Realistic Simulation of 3D Cloud

HANG QIU[1,2,3] LEI-TING CHEN[1,3] GUO-PING QIU[2] HAO YANG[1,2,3]
[1] School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu, CHINA
[2] School of Computer Science, University of Nottingham, Nottingham, UNITED KINGDOM
[3] Provincial Key Laboratory of Digital Media, Chengdu, CHINA
qiuhang@uestc.edu.cn

Abstract: -The digital creation of cloud is important for many applications in computer graphics, including outdoor simulations and the digital rendering of atmospheric effects. Unfortunately, it is still difficult to simulate realistic cloud with interactive frame rates due to its peculiar microstructures and complex physical process of formation. Realistic simulation of cloud turns to be one of the most challenging topics in computer graphics. In this paper, we present a method for simulating 3D cloud. The Coupled Map Lattice (CML) is adopted for the modeling of cloud, and the simulation of light scattering in clouds is achieved by using a series of spherical harmonics and spherical harmonic coefficients that represent incident-light distribution. A frequency domain volume-rendering algorithm combined with spherical harmonics is applied to implement fast rendering of cloud scenes. Experiments demonstrate that our method facilitates computing efficiency, while yielding realistic visual quality.

Key-Words: - Natural Scene, Coupled Map Lattice (CML), Spherical Harmonics, Frequency Domain, Cloud Rendering, Volume Rendering

1 Introduction

Simulation of natural phenomena has become an important research field and one of the most difficult tasks in computer graphics. Cloud is an essential component of natural scenes, which plays an important role in various fields of applications, such as computer games, military training simulations, flight simulations, virtual reality and special effects of movies. Unfortunately, like other amorphous phenomena, clouds elude traditional modeling techniques with their peculiar pattern of intricate, ever-changing volume-filling microstructures. In addition, the visual aspect of a cloud depends on complex light interactions. Simulating the effect of light propagation through clouds is a time consuming and computational intensive process. Moreover, the motion of cloud is extremely complex, which covers atmospheric flows, evaporation of water, water condensation and heat exchange. All aspects above-mentioned make it difficult to generate visually convincing clouds scene efficiently. Limited computational power and the extreme complexity of physical process have called for tradeoffs between efficiency and realism.

In this paper, a method for simulating realistic 3D cloud is proposed. Our goal is to look for the right compromise between realism and efficiency. The major contributions of our work can be listed as follows.

(1) To simulate the complex light scattering in clouds, spherical harmonics-based approach is presented. A series of spherical harmonics and spherical harmonic coefficients are used to represent incident light distribution. The values of spherical harmonic coefficients are obtained by off-line sampling. This method can not only control computation cost, but also meet the demands of reality.

(2) Considering the existence of large-scale data, massive integrals and ever-changing viewpoint in the scene, a frequency domain volume rendering algorithm combined with spherical harmonics is implemented, which improves the rendering efficiency effectively.

The rest of the paper is organized as follows. Section 2 briefly overview related work in simulation of clouds. Section 3 gives the overview of our method. Section 4 depicts the modeling of cloud based on CML, while Section 5 proposes an approach for the lighting simulation based on spherical harmonics. Section 6 discusses cloud rendering based on frequency- domain volume rendering algorithm. Section 7 presents experiments to demonstrate the efficiency of our method. The conclusion of this paper is given in Section 8.

2 Related Work

Extensive works made efforts to simulate realistic clouds. The related technologies can roughly be classified into two categories: modeling and rendering. In this section, we give a briefly review of related work. The more specific survey can be found in [1].

2.1 Modeling of Cloud

Cloud modeling deals with the data used to represent clouds in the computer, and how the data are generated and organized. The methods for the modeling of cloud can be classified into two main categories. One is procedural techniques and the other is based on physical simulation.

Schopik et al. [2] used volumetric implicit to model clouds, which has the beneficial attributes of smooth blending, simple computation. Nishita et al. [3] adopted fractals modeling method to represent cloud. Texture sprite modeling approach is developed in [4,5]. Above mentioned procedural techniques are computationally inexpensive, and the cloud shapes can be controlled by various parameters. However, the adjustment of parameters is often conducted by trial and error, which has limited extension.

