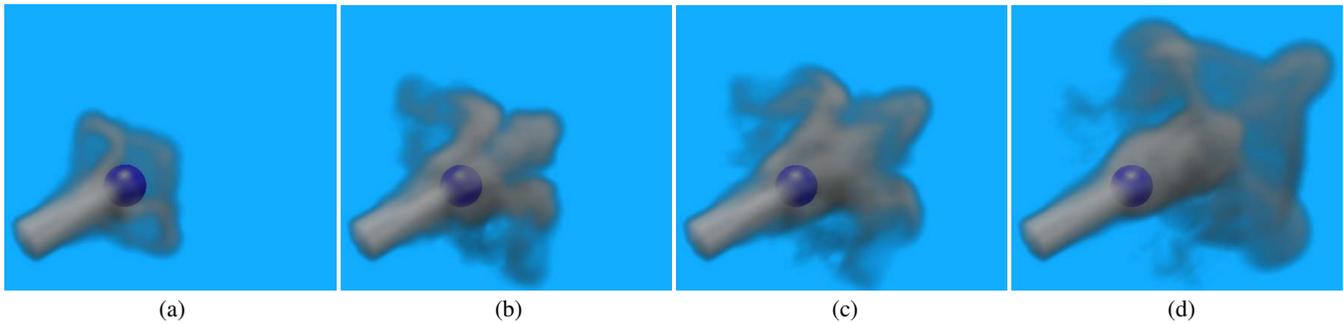| Simulation Example	Coarse Grid Resolution	Fine Grid Resolution	LBM Simulation CPU : GPU (msec)	Simulation Frame Rate CPU : GPU (frames/sec)	Smoke Generation CPU : GPU (msec)	GPU Rendering (msec)	Total Frame Rate (frames/sec)
Fig. 5	$41 \times 41 \times 41$	$35 \times 33 \times 33$	584 : 21	1.7 : 47.6	2435 : 41	52	8.8
Fig. 6	$50 \times 25 \times 25$	$34 \times 20 \times 20$	209 : 15	4.8 : 66.7	1311 : 20	42	13.0
Fig. 7	$25 \times 25 \times 25$	$22 \times 19 \times 19$	125 : 10	8 : 100	886 : 15	36	16.4

185

Figure 7: Smoke passing a sphere using a coarse grid: (a) At first, no fine grid is used; (b)-(c) A fine grid is initiated to model small-scale details around the sphere; (d) The fine grid is terminated and small-scale details disappear.

values are accumulated at each sampling point, where the lighting intensity is interpolated from the lighting volume for shading.

Besides the flow simulations, we have also implemented the back-tracing algorithm, the monotonic cubic interpolation and single-scattering ray tracing on the GPU. Therefore, no read-backs to the CPU are needed, which is the bottleneck in current graphics hardware. If rendering speed can be compromised in some applications, our fluid density volumes are ready to generate even more realistic visualization results with multi-scattering ray tracing, photon-mapping and other advanced approaches.

7   Results

We have used our adaptive modeling method in several examples to show its benefits in interactive 3D fluid simulations. In this paper, we illustrate the examples with several snapshots and the whole simulations are animated in the associated movies.

Figure 5 illustrates our simulation of smoke passing a static sphere. We use a $41 \times 41 \times 41$ coarse grid to simulate the flow in the simulation domain and Figures 5a-5d are the rendering results. The smoke is quiescent and no small-scale details are visible. Specifically, streamlines from each grid site on a cutting plane illustrate the flow field in Figure 5a, where red color means the velocity is going up and yellow means going down. In Figures 5e-5h, a fine grid is used in the region surrounding the sphere, which has a $35 \times 33 \times 33$ resolution. For comparison, each image is generated at the same simulation time as its counterpart in Figures 5a-5d. Small-scale vortices are modeled and the smoke shows more visual details. Figures 5d and 5h show the zoom-in view of Figures 5c and 5g, respectively.

Figure 6 shows that our adaptive simulation can easily handle moving objects. The sphere moves towards the smoke inlet. A $50 \times 25 \times 25$ coarse grid covers the simulation space and a $34 \times 20 \times 20$ fine grid moves together with the sphere. In Figures 6e-6h, rich visual details of the smoke are modeled near the sphere, compared with Figures 6a-6d, where only the global coarse grid is used.

