PAPER *Special Section on Parallel and Distributed Computing and Networking*

Parallel Dynamic Cloud Rendering Method Based on Physical Cellular Automata Model

Liqiang ZHANG[†a)], ***Member***, **Chao LI**[††], ***Nonmember***, **Haoliang SUN**[††], ***Member***, **Changwen ZHENG**[†], ***and*** **Pin LV**[†], ***Nonmembers***

SUMMARY Due to the complicated composition of cloud and its disordered transformation, the rendering of cloud does not perfectly meet actual prospect by current methods. Based on physical characteristics of cloud, a physical cellular automata model of Dynamic cloud is designed according to intrinsic factor of cloud, which describes the rules of hydro-movement, deposition and accumulation and diffusion. Then a parallel computing architecture is designed to compute the large-scale data set required by the rendering of dynamical cloud, and a GPU-based ray-casting algorithm is implemented to render the cloud volume data. The experiment shows that cloud rendering method based on physical cellular automata model is very efficient and able to adequately exhibit the detail of cloud.

key words: *cellular automata model, dynamic cloud, ray-casting, GPU computing*

1. Introduction

Real-time cloud rendering and simulation is an important research area in computer graphics research. It can boost up the user's immersion and reality in the field such as aircraft view scene simulation, large spatial war simulation and movie cartoon, thus efficient and realistic cloud render has a broad prospect in practical applications.

For the complicated composition of cloud and its disordered transformation, it is a challenge to render the cloud in real time. Dobashi [1], [2] and Miyazaki [3] proposed a method to conduct real-time rendering the cloud with the dynamic produced imposter and octree. This method uses the intersection of the current viewpoint and the octree to determine the texture casting object. When the viewpoint is moving, most of the objects are reserved and the time used to produce new imposter object is saved. Ren Wei [4] proposed a method with updating strategy based on grid element, which combined the technology of volume rendering and imposter. A cloud texture object queue is constructed, and the objects in the limited view scope and distance are selected from the queue and rendered. These methods use the texture instead of the actual cloud unit to achieve high efficiency. However, there are three main disadvantages presented in the methods: First, when the simulation scale increases, massive texture objects are generated which decrease the efficiency of real-time rendering; second, with movement speed of viewpoint being fast, the rendering FPS will become unstable, which partly affects the rendering fluency. Third, the ceaseless attenuation bound with time can't be realistic rendered by these methods because the texture of cloud is static.

Manuscript received January 9, 2012.
Manuscript revised June 14, 2012.
[†]The authors are with Institute of Software of Chinese Academy of Science, China.
[††]The authors are with Graduate University of Chinese Academy of Sciences, China.
a) E-mail: zlqiang@163.com
DOI: 10.1587/transinf.E95.D.2750

Liu [5] proposed a novel simulation method in which the equations of motion was solved in a lower resolution of the grid to obtain the velocity field, then drifting noise texture is generated to increase the cloud surface detail, and GPGPU (General-Purpose computing on Graphics Processing Units) method is employed to accelerate rendering speed. Wang [6] used human-computer interaction, to roughly set bounding box of cloud by giving its shape and size, and set the attitude of the bound box to adjust the face direction of cloud, then adjust the number of the cloud particles in the box to get corresponding density of the cloud. Neyret [7] uses some heuristic rules to simulate convection features of cirrocumulus convolution effects. The above methods can acquire desired effect of cloud. However, as little modeling elements are considered, the rendering performance of realistic effect is weak, and due to the predefined shape of the dynamic cloud, users need many trials of selecting parameters to get the desired one, which limits the practical application of the algorithm. In addition, rendering the scene with lighting effects doesn't consider the physical characteristics; therefore, they cannot simulate the high spot reflection phenomenon and the high beam effect during light transmission in the cloud.

In recent years, with the rapid development of graphics hardware, some researchers have proposed the usage of graphics hardware to accelerate solving the Navier-Stokes (hereinafter referred as N-S) equations to generate the dynamic effect of cloud simulation, and the representative method, proposed by Harris et al [8], is implemented by combination 3D textures and 2D texture. This method presents bottlenecks in stratospheric cloud computing, and the resolution of adopted textures is low. There are some numerical solutions of hydrodynamic equations [9]–[11] to achieve three-dimensional cloud simulations. However, the time complexity of those methods is high, and the algorithm has some limitations on simulation effect, as it can only simulate the higher viscosity of cloud, such as rain clouds and thunder, for some other types of clouds, such as spread and cohesive, fluid accumulation process of cirrocumulus and altocumulus which does not meet the characteristics of fluid cloud, are not well simulated.

Dobashi [1] first proposed a cellular automata-based modeling of dynamic cloud, the method uses three Boolean variables for humidity, cloud and change factors to represent the cloud state property, and it defines a Boolean operation to simulate the growth of cloud, the cloud particles disappear and wind shift motion (i.e. three physical processes). The approach is focused on the simulated beam transmission phenomena in the cloud and cloud shadow, demands little on cloud detail character, and so adopts relatively simple rules of Boolean operations, but it is difficult to simulate for complex cloud movement change. Eric [12] adopt the modeling method of Dobashi [1], by combining real picture to enhance and complement rendering technology in the rendering stage. But as the model used didn't get out of the Dobashi's design restriction, the growth of cloud can be simulated, however not realistically.