The physically-based simulation methods are more accurate when describing the shape of cloud and other attributes of cloud. Dobashi et al. [6] developed an efficient method for cloud simulation using cellular automaton. Nevertheless, it is hard to simulate complicated physics process. Miyazaki et al. [7] used Coupled Map Lattice, an extension of cellular automaton, to simulate various types of clouds. Harris [8,9] took the phase transition into account when using Stam's stable fluid solver [10] for fluid dynamics. Unfortunately, due to its high computational demands, physics-based simulation is not appropriate to real-time rendering.

2.2 Rendering of Cloud

Realistic rendering of clouds involves solving the complex interaction of light within the cloud and with its environment. Interactive methods achieve efficient cloud rendering by ignoring most light interaction modes. As early as 1985, Gardner [11] has developed mapping technique to display clouds. Harris et al. [8,12] developed a fast rendering method for dynamic clouds. Both the motion and the images of clouds are computed on the GPU using 3D textures.

To improve realism, scattering of light should be taken into account. However, simulating the multiple scattering of light in a cloud would require an unacceptable time to converge. Various simplifications have been proposed. Miyazaki et al. [13] considered single scattering of light and shadow-view slices-based rendering algorithm. Bouthors et al. [14, 15] analyzed light behaviour in plane-parallel homogeneous slabs and based on this analysis design a series of ad-hoc functions to obtain illumination in the rendered cloud. Most recently, Elek et al. [16] proposed an interactive algorithm. It utilizes a temporally coherent illumination caching process to amortize the simulation costs across multiple frames.

3 Overview of Our Method

In this section, we give an overview of our method. As shown in Fig.1.

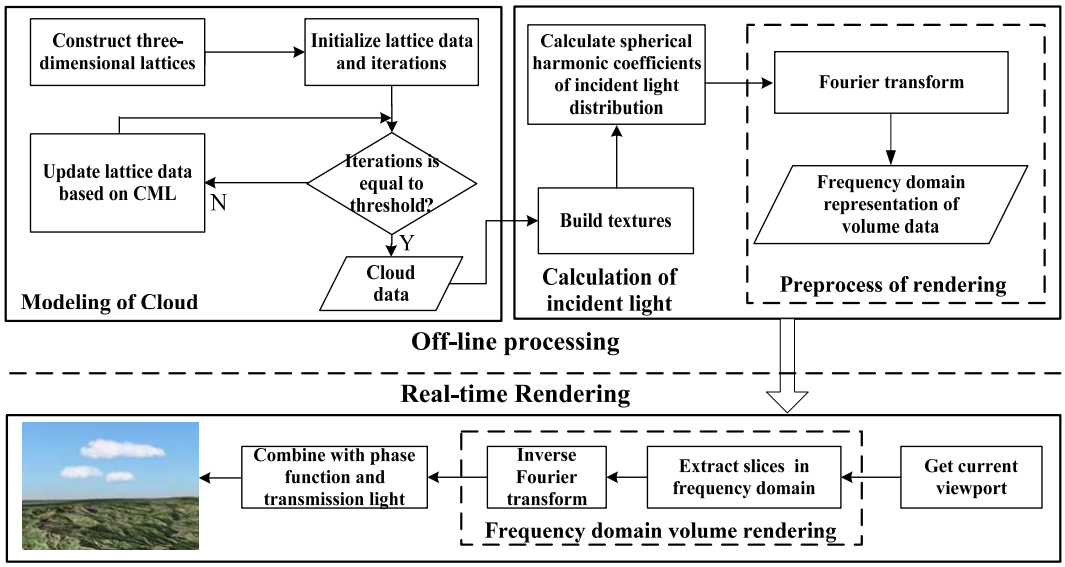


Fig. 1 Simulation framework

In our method, the simulation process is divided into two phases. The main work of off-line processing is to generate cloud data, pre-compute incident light and transform volume data to the representation of frequency domain. The work of real-time rendering processing is to carry out frequency domain volume rendering. The details of aforementioned two phases are amplified as follows.

(1) Off-line processing: firstly, the simulation space is subdivided into lattices. Lattice data and iterations are initialized. Using Coupled Map Lattice (CML), the physical process of phase transition is simulated and cloud data can be generated. To simulate the interaction between light and cloud, we use a series of spherical harmonics and spherical harmonic coefficients to represent incident light distribution. At the same time, the cloud data are rendered to textures and the texture slices are constructed. The volume data, which include spherical harmonic coefficients and density, can be obtained by iterative computation. At last, using Fourier transform to achieve the frequency domain representation of volume data.