In Figure 7, we show the dynamic initiation and termination of a fine grid. At first (Figure 7a), a $25 \times 25 \times 25$ coarse grid is used for the simulation. Then, a $22 \times 19 \times 19$ fine grid is initiated to model small-scale details surrounding the sphere and thus the smoke turbulent behavior appears (Figures 7b-7c). After we terminate the fine grid in the middle of the simulation, the smoke details near the sphere disappear (Figures 7d).

8   Performance

We achieve fast computational speed for 3D flow simulation by GPU acceleration and simulation with local refinement on multiple grids with different resolutions.

GPU Acceleration: The three examples are implemented both on a CPU and a GPU for comparison. For the CPU version, our simula-

tion and smoke generation are timed on one 3.0GHz Pentium Xeon CPU with 1GB memory. We accelerate the simulation, smoke generation and the rendering on a GPU, nVIDIA Quadro FX 4500. Table 1 shows the performance of our examples. We report the time (in millisecond) used in adaptive LBM computation, smoke density volume evolution for the same configuration and results on the CPU and the GPU respectively, while both are not fully optimized.

Local Refinement: For Figure 5, our LBM computation runs in 47.6 frames per second for a moderate $41 \times 41 \times 41$ coarse grid and a $35 \times 33 \times 33$ fine grid. For comparison, we use a uniform grid with an effective resolution of $82 \times 82 \times 82$ to generate similar results. This uniform resolution LBM runs 3.1 times slower and consumes 5.2 times more texture memory (140 MB) than our adaptive method (27 MB).

Together with the GPU implementations of the back-tracing method and the monotonic cubic interpolation for smoke generation, as well as the ray-casting rendering, we have accomplished all the computations with hardware acceleration and achieved the interactive simulation speed (8.8 frames/second for Figure 5, 13.0 frames/second for Figure 6, and 16.4 frames/second for Figure 7). Here one frame corresponds to one simulation time step, and we generate a 512×512 image for each frame on the GPU, and avoids transferring simulation results back to the main memory.

9   Conclusions and Future Work

We have adopted a level-of-detail scheme to simulate 3D fluid dynamics based on the locally-refined LBM. The global flow behavior is simulated on a coarse grid with low consumption of computational resources. At the same time, small-scale visual details in each region of interest are modeled using a fine grid superposed on the coarse grid. Our LBM-based adaptive simulation, as well as the rendering, are accelerated on the GPU. Our 3D fluid simulation achieves interactive performance and has great potential in games, movies and other interactive applications.

In the future, we plan to improve our simulation with two-way coupling between the coarse and fine grids, and multiple levels of grid resolutions. We will also improve our system with artifacts elimination, automatic fine grid setup and interactive user interface.

Appendix: Lattice Boltzmann Method

LBM is a relatively new approach in CFD that models Boltzmann particle dynamics on a 3D lattice (grid). It is a microscopically-inspired method designed to solve macroscopic fluid dynamics problems. It lives at the interface between the microscopic (molecular) and macroscopic (continuum) worlds, hopefully capturing the best of both. The Boltzmann equation expresses the variation of the average number of microscopic flow particles moving with a given velocity between each pair of neighboring sites. Such variation is caused by inter-particle interactions and ballistic motion

of the particles. The variables associated with each grid site are the particle distributions that represent the probability of a particle presence with a given velocity. Particles stream synchronously along links from each site to its neighbors in discrete time steps and perform collision between consecutive streaming steps. The Bhatnager, Gross, Krook (BGK) model is used to model collisions as a statistical redistribution of momentum which locally drives the system toward equilibrium while conserving mass and momentum. The LBM is second-order accurate both in time and space, and in the limit of zero time step and grid spacing, it yields the Navier-Stokes equation for an incompressible fluid [He and Luo 1997] with low Mach number (i.e., velocities are small compared to the speed of sound).
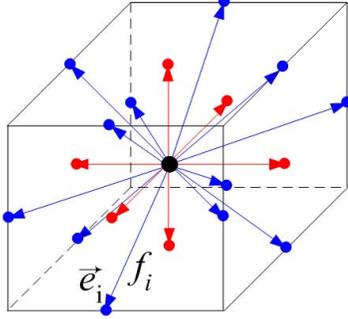

Figure 8: D3Q19 LBM grid. $\vec{e}_i$ is the velocity vector and f_i is the fluid particle distribution moving along the vector.