We proposed the method of modeling the changing process of the cloud based on its physical characteristics, then via concept of cellular automata, dynamic cloud model can be defined as a dynamical system which evolutes in the discrete time dimension in accordance with local rules. As the key element of cellular automata system, three cellular evolution rules is modeled as fluid dynamics, deposition and accumulation, diffusion and aggregation.

Based on above, physics-based cellular automata model is adopted to simulate the dynamic cloud in three steps: First, we define cell, cell space and cell neighbors; then, cellular change rules, including fluid movement rules, sedimentation and accumulation rules, cohesion and proliferation rules are analyzed and modeled. Finally the rendering process using light projection method renders the data that have been generated. Experiments show that the method has high flexibility, can take high realistic simulation effect on all types of clouds, and the algorithm has high rendering efficiency due to the parallel computing architecture.

2. Cloud Physical Cellular Automata Model

The cloud in the atmosphere consists of a large number of small drops, which iterative with each other. The evolution process is too complex to be described by a single mathematical model. Early research pursues the visual approximation, not considering its inherent physical laws. Then more and more studies based on physical methods, which can be mainly divided into two categories: The first category was the Euler method, which studied the movement of fluid from the space occupied by the various fixed points, and analyzed pressure, density and other parameters changing over time. The second category is called Lagrangian method, which analyzed the fluid from the view of each micro-cell, and the state transforming when a fluid micelle moves to other ones, which can be consider as a particle-based method [13]. The existing methods cannot well simulate clouds' chaotic motion process. Cellular automata [14] is a discrete time and space dynamical system, which can simulate complex natural evolution, and the chaotic state in it and attractor phenomenon.

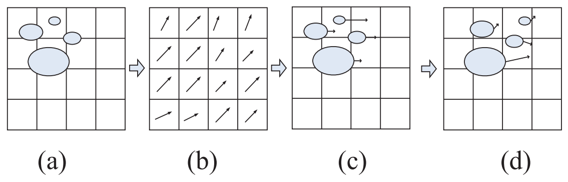

Fig. 1 Changing process of cloud micelle.

Cloud particles are generated in the process of life, integrating of diffusion and attenuation, which is related with the temperature, humidity, force field and other environmental conditions. When the body of cloud goes up for the buoyancy, cloud particles break up at certain probability. In the division process, splitting speed of the cloud particles is slow when the density of cloud micelle is large. After the splitting the cloud particles occupy more space [15]. On the contrary the larger quality of light cloud particles, the faster it splits, as Fig. 1 (a) shows. Cloud will drifts under the action of the wind effect, as Fig. 1 (b), (c). Then with time advanced and their own movement, the cloud particles are continuously generated and dispersing as shown in Fig. 1 (d). Based on the above, this modeling of the cloud takes into account three aspects as fluid dynamics, deposition and accumulation, diffusion and aggregation. These three aspects are modeled as evolution rules in cellular automata model.

Other than evolution rules, cellular automata consist of basic part of cell state set, cell space and cell neighbors. We map the cloud particles to cellular automata cell, and map the movement of particles dynamics, deposition and accumulation, diffusion and aggregation process to a variety of rules. Then based on concept of cellular automata, dynamic cloud model can be defined as a dynamical system which evolutes in the discrete time dimension in accordance with certain local rules, which consists of cellular automata with finite state in cellular space, and in accordance with certain local rules, the time dimension in the evolution of discrete dynamical systems.

Cell state set is an essential element of cellular automata, which stores the state of the change process. Cell state set is a discrete set. In the simplest case, each cell can have two states 0 and 1. In complex cases, the cell may have a number of different state variables, including location pos, size r, speed v, acceleration a and effect range r_a. The pos represents the position of the cell, and it is related to the local evolution rules. The size r represents the radius of the cell, and it's value is determined by the process of accumulation, sedimentation, diffusion and aggregation. It's important in the rendering phase for contributing the density of cloud. The moving speed v and acceleration a are the key state in fluid dynamics process.

Cellular space is discretized in each dimension to form a space grid collection. The state of the environment parameters in cellular space contains: pressure, concentration and humidity parameters. The whole target space is discretized at three dimensions. The pressure, concentration and humidity parameters are attributes of the discretized space grid to provide the environment parameters for the cellular automata motion.

Neighbor cells are the adjacent ones to itself, generally referring to cell which can have the impact to it. Takes two dimensional cellular automata for example, there are some traditional form of neighbor cell, as von Neumann type, Moore type, expansion of Moore type [14]. Considering cloud cell is in the three-dimensional space, in our model the two-dimensional neighbors' cell type extends three-dimension. We adopt the extension neighbor type of von Neumann for three dimensions.