(2) Real-time rendering: according to the current viewpoint, texture slices are extracted in frequency domain and transformed to spatial domain data. Combining with phase function and transmission light, the final image in the direction of sight-line can be achieved.

4 The Modeling of Cloud based on CML

In nature, clouds are formed as a bubble of air is heated by the underlying terrain, causing the bubble to rise into regions of low temperature. Then phase transition occurs, that is, water vapor in the bubble becomes water droplets (clouds). Obviously, atmospheric fluid dynamics is the most direct approach to model clouds. However, it requires a great deal of computational cost.

Coupled Map Lattice (CML) is a dynamical system that models the behaviour of non-linear systems. It is easy to implement and computational cost is small. Therefore, we applied CML to implement the modeling of cloud.

As shown in Fig. 2. The simulation space is subdivided into lattices. The number of lattices is $N_x \times N_y \times N_z$.

Each lattice point (x, y, z) contains a set of properties for simulating the physical state of cloud. The set of properties can be expressed as:

$$P(x, y, z) = \{v, w_v, w_l, E\} \quad (1)$$

where $v(x, y, z)$ is the velocity vector. $w_v(x, y, z)$ is the water vapour. $w_l(x, y, z)$ depicts the water droplets, while $E(x, y, z)$ is the internal energy.

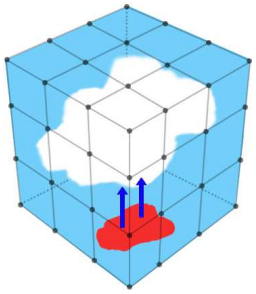


Fig. 2 The generation of cloud data

We adopt CML to simulate the formation of cloud. During the simulation, the distribution of the state values in each lattice are calculated and updated at every time step. Let P be the state value for current time step. The state value P^* in the next time step can be calculated by P.

Similar as [7], we considered the factors including viscosity and pressure effects, advection, diffusion of water vapour, thermal diffusion, thermal buoyancy and phase transition.

Velocity field is affected by viscosity and pressure. To avoid complex computation and simulate the influence of viscosity and pressure on velocity field, the z component of the velocity v for the next time step (v_z^*) is calculated as:

$$\begin{aligned} v_z^*(x, y, z) = v_z(x, y, z) + k_v \Delta v_x(x, y, z) \\ + k_p grad(divV)_x \end{aligned} \quad (2)$$

where v_z is the z component of velocity v for the current time step. k_v is the viscosity ratio and k_p is the coefficient of pressure effect. The computations of x and y components are in the similar way.

The buoyancy effect is simulated by the following equation:

$$\begin{aligned} v_z^*(x, y, z) = v_z(x, y, z) + \frac{k_b}{4}[4E(x, y, z) \\ - E(x+1, y, z) - E(x-1, y, z) \\ - E(x, y+1, z) - E(x, y-1, z)] \end{aligned} \quad (3)$$

where k_b is the coefficient affecting the strength of the buoyancy force.

Diffusion of water vapour and thermal diffusion are simulated as follows:

$$w_v^*(x,y,z) = w_v(x,y,z) + k_{w_v}\Delta w_v(x,y,z) \quad (4)$$

$$E^*(x,y,z) = E(x,y,z) + k_E\Delta E(x,y,z) \quad (5)$$

k_{w_v} is the diffusion coefficient for water vapour. $k_{w_v}\Delta w_v(x,y,z)$ depicts the diffusion effects. k_E is the coefficient of thermal diffusion. Δw_v and ΔE represent the difference of water vapour and temperature per time.

Affected by viscosity field, advection, diffusion and buoyancy, water vapour value in each lattice is transported to the adjacent lattices. When the amount of water vapour reaches to the maximum amount of water vapour, phase transition that generates water droplets (clouds) happens.

5 Lighting Simulation based on Spherical Harmonics

To achieve realistic performance, interactions between light and cloud must be considered. In nature, the colour of a cloud depends on the light that cloud received and the position of the viewpoint.

The incident light intensity of a cloud is composed of sunlight, sky light and reflected light from earth [3]. As shown in Fig. 3, when incident light spreads in cloud, it will be affected by transmission and multiple scattering. Transmission is only related to the attenuation of light intensity. However, multiple scattering affects both the intensity and propagation direction of light. The intensity of emergent light is formed by transmission light and scattering light.