Each grid cell in a 3D grid has at most 27 links with its neighbors. A typical LBM grid structure (Figure 8) is the D3Q19 (3D with 19 links), which uses eighteen axial and minor-diagonal neighboring links for fluid computation, and including the center cell itself with a zero velocity link. Each link has its velocity vector $\vec{e}_i(\vec{x},t)$ and the particle distribution $f_i(\vec{x},t)$ that moves along this link, where $\vec{x}$ is the position of the cell and t is the time. The macroscopic fluid density, $\rho(\vec{x},t)$ and velocity $\vec{u}(\vec{x},t)$ are computed from the particle distributions as

$$\rho = \sum_i f_i \qquad \vec{u} = \frac{1}{\rho}\sum_i f_i \vec{e}_i. \tag{5}$$

Using the BGK model, the Boltzmann dynamics can be represented as a two-step process of collision and ballistic streaming. Taken together they can be represented as

$$f_i(\vec{x}+\vec{e}_i,t^+) = f_i(\vec{x},t) - \frac{1}{\tau}(f_i(\vec{x},t) - f_i^{eq}(\rho,\vec{u})), \tag{6}$$

$$f_i(\vec{x}+\vec{e}_i,t+1) = f_i(\vec{x}+\vec{e}_i,t^+), \tag{7}$$

where the local equilibrium particle distribution is given by (BGK model)

$$f_i^{eq}(\rho,\vec{u}) = \rho(A + B(\vec{e}_i \cdot \vec{u}) + C(\vec{e}_i \cdot \vec{u})^2 + D\vec{u}^2). \tag{8}$$

In the LBM Equations 5 to 8, A to D are constant scalar coefficients and $\vec{e}_i$ are constant vectors specific to the chosen grid geometry. The constant τ represents the relaxation time determining the kinematic viscosity ν of the fluid:

$$\nu = \frac{1}{3}(\tau - \frac{1}{2}). \tag{9}$$

The equilibrium distribution is a local distribution (Equation 8) whose value depends only on conserved quantities – mass ρ and

momentum $\rho\vec{u}$. The calculation of Equations 5 to 7 requires only local and neighboring properties. Therefore, all the LBM computations can be efficiently accelerated on the GPU.

Body forces can be added to the LBM as an external input to control the flow behavior. In our system, a vorticity confinement force $\vec{F}$ is added to further increase small-scale rolling features in regions of interest. $\vec{F}$ is computed in the same way as Fedkiw [2001] with a user controlled magnitude factor, and it is included in the LBM as

$$f_i \quad \longleftarrow \quad f_i + \frac{(2\tau - 1)}{2\tau}B(\vec{F} \cdot \vec{e}_i). \tag{10}$$

Interactions between an LBM flow field and an immersed object result from the exchange of momentum at their shared boundaries. The treatment of such boundaries and the boundaries of the simulation domain are handled by modifying the discrete packet distributions after a general streaming step. The full gamut of boundary conditions have been developed for the LBM including periodic, no-slip, free-slip, frictional slip, sliding walls, and open inlets and outlets [Succi 2001]. This includes simple geometries where the boundary is aligned with the grid, complex boundaries such as curves and porous media, and moving boundaries representing dynamic objects. The exchange of momentum at the boundary between a fluid and a free object immersed in it can further be used to deform or move the object [Wei et al. 2004].

Acknowledgement

References

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖDER, P. 2003. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *Proceedings of SIGGRAPH*, 917–924.

CHU, N., AND TAI, C. 2005. Moxi: real-time ink dispersion in absorbent paper. *Proceedings of SIGGRAPH*, 504–511.

DUPUIS, A., AND CHOPARD, B. 2003. Theory and applications of an alternative lattice Boltzmann grid refinement algorithm. *Physical Review E 67*, 066707.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *Proceedings of SIGGRAPH*, 736–744.