According to the current state of its neighbor cell and the cellular space environmental parameters to determine the cell's next state, in which the transfer function of this state changing constitutes the cell's rules. The transfer function integrates all the possible states per cell and the cell's state of transformation rules, and constructs local evolution model in the space and time dimension. As a dynamic system, the cellular automata's change in the time dimension is discrete and continuous. The movement of the cloud cell has the characteristics of incompressible fluid [16], and also has the cohesive and diffusion effects and precipitation and accumulation in the evolution of cellular, thus we define the evolution rules of cloud cellular automata, including fluid dynamical movement, deposition and accumulation and diffusion and aggregation as follow.

2.1 Cellular Automata Fluid Movement Rule

Cellular automata's movement in the space meets the basic law described by the N-S equations:

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u + v\nabla^2 u - \frac{1}{\rho}\nabla P + f \tag{1}$$

Where u is the fluid velocity vector, v is the fluid dynamic viscosity, ρ is the fluid density, P represents the pressure term, and f is the force applied to the fluid. The equation for the momentum conservation equation, deduced by the Newton's second law, also meets the fluid mass conservation equation as $\nabla \cdot u = 0$.

Stam [16] divided change process of fluid state into three sub processes according to the N-S equations as force, advection, diffusion and projection process:

$$w_0(pos) \to w_1(pos) \to w_2(pos) \to w_3(pos) \to w_4(pos)$$

Here, $w_0 \sim w_4$ represent the cell velocity field in the four processes respectively.

In the action of external forces, the velocity field of cellular automata changes in this manner:

$$w_1(pos) = w_0(pos) + f \times \Delta t \tag{2}$$

Where pos represents the cell position, f is the external forces vector field such as wind force and buoyancy, after a time increment Δt, the velocity field changes from w_0 to w_1.

State inverse deducing method is adopted to get velocity field results of advection process:

$$w_2(pos_t) = w_1(pos_{t-\Delta t}) \tag{3}$$

Influence on the cell's state is not calculated by deducing directly from the time t to $t + \Delta t$, but by tracing each grid cell's track, to get the location of cellular state at time $t - \Delta t$, and then replacing the current state of cell with it.

Viscous diffusion partial differential equation which represents process of diffusion is:

$$\frac{\partial w_2}{\partial t} = v\nabla^2 w_2 \tag{4}$$

Where v is the fluid viscosity. Implicit integration method is used to solve this equation:

$$u(x,t) = u(x,t+\Delta t) - \Delta t \cdot v\nabla^2 u(x,t+\Delta t) \tag{5}$$

This equation is Poisson equation, which requires multiple iterations to get approximate result. Foter and Metaxas use relaxation iteration method to solve these equations, but it has the poor convergence of iterative, and needs many iteration times. Here, we adopt SOR (Successive Over-Relaxation) method which is a variant of the Gauss-Seidel method for solving a linear system of equations, resulting in fast convergence, to solve these equations to exhibit cloud detail accurately:

$$X_i^{(k+1)} = X_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}X_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}X_j^{(k)}\right)/a_{ii}, \tag{6}$$

X_i^{k+1} is the $k+1$ root for $i-th$ iteration, and we set relaxation factor ω to 1.5. We can see that the equation solver for SOR method is suitable for parallel calculations of cell automata updates.

After the calculation of the above three processes, we get the velocity field with divergence w_3. According to the mass conservation constraint $\nabla \cdot u = 0$ for N-S equations, the movement of fluid must meet the energy conservation law. Divergence of the velocity field has to be eliminated, that is, the projection process of cellular automata movement:

$$w_4 = w_3 - \nabla^2 q \tag{7}$$

w_4 is the destination field and has zero divergence. According to the Helmholtz-Hodge decomposition theorem we can get the scalar field q by $\nabla^2 q = \nabla \cdot w_3$. Then we can see that this equation is another Poisson equation, which can also be solved by Eq. (6).

2.2 Cellular Automata Accumulation and Sedimentation Rule

In the case of cumulus and nimbostratus, the cloud has sedimentation accumulation behavior which can be described by cellular automata sand pile model [17]. One cellular cell is represented by a grain of sand. With a grain of sand below the sand, there is one sand on its bottom left and right sides (Fig. 2 left), the sand is stable. In the other side, it will

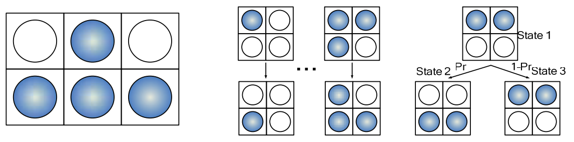

Fig. 2 Cellular automata sand pile model.

turn into other states at a certain probability (Fig. 2 middle). The sand will be converted from state 1 to state 2 with probability P_r, or with to another state3 with probability $1 - P_r$ (Fig. 2 right side).