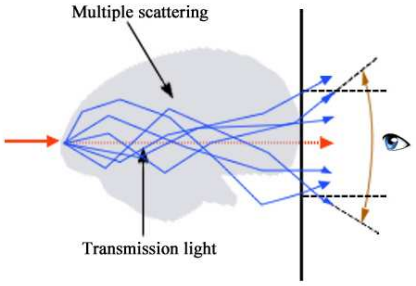


Fig. 3 Light spreading inside cloud

Riley et al. [17] used angular distribution, which represented by convolution, to simulate multiple scattering. This method considered only the forward portion of multiple scattering, and it is still semi-empirical. Some other researchers, represented by Dobashi [6, 19] and Miyazaki [18], simplified the complicated integral calculation to polynomial. Although these approaches can achieve real-time performance, the display results are not satisfied.

Compared with the above-mentioned methods, in this section, we present a trade-off approach.

5.1 Basic Principle

To simulate the multiple scattering of light in cloud, the incident light distribution, which includes transmission light and scattering light, is calculated. To improve clarity in the following sections, we define some symbols. As shown in Table 1.

Table 1. Definition of Symbol

Symbol	Definition
$\vec{\omega}$	Direction
$\vec{x}_n$	Position of cloud particle
$L(\vec{x}_n,\vec{\omega})$	The intensity reflected from position $\vec{x}_n$ in direction $\vec{\omega}$
L_{in} / $\tilde{L}_{in}$	Incident light intensity/sample of incident light intensity
L_{out}	Emergent light intensity
L_{back}	Background transmission light intensity
$c^m_{l(\delta)}$	Spherical harmonic coefficients
$P(\vec{\omega},\vec{\omega}')$	Phase function
K_s	Dissipation coefficient
C	Polynomial
ξ	Approximate value of backward scattering light at position $\vec{x}_n$
$P(\vec{\omega}_L,\vec{\omega}')$	Phase function. Its reference point is light source

5.1.1 Representation of Incident Light Distribution

Incident light distribution depicts the incident light intensity of cloud particles under the influence of transmission and scattering.

We use spherical harmonics [20] to represent incident light distribution, which can be expressed as follows:

$$L_{in}(\vec{\omega}) = \sum_{l,m} c^m_{l(L_{in})} Y_l^m(\vec{\omega}) \quad (6)$$

$L_{in}(\vec{\omega})$ is the incident light intensity in direction $\vec{\omega}$. $Y_{l,m}$ is spherical harmonics that can be traditionally represented as:

$$Y_l^m(\vec{\omega}) = Y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}K_l^m \cos(m\varphi)P_l^m(\cos\theta), & m>0 \\ \sqrt{2}K_l^m \sin(-m\varphi)P_l^{-m}(\cos\theta), & m<0 \\ K_l^0 P_l^0(\cos\theta), & m=0 \end{cases} \quad (7)$$

where $l \in R^+$, $m \in [-l,l]$. $P_l^m(\cos\theta)$ is associated Legendre polynomials. K_l^m is a polynomials that related to l and m . $Y_l^m(\vec{\omega})$ can be calculated by transforming it to spherical coordinates. That is to

say we can use equation (6) to calculate the incident light intensity in any directions when the value of spherical harmonic coefficient $c^m_{l(L_{in})}$ could be obtained. Therefore, we store a set of spherical harmonic coefficients for each cloud particle to represent the incident light distribution of each cloud particle. The spherical harmonic coefficient can be calculated by using a series of sampling value $\tilde{L}_{in}(\vec{\omega})$:

$$c^m_{l(L_{in})} = \int_{\Omega} \tilde{L}_{in}(\vec{\omega}) Y_l^m(\vec{\omega}) d\vec{\omega} \quad (8)$$

5.1.2 Choice of Sampling Equation

In our approach, the sampling object is the incident light of current cloud particle in various directions. We divide the incident light into three different types: direct sunlight, forward incident light and backward incident light. As shown in Fig. 4.