FAN, Z., QIU, F., KAUFMAN, A., AND YOAKUM-STOVER, S. 2004. GPU cluster for high performance computing. *Proceedings of ACM/IEEE Supercomputing Conference*, 47.

FAN, Z., ZHAO, Y., KAUFMAN, A., AND HE, Y. 2005. Adapted unstructured LBM for flow simulation on curved surfaces. *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 245–254.

FEDKIW, R., STAM, J., AND JENSEN, H. 2001. Visual simulation of smoke. *Proceedings of SIGGRAPH*, 15–22.

FILIPPOVA, O., AND HÄNEL, D. 1998. Grid refinement for lattice-BGK models. *J. of Computational Physics 147*, 219–228.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5, 471–483.

GOKTEKIN, T., BARGTEIL, A., AND O'BRIEN, J. 2004. A method for animating viscoelastic fluids. *Proceedings of SIGGRAPH*, 463–468.

GOODNIGHT, N., WOOLLEY, C., LEWIN, G., LUEBKE, D., AND HUMPHREYS, G. 2003. A multigrid solver for boundary value problems using programmable graphics hardware. *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware*, 102–111.

HARRIS, M., COOMBE, G., SCHEUERMANN, T., AND LASTRA, A. 2002. Physically-based visual simulation on graphics hardware. *SIGGRAPH/Eurographics Workshop on Graphics Hardware* (September), 109–118.

HARRIS, M., BAXTER, W., SCHEUERMANN, T., AND LASTRA, A. 2003. Simulation of cloud dynamics on graphics harware. *Proceedings of the ACM SIGGRAPH/Eurographics conference on Graphics Hardware*, 92–101.

HE, X., AND LUO, L. 1997. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *Journal of Statistical Physics 88*, 3/4.

KLINGNER, B., FELDMAN, B., CHENTANEZ, N., AND O'BRIEN, J. 2006. Fluid animation with dynamic meshes. *Proceedings of SIGGRAPH*, 820–825.

KRUEGER, J., AND WESTERMANN, R. 2003. Linear algebra operators for GPU implementation of numerical algorithms. *Proceedings of SIGGRAPH*, 908–916.

LI, W., FAN, Z., WEI, X., AND KAUFMAN, A. 2005. Flow simulation with complex boundaries. In *GPU Gems 2*, M. Pharr, Ed. Addison-Wesley, ch. 47, 747–764.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *Proceedings of SIGGRAPH*, 457–462.

MUELLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 154–159.

NEYRET, F. 2003. Advected textures. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 147–153.

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

OWENS, J., LUEBKE, D., GOVINDARAJU, N., HARRIS, M., KRUEGER, J., LEFOHN, A., AND PURCELL, T. 2005. A survey of general-purpose computation on graphics hardware. *Proceedings of Eurographics*, 21–51.

PREMOZE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND R.WHITAKER. 2003. Particle-based simulation of fluids. *Proceeding of Eurographics*, 401–410.

RASMUSSEN, N., NGUYEN, D., GEIGER, W., AND FEDKIW, R. 2003. Smoke simulation for large scale phenomena. *Proceedings of SIGGRAPH*, 703–707.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *Proceedings of SIGGRAPH*, 910–914.

SHAH, M., COHEN, J., PATEL, S., LEE, P., AND PIGHIN, F. 2004. Extended galilean invariance for adaptive fluid simulation. *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 213–221.

STAM, J. 1999. Stable fluids. *Proceedings of SIGGRAPH*, 121–128.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *Proceedings of SIGGRAPH*, 724–731.

SUCCI, S. 2001. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Numerical Mathematics and Scientific Computation. Oxford University Press.

THUREY, N., AND RUDE, U. 2004. Free surface lattice-Boltzmann fluid simulations with and without level sets. *Workshop on Vision, Modelling, and Visualization (VMV Stanford)*, 199–208.

WEI, X., ZHAO, Y., FAN, Z., LI, W., QIU, F., YOAKUM-STOVER, S., AND KAUFMAN, A. 2004. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics 10*, 6 (November), 719–729.

ZHAO, Y., HAN, Y., FAN, Z., QIU, F., KUO, Y.-C., KAUFMAN, A., AND MUELLER, K. 2007. Visual simulation of heat shimmering and mirage. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (Jan/Feb), 179–189.