In cellular automata sand pile model, the declination in the height of cell will make it to obtain energy as follow:

$$E'(x, y, z) = E(x, y, z) + G_A \Delta h \tag{8}$$

Where G_A is coefficient for the current force field, which can be sum of gravity, wind and buoyancy field. h is the cell's height. $E'(x, y, z)$ were cell's kinetic energy located at (x, y, z) after the dropped process, while $E(x, y, z)$ is the previous one.

Cell's state of motion located at (x, y, z) determined by its kinetic energy $E_k(x, y, z)$ and potential energy in all direction. According to the probability of static or movement, specific direction of motion is determined. When cell located at (x, y, z) moves to the neighbor cell, the conversion of kinetic energy and potential energy occurs:

$$E_p(i) = G_A[h(x, y, z) - h_{von_i}] \tag{9}$$

Here, $E_p(i)$ is the change potential energy after cell moves to the i − th von Neumann neighbor von_i, and h_{von_i} is height of von Neumann neighbor.

The cell's probabilities of staying P_s or moving in certain direction P_d are as follow [17]:

$$P_s = \sqrt{\frac{C}{E_p(i) + \varepsilon} \frac{10}{10 + E_k}} \tag{10}$$

$$P_d = (1 - P_s) \frac{n_i E_k + E_p(von_i)}{E_k + \sum E_p von} \tag{11}$$

Where E_k is the kinetic energy of cell, ε is momentum compensate factors, n_i is the weighting factor for the direction of motion, C is the static coefficient, and $\sum E_p von$ is the total change potential energy of von Neumann neighbor. Equation (10) is deduced from the sand pile model experiment. It is an experiential equation which depends on the momentum compensates factors ε and the static coefficient C. The probability of CA (Cellular Automata) moving to its neighbor determined by ratio between E_k, the changing potential energy after cell move to the i − th von Neumann neighbor and E_k, the total changing potential energy of von Neumann neighbor. In the movement process of cell at (x, y, z) moved to neighbor position (x', y', z'), the speed, location and height attribute of cell are updated, when the probability of the kinetic and potential energy conversion according to Eqs. (10) and (11).

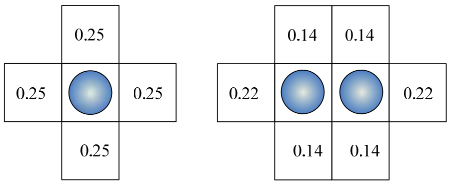

Fig. 3 Cellular automata diffusion and aggregation model.

2.3 Cellular Automata Diffusion and Aggregation Rule

Cellular automata diffuse and aggregation rule produced more twigs from its shielding effect, which is a nonlinear effect. Growth ratio at random arising tip in the growing process is faster, which makes cell in front of the twig move faster, and shields cell from proliferation on the road into the twig:

Based on the experiment of a cellular DLA (Diffusion Limited Aggregation) model, in the case of a single cell, the four directions around the growth probability are all 0.25. In the case of cellular clusters, the growth probability of Cellular clusters in two state-of-the-art location is 0.22, but other positions can only be 2/3 of the previous one, as shown in Fig. 3.

From view of the process of cell diffusion and aggregation, the growth ratio of cell away from the center of concentration is larger, and smaller at the edge. From far away to the edge of clusters a concentration gradient is formed. For such a concentration field, we use the first Fick diffusion equation $g = -D\nabla\mu$, and the second equation $\partial\mu/\partial t = -\nabla \cdot g = D\nabla^2\mu$ to describe it. g is the diffusion field which is product of the concentration gradient diffusion field $\nabla\mu$ and the diffusion coefficient D, the second equation represents that concentration versus time equals divergence of diffusion field, where μ is statistical probability of occurring in diffusion and aggregation process.

Fick's second equation can be discretized as:

$$\frac{\partial\mu}{\partial t} = \sum \frac{g(x + i, y + j, z + k)}{N} - g(x, y, z) \tag{12}$$

Where i, j and k represent relative location of the neighbor cells at (x, y, z). The first term on the right of equation is sum of the N nearest neighbor's diffusion inflow, while the second term represents the diffusion outflow from the cell. In the steady-state conditions, the divergence is zero, that is to say, $\sum g(x+i, y+j, z+k)/N - g(x, y, z) = 0$. Then according to the second Fick equation, we can get $\nabla^2\mu = 0$, which is a Laplace equation, and it can be discretized into:

$$\begin{aligned}&[(\mu_{i+1,j,k} - \mu_{i,j,k}) - (\mu_{i,j,k} - \mu_{i-1,j,k})]\\&+[(\mu_{i,j+1,k} - \mu_{i,j,k}) - (\mu_{i,j,k} - \mu_{i,j-1,k})]\\&+[(\mu_{i,j,k+1} - \mu_{i,j,k}) - (\mu_{i,j,k} - \mu_{i,j,k-1})] = 0\end{aligned} \tag{13}$$

The boundary conditions of above equation are that $\mu = 1$ in the far distance and $\mu = 0$ in the aggregate cluster boundary, and growth rate between the boundary and far distance can be expressed as:

$$V = -n \cdot g = Dn \cdot \nabla\mu \quad (14)$$

g is the cell diffusion flow, D is the diffusion coefficient. μ is concentration gradient, n is the unit vector of the cluster boundary. According to the boundary conditions (far distance $\mu = 1$, cluster border $\mu = 0$), the value of μ between the sample point is interpolated according to the geometric relationship between the points. Thus cell generation rate V on the cluster border is calculated, and growth point in the next cycle can be calculated by V and the cell's growth probability of μ in the model solver process.