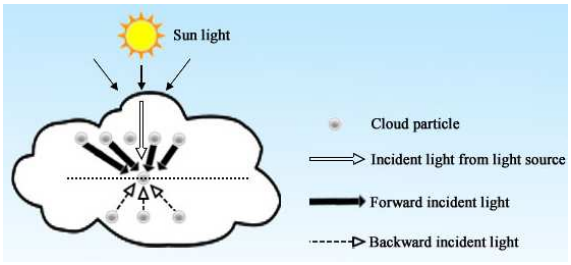


Fig. 4 Composition of incident light

Sampling is time consuming. To reduce the computation cost, we adopt different sampling approaches according to different types of incident light.

(1) Direct sunlight. It provides the most incident energy for cloud particle by transmission and scattering. We use a revised light transport model for sampling, which can not only guarantee calculation precision, but also reduce the calculation time.

(2) Forward incident light. It affects the intensity of incident light by scattering. We ignore the influence of transmission, and adopt the simplified light transport model to calculate.

(3) Backward incident light. Compared with the two types of incident lights above mentioned, its intensity is weaker. We put forward an approximate simulation by adopting a constant term.

Through sampling calculation of three types of incident light, the sampling data of cloud particle in various directions can be achieved, thus $c^m_{l(L_{in})}$ can be obtained.

5.1.3 Calculation of Lighting based on GPU

To benefit from GPU processing, we adopt render to texture (RTT) technique. The density and spherical harmonic coefficients are organized to textures. A series of spherical harmonic coefficients of each cloud particle are achieved by iterative calculation. Thus, the incident light intensity of each cloud particle can be obtained.

5.2 Revised Light Transport Model based on Spherical Harmonics

To simulate the complicated scattering, the light transport model is usually applied:

$$L(\vec{x}_n, \vec{\omega}) = L(0, \vec{\omega}) T(0, D) + \int_0^D g(s) T(s, D) ds \quad (8)$$

$L(0, \vec{\omega})$ is the incident radiance. $L(0, \vec{\omega}) T(0, D)$ indicates incident light that reaches current particle by transmission in direction $\vec{\omega}$. $T(s, D)$ is the transmittance.

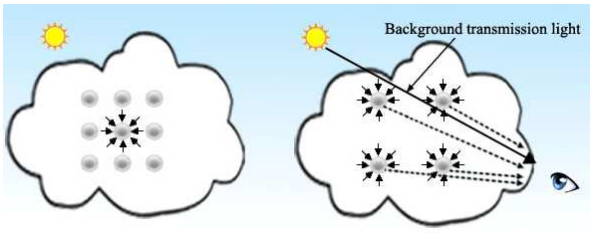
$$g(s) = K_s(\vec{x}(s)) \int_{4\pi} P(\vec{x}, \vec{\omega}, \vec{\omega}') L(\vec{x}(s), \vec{\omega}') d\vec{\omega}' \quad (9)$$

$g(s)$ represents the attenuation of intensity $L(\vec{x}_n, \vec{\omega})$ at $\vec{x}_n$ in direction $\vec{\omega}$ due to the scattering of current particle. At the same time, the scattered light from other particles enhanced its intensity.

In order to use the light transport model above-mentioned to implement sampling calculation in various directions, we make light transport model equivalent to view-independent incident light model and view-dependent emergent light model.

5.2.1 Equivalent Representation of Light Transport Model

When parallel light reaches cloud, cloud particles receive different intensity of incident light in various directions. Therefore, view-dependent incident light distribution is formed. As shown in Fig. 5 (a), even if the viewpoint changes, the incident light distribution of particle will not be changed. For the emergent light, its intensity includes background transmission light and scattered light. The intensity of light that received by viewpoint will be changed according to the position of viewpoint. As shown in Fig. 5 (b).



(a) View-independent (b) View-dependent

Fig. 5 View-independent incident light distribution and view-dependent emergent light

The equation (8) is made equivalent to two parts: calculation of incident light and calculation of emergent light .As shown in equation (10) (11):

$$L_{in}(\vec{x}_n,\vec{\omega}) = L_{in}(0,\vec{\omega})T(0,D_n(\vec{\omega})) + \int_D T(s,D_n)K_s(\vec{x}(s))\int_{4\pi} P(\vec{x}(s),\vec{\omega}',\vec{\omega})L_{in}(\vec{x}(s),\vec{\omega}')d\vec{\omega}'ds \quad (10)$$

$$L_{out}(u,v,\vec{\omega}_V) = L_{back}(0,\vec{\omega}_V)T(\vec{x}_{u,v},D) + \int_s T(s,D_n)K_s(\vec{x}(s))\int_{\Omega} L_{in}(\vec{x}(s),\vec{\omega}')P(\vec{\omega},\vec{\omega}_V)d\vec{\omega}'ds \quad (11)$$

Equation (10) obtains intensity of incident light in various directions by background transmission light and scattered light of surrounding particles. The core of equation (10) is to calculate multiple scattering model, which is implemented in off-line processing by using spherical harmonics.

Equation (11) uses incident light distribution, which obtained by equation (10), to calculate intensity of emergent light in direction of sight line. According to the relationship between emergent light and the position of viewpoint, we use equation (11) in real-time processing.

5.2.2 Spherical Harmonic-based Sampling

(1) Direct sunlight

To achieve sampling value of incident light $\tilde{L}_{in}(\vec{x}_n,\vec{\omega}_L)$ in light source direction, we construct the recursive form of equation (10):

$$L_{in}(\vec{x}_n,\vec{\omega}) = C(\vec{x}_{n-1}(\vec{\omega}_L)) + K_s\int_{4\pi} P(\vec{\omega}_L,\vec{\omega}')L_{in}(\vec{x}_{n-1}(\vec{\omega}_L),\vec{\omega}')d\omega' \quad (12)$$

$$C(\vec{x}_{n-1}(\vec{\omega}_L)) = L_{in}(\vec{x}_{n-1},\vec{\omega}_L)T(\vec{x}_{n-1},\vec{x}_n) \quad (13)$$

$C(\vec{x}_{n-1}(\vec{\omega}_L))$ can be calculated according to the result of the previous iteration. $P(\vec{\omega}_L,\vec{\omega}')$ is phase function. Its reference direction is equivalent to the direction of light source. We use the orthogonality of spherical harmonics to achieve the final sampling equation as follows:

$$L_{in}(\vec{x}_n,\vec{\omega}) = C(\vec{x}_{n-1}(\vec{\omega}_L)) + K_s\sum_{l,m} c^m_{l(P)}c^m_{l(L_{IN})}(\vec{x}_{n-1}) \quad (14)$$

$$c^m_{l(P)} = \int_{\Omega} P(\vec{x}_n,\vec{\omega})Y^m_l(\vec{\omega})d\vec{\omega} \quad (15)$$

(2) Forward incident light

For the forward incident light, only the scattering lights of the adjacent particles are considered. Therefore, the computational model can be represented as:

$$\tilde{L}_{in}(\vec{x}_n,\vec{\omega}) = \int_D T(s,D_n)K(\vec{x}(s))\int_{4\pi} P(\vec{x}(s),\vec{\omega}',\vec{\omega})L_{in}(\vec{x}(s),\vec{\omega}')d\vec{\omega}'ds \quad (16)$$

The recursive form of equation (16) can be expressed as:

$$\tilde{L}_{in}(\vec{x}_n,\vec{\omega}) = T(\vec{x}_{n-1},\vec{x}_n)[L_{in}(\vec{x}_{n-1},\vec{\omega}) + K_s(\vec{x}_{n-1})\int_{4\pi} P(\vec{x}_{n-1}(\vec{\omega}),\vec{\omega}',\vec{\omega})L_{in}(\vec{x}_{n-1}(\vec{\omega}),\vec{\omega}')d\vec{\omega}'] \quad (17)$$

According to the orthogonality and rotation invariance of spherical harmonics, equation (17) can be rewritten as:

$$\tilde{L}_{in}(\vec{x}_n,\vec{\omega}) = T(\vec{x}_{n-1},\vec{x}_n)L_{in}(\vec{x}_{n-1},\vec{\omega}) + T(\vec{x}_{n-1},\vec{x}_n)K_s(\vec{x}_{n-1})\sum_{l,m} R_{\vec{\omega}_L,\vec{\omega}}(c^m_{l(P)})c^m_{l(L_{in})}(\vec{x}_{n-1}) \quad (18)$$

$R_{\vec{\omega}_L,\vec{\omega}}$ represents the new coefficient after rotation.

(3) Backward incident light

Backward incident light has little contribution to the intensity. We adopt a constant value ξ to simulate it.