3. Algorithm Implementation and Experiments

We divided process of rendering cloud into two phases:
1) Parallel executing the evolution rules to generate and update cloud volume data
2) Parallel rendering of cloud volume data by ray-casting method

In the first phase, we map the three evolution rules into kernel function for parallel computing.

In the second phase, we adopt volume rendering method in which each tracing ray's color is computed in parallel.

3.1 Parallel Executing the Evolution Rules to Generate and Update Cloud Volume Data

As the computing scope of the cellular automaton model continues to expand, serial program on solving large-scale computation will take considerable time. In the cellular automata, the calculation of status change of each cell is independent in each iteration; the next iteration of the cell state parameters is related with current state set of cell parameters, the current space environment parameters and parameters related to cellular neighbors. The status of cellular automata change process can be considered as the calculation process of the data, and this process can be performed in parallel kernel function in the parallel computing architecture such as CUDA (Compute Unified Device Architecture).

So we map the three cellular evolution rules as fluid dynamics rule, deposition and accumulation rule, diffusion and aggregation rule to the parallel kernel function on GPU. Cellular status updates according to the previous step's state by carrying out this calculation. It requires two buffers to store the state of cellular automata, using a mutual-update way to conduct the status update. In one simulation loop, kernel function parallel generating and updating cloud volume data for rendering.

3.2 Parallel Rendering of Cloud Volume Data by Ray-casting Method

The output of cellular automata model is the space volume data. We get the volume data for the cloud volume rendering, then we use the parallel ray-casting method to implement high realistic rendering. Each grid point holds the status information of the cell. Render program generates casting ray for cloud volume data, then employs direct volume rendering way based on image sequence, and calculates the final color for each casting ray.

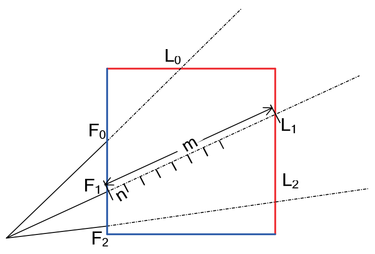

Fig. 4 Parallel ray casting algorithm.

From the viewpoint of each pixel, along a fixed direction (usually the direction of viewpoint) a ray of light is launched across the entire image sequence, and the images sequence is sampled to obtain color information in the process; meanwhile, the light absorption model accumulates the color values, until the light goes across the entire image sequence, then the resulting color value is the color of the rendered image. As shown in Fig. 4, assuming that the light is projected from point F to the cube, and cast out from the L point, the distance across in the cube is m. When the light is projected onto the cube from the F point, the sampling is started when the crossing distance is $n(n < m)$, there is a formula:

$$TEX = TEX_{start} + d * delta \quad (15)$$

Where *start* denotes the texture coordinates of the body point projected in the surface of the cube; d denotes projection direction; *delta* is the sampling interval, which increases with n; *TEX* is the sampling texture coordinates. By sampling the texture coordinates, the volume data can be queried. Until $n > m$, or cumulative transparency is over 1, the sampling process of a ray will be finished.

Light passing through an object will lead to wavelength ratio changes. When through multiple objects, this change is cumulative. Therefore, the transparent object rendering, in essence, is the mixture of color of the transparent object and the colors of subsequent objects, which is called alpha blending technology. Graphics hardware has implemented alpha blending technology, the expression is:

$$C_0 = \alpha_s C_s + (1 - \alpha_s) C_d \quad (16)$$

It means that color values C_0 obtained though the objective observed from a transparent object, is mixed with the weight for transparent objects' color C_d and the original color C_s of the objective.

The weighted coefficient distributions are α_s and $(1 - \alpha_s)$, and opacity α_s is obtained from volume data calculated by the transfer function. The mapping from volume data to opacity is called classification [18].

3.3 Experiments

We design several experiments to test the proposed algo-

Table 1 Initial conditions for cloud simulations.

Type	*h*	*V*	*D*	*T*	*P*
	(m)	(N-s/m^2)		(K)	(hPa)
Cirrostratus	6500	3.329E-7	1.21	246	4.3E2
Altostratus	3000	3.582E-7	1.73	269	7.1E2
Nimbostratus	1500	3.688E-7	1.92	279	8.3E2

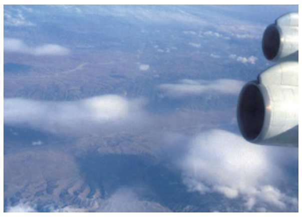

Fig. 5 Cirrostratus photo.