In conclusion, the final equation that depicts incident light distribution can be expressed as follows:

$$c^m_{l(L_{in})}(\vec{x}_n) = 4\pi/N \times [\tilde{L}_{in}(\vec{x}_n,\vec{\omega}_L)Y^m_l(\vec{\omega}_L) + \sum_{\omega\in forward}\tilde{L}_{in}(\vec{x}_n,\vec{\omega})Y^m_l(\vec{\omega}) + \sum_{\omega\in backward}\xi\ Y^m_l(\vec{\omega}')] \quad (19)$$

N is the total sample number. Equation (19) can be used to calculate view-independent illumination model.

5.3 Implementation

We adopt RTT technique to reiterate the SH coefficients for each cloud particle. The calculation process is divided into two phases: CPU processing and GPU processing. As shown in Fig. 6.

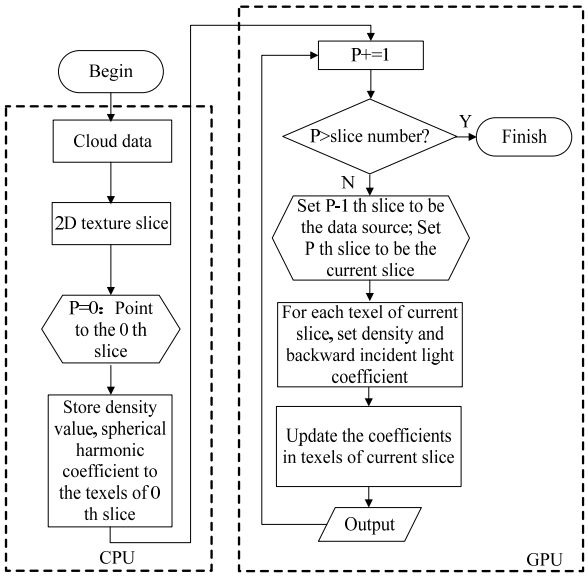


Fig. 6 Calculation flow

(1) CPU processing

Firstly, based on RTT technique, the lattice data are translated into 2D texture slices. The first texture is set to be the data source of the first iterative calculation.

Each texel of texture corresponds to a cloud particle. The spherical harmonic coefficients and density are stored in RGBA channels. As shown in Fig. 7.

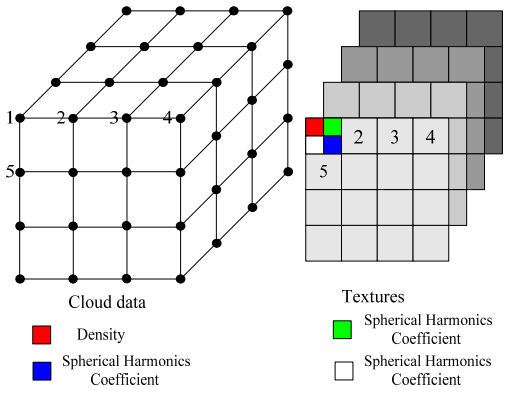


Fig. 7 Cloud data storage structure

In the following calculation process, the spherical harmonic coefficients are updated according to the iterative calculation results.

(2) GPU processing

Firstly, using equation (19), spherical harmonic coefficients of current texture are calculated according to the information of the first texture. The results of the first iteration are set to be the input data for the following iterative calculation. This procedure can be repeated, until all the textures are processed.

6 Cloud Rendering based on Frequency Domain Volume Rendering

The most popular rendering techniques for 3D cloud simulation include splatting [6, 7] and slice-based volume rendering [8, 9, 17]. These methods take full advantage of parallel capability of GPU and achieve high rendering speed. However, when viewpoint changes, the volume clouds need to be reconstructed, the computation cost can not be ignored.

Considering the ever-changing viewpoint, integral calculation and massive cloud data in cloud scene, frequency domain volume rendering [21, 22] is a feasible approach. Nevertheless, traditional frequency domain volume rendering algorithm lacks of depth information. Therefore, we present a frequency domain volume rendering method combined with spherical harmonics to look for the right compromise between realism and efficiency.

6.1 Frequency Domain Volume Rendering Algorithm

The traditional frequency domain volume rendering algorithm is based on Fourier projection slice theorem. Once a volume data is Fourier transformed, an image for any viewing direction can be obtained by extracting a 2D slice of the 3D spectrum at the appropriate orientation and then inverse Fourier transforming it [21, 22]. As shown in Fig. 8.