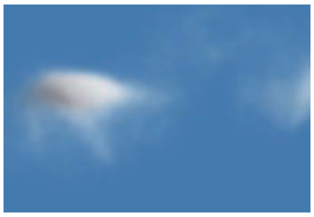

Fig. 6 One frame captured in simulation of cirrostratus.

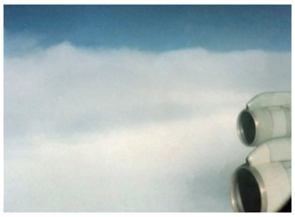

Fig. 7 Altostratus photo.

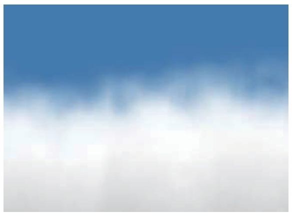

Fig. 8 One frame captured in simulation of altostratus.

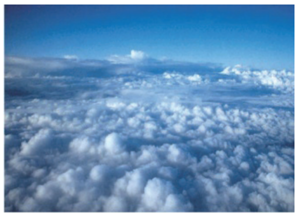

Fig. 9 Nimbostratus photo.

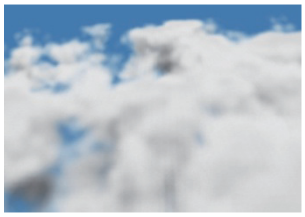

Fig. 10 One frame captured in simulation of nimbostratus.

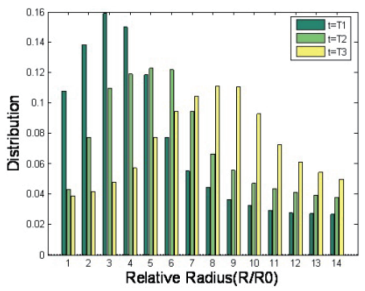

Fig. 11 Droplet distribution by size in simulation of Cirrostratus.

rithm from the performance and render effect perspective. In the model, the CPU we adopted is configured with Intel Core Q9550, with 2.83GHZ, 2GB RAM; two graphics processors capable of CUDA ability are used to test GPGPU procedures, respectively GeForce GTX560Ti and GeForce GTX480. The former one contains 384 processing cores, with a maximum of 12288 concurrent threads. The processor clock frequency is 1.66 GHz, memory clock frequency is 2004.00 MHz; the second one contains 336 processing cores and the processor clock frequency is 1350 GHz, with the memory clock frequency 1800Gbps; maximum concurrent threads is 10752.

The initial conditions for cloud simulation are listed as Table 1. In the table *h*, *v*, *D*, *T*, *P* represent height of cell space center, fluid dynamic viscosity, diffusion coefficient, environment temperature, atmospheric pressure respectively. Then cellular automata model executes the same evolution rules based on their own initial condition.

Figure 5–10 adopts the proposed render algorithm to simulate several typical clouds, and compares with actual aerial picture of typical clouds. By comparison, we can observe that our cloud rendering algorithm reflects the high cloud with transparent thin fibrous, the middle cloud ceiling with streaks and fiber strand structure, low clouds with a strip and the wave detailed characteristics.

To validate the simulation result, we fetch the experiment data generated by simulation for statistic analysis. We compared the cell generated by our algorithm with the existing different types of cloud droplet size distribution analysis report [19] according to the scale size. Here, the simulation iterations (each iteration time is 1 ms) T1 is 100, T2 is 200, T3 is 500 , shown in Fig. 11, 12, and 13, the distribution ratio in the entire space for cloud particles at different scales is given. Figures 11, 12, and 13 correspond to Figs. 6, 8, and

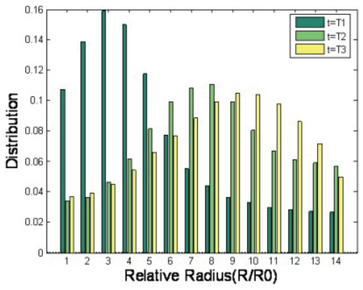

Fig. 12 Droplet distribution by size in simulation of Altostratus.

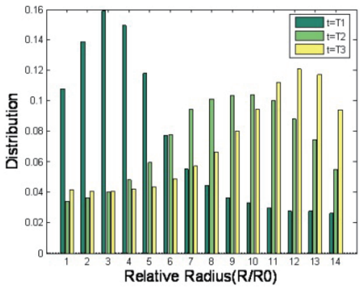

Fig. 13 Droplet distribution by size in simulation of Nimbostratus.

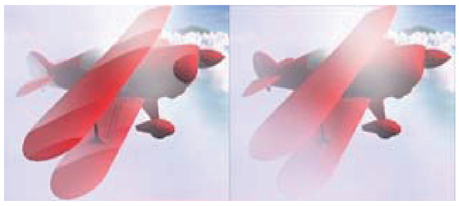

Fig. 14 Cloud rendering effect of Harris's method.

10.

The experimental results are consistent with rules presented in the report : in the process from cell proliferation to cohesion stability, cell groups have different size distribution, in high humidity and low-sky environment, it is easy to form a larger droplet size and thus the formation of higher density clouds are formed, so thick cloud block is presented; in dry atmosphere with thin air pressure, in which the cell maintains a very small droplet size instead of convergence, it forms the structure of the fibrous strand fiber shape in the rendering effect.

In the experiment, we compare the proposed method in our paper with several typical methods. Figure 14 shows the cloud rendering effect using texture-based method. Here is the cloud effect rendered by Harris method. Figure 15, 16 respectively, represent the rendering effect by Dobashi's and Eric's method which are based on Cellular automata method. Figure 17 shows the dynamic cloud picture series rendered by our method. Compared Fig. 14–16, with Fig. 17 show more realistic effect in cloud detail characteristic; besides, it also vividly presents the accumulation and deposition effect, and the details of the characteristics of the cloud edge caused by diffusion and condensation effect.

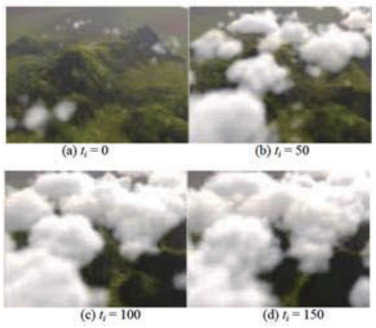

Fig. 15 Cloud rendering effect of Dobashi's method.

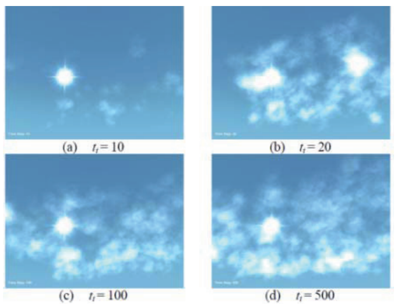

Fig. 16 Cloud rendering effect of Eric's method.

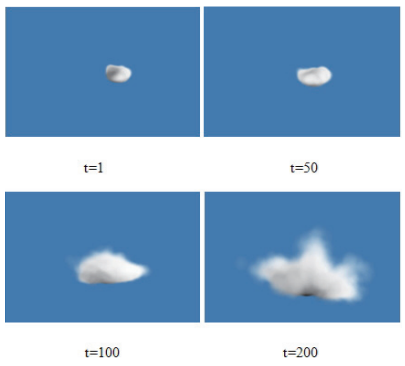

Fig. 17 Cloud rendering effect of our method.

Figure 18 shows rendering algorithms performance comparison on two different GPUs and CPU platform. In the GPU implementation, thread grid size is $256*256*256$. Though the view point is implemented as always changing dynamically, in the experiment, we put static view point out of the cloud simulation space and towards the center of it for convenience. In the simulation, when the number of cell to update is small, GPU-based rendering doesn't present its advantage in computing compared with CPU-based one, this is because the transmission overhead between host system memory and device memory has occupied a large proportion of the application time, which hide the parallel computing advantage. However, with the cell number increasing, the time required to render is growing, and the computing percent of application has accounts for most of the overall application time.

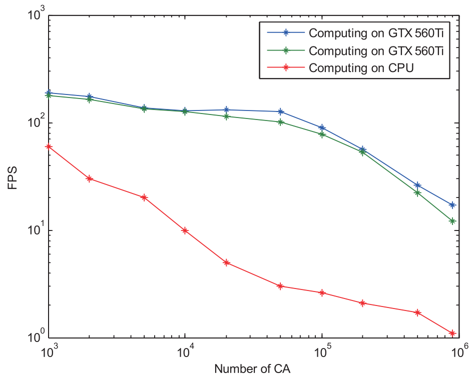

Fig. 18 The comparison of frequency per second of algorithms on GPU and CPU.

The computing advantage of computing performance on GPU compared to CPU has translated into the better overall application execution on GPU. Besides, we also found that, for the two different graphics cards GTX 480 and GTX 560TI, their rendering performance decreased exponentially after different cell number value, the former one with the cell number reaching 7E4 and the second one reaching 5E4. This is because during the model solver process, the number of cell for parallel execution has exceeded the number of concurrent threads that the card can support, as different GPU with different capacity has different thread maximum number. It is suggested that the parallel task should be scheduled according to the power of GPU.

4. Conclusion

Based on the principle of cellular automata with fluid dynamics, deposition and accumulation of cellular models and diffusion and aggregation model, we propose a CA-based parallel computational model updating, combined with the scattering model of the cloud particles. Then we present a parallel volume-rendering based algorithm for real-time cloud rendering. Experiments result show that, the physical characteristics of cloud body movements can be exhibited excellently based on cellular automata model. From the experimental performance analysis we can see that, GPU-based ray-casting algorithm combined with the image projection algorithms are able to achieve a better rendering efficiency. It can also be seen that, with increasing number of cell, performance of algorithms migrated from CPU to GPU has improved significantly.

In addition, our ray-casting algorithm is based on volume data rendering algorithm which is simple and efficient, and also easy to implement. In our future version, we will consider the multiple scattering and reflecting effect of light within the clouds, not just the case of a single scattering and transmission, the rendering effect will be more realistic.

Acknowledgements

This project was supported by a grant from the National High Technology Research and Development Program of China (863 Program) (No.2009AA01Z303).

References

[1] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita, "A simple, efficient method for realistic animation of clouds," Proc. 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00, pp.19–28, 2000.
[2] Y. Dobashi, T. Nishita, H. Yamashita, and T. Okita, "Using meatballs to modeling and animate clouds from satellite images," The Visual Computer, vol.15, no.9, pp.471–482, 1999.
[3] R. Miyazaki, S. Yoshida, Y. Dobashi, and T. Nishita, "A method for modeling clouds based on atmospheric fluid dynamics," Proc. Pacific Conference on Computer Graphics and Applications, pp.363–372, Tokyo, 2001.
[4] W. Ren, X. Liang, S. Ma, and X. Shen, "A real time simulation method for large scale 3D clouds," J. Computer-Aided Design & Computer Graphics, vol.22, no.4, pp.662–669, 2010.
[5] S. Liu, J. Chai, and Y. Wen, "A new method for fast simulation of 3D clouds," J. Computer Research and Development, vol.46, no.9, pp.1417–1423, 2009 (in Chinese).
[6] N.N. Wang, "Realistic and fast cloud rendering," J. Graphic Tools, vol.9, no.3, p.21, 2004.
[7] F. Neyret, "Qualitative simulation of convective clouds, formation and evolution," Proc. Eurographics Workshop on Animation and Simulation, Berlin, Springer, pp.113–124, 1997.
[8] M.J. Harris, W.V. Baxter, T. Scheuermann, and A. Lastra, "Simulation of cloud dynamics on graphics hardware," Proc. Graphics Hardware, pp.92–101, 2003.
[9] R. Miyazaki, Y. Dobashi, and T. Nishita, "Simulation of cumuliform clouds based on computational fluid dynamics," Proc. EURoGRAPHICS, pp.405–410, 2002.
[10] D. Overby, Z. Melek, and J. Keyser, "Interactive physically-based cloud simulation," Proc. 10th Pacific Conference on Computer Graphics and Applications, Beijing, pp.469–470, 2002.
[11] Tang Zhao and Wu Pingbo, "Real-time modeling and rendering of 3D cloud and its application in industrial simulations," J. Computer-Aided Design & Computer Graphics, vol.19, no.8, pp.1051–1055, 2007 (in Chinese).
[12] M.U. Eric and K. Sudhanshu, "Dynamic cloud simulation using cellular automata and texture splatting," Summer Simulation Multiconference, pp.270–277, 2010.
[13] F. Pighin, J.M. Cohen, and M. Shah, "Modeling and editing flows using advected radial basis functions," Proc. 2004 ACM SIGGRAPH/Euro graphics Symposium on Computer Animation, Grenoble, pp.223–232, 2004.
[14] B. Chopard and M. Droz, Cellular Automata Modelling of Physical Systems, Cambridge, Cambridge University Press, 1998.
[15] C. Ponticks and E. Hicks, "Droplet activation as related to entrainment and mixing in warm tropical maritime clouds," Atmos. Sci., vol.50, pp.1888–1896, 1993.
[16] J. Stam, Stable fluids, ACM SIGGRAPH, ACM Press, New York, 1999.
[17] C. Kim, C.F. Hans, and J.J. Henrik, "Dynamical and spatial aspects of sandpile cellular automata," J. Statistical Physics., vol.63, no.3, pp.653–684, 1991.
[18] M. Levoy, "Display of surfaces from volume data," IEEE Comput. Graph. Appl., vol.8, no.3, pp.29–37, 1988.
[19] S.K. Paul, "Cloud drop spectra at different levels and with respect to cloud thickness and rain," Atmospheric Research, vol.52, no.4, pp.303–314, 2000.

Liqiang Zhang received the M.S. degrees in Computer Science from Beijing Institute of Technology in 2005. He now studied in Institute of Software, Chinese Academy of Sciences as a Ph.D. candidate. He had participated in a project related to research the parallel algorithms and applications in field of computer graphics, virtual reality and computer simulation.

Chao Li received the B.S. degree in Computer Science from Dalian University of Technology in 2009. During 2009–2012, he stayed in Institute of Software, Chinese Academy of Sciences, study high performance parallel algorithms and applications. He now is a 3rd year master in computer science.

Haoliang Sun is a Ph.D. candidate in Institute of Software, Chinese Academy of Sciences. His research interest is the transport layer protocols in deep space communication. He had participated in a project related to research the communication mechanisms and design transport protocols in deep space networks.

Changwen Zheng is a Professor in the National Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences. He received his Ph.D. degree from Huazhong University of Science and Technology. His research interests include computer graphics, virtual reality, computer simulation and artificial intelligence.

Pin Lv is an Associate Professor in the National Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences. He received his Ph.D. degree from Chinese Academy of Sciences. His research interests include computer graphics, virtual reality, and computer simulation.