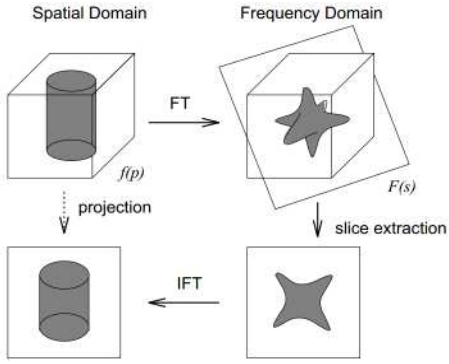


Fig. 8 Frequency-domain volume rendering

The implementation of frequency domain volume rendering includes two steps:

(1) Pre-processing step. 3D spatial data are transformed into the frequency domain, which can be expressed as:

$$F(u,v,w)=\iiint f(x,y,z)e^{-2\pi i(xu+yv+zw)}dxdydz \quad (20)$$

(2) Real-time rendering step. A parallel projection of $f(x,y,z)$ can be computed by evaluating $F(u,v,w)$ along a plane defined by the arbitrary orthonormal vectors. Each point of the plane corresponds to a function value:

$$P_\theta(\omega_u,\omega_v)=F(\omega_u\vec{u}+\omega_v\vec{v}) \quad (21)$$

At last, taking its inverse 2D Fourier transform yields the projection.

The primary motivation of frequency domain volume rendering is that, once the forward 3D transform is computed as a pre-processing operation, we can compute projections at arbitrary angles quickly by working with 2D manifolds of the data in frequency domain. Its essence is linear integral along some directions.

6.2 Frequency Domain Volume Rendering Algorithm Combined with Spherical Harmonics

In real-time rendering phase, equation (11) is used to obtain the light intensity in direction of sight line. As shown in equation (11).

Set $A = T(s, D_n)K(\vec{x}(s))\int_{\Omega} L_{in}(\vec{x}(s), \vec{\omega})P(\vec{\omega}, \vec{\omega}_V)d\vec{\omega}$

Equation (11) can be expressed as:

$$L_{out}(u, v, \vec{\omega}_V) = L_{back}(0, \vec{\omega}_V)T(\vec{x}_{u,v}, D) + \int_s Ads \quad (22)$$

Obviously, $L_{in}(\vec{x}(s), \vec{\omega})$ is represented by a series of spherical harmonic coefficients $c^m_{l(L_{in})}$. $T(\vec{x}_{u,v}, D)$ is optical depth, which can be calcuated as follows:

$$T(\vec{x}, D(\vec{\omega})) = e^{-\int_s^{s'} \sigma\eta(x+t\vec{\omega})dt} \quad (23)$$

η is density. σ depicts albedo. All of them are set to be constant values. Thus, we only need to calculate:

$$F(u, v, w) = \iiint \sigma\eta e^{-2\pi i(xu+yv+zw)}dxdydz \quad (24)$$

(1) Depth information support

Traditional frequency domain volume rendering lacks depth information. Totsuka et al. [22] provided an approach to overcome this drawback. However, it is time consuming. In our method, we adopt a function $\psi(\eta(s))$ related to density to approximately substitute $T(s, D)$. Thus, in equation (22), $\int_s Ads$ can be depicted as:

$$\int_s \psi(\eta(s))K(\vec{x}(s))\int_{\Omega} L_{in}(\vec{x}(s), \vec{\omega})P(\vec{\omega}, \vec{\omega}_V)d\vec{\omega}ds \quad (25)$$

(2) Phase function support

Firstly, we transform equation (25) into the following form:

$$\int_s \psi(\eta(s))K(\vec{x}(s))[\vec{V}(c^m_{l(P)}) \cdot \vec{V}(c^m_{l(L_{in})}(\vec{x}(s)))]ds \quad (26)$$

where $\vec{V}(c^m_{l(L_{in})}(\vec{x}(s)))$ is the spherical harmonic coefficient vector corresponding to incident light distribution. $\vec{V}(c^m_{l(P)})$ is the spherical harmonic coefficient vector corresponding to phase function. Equation (26) can be formed as:

$$\sum_s [\psi(\eta(s))K(\vec{x}(s)) \times \vec{V}(c^m_{l(L_{in})}(\vec{x}(s)))] \cdot \vec{V}(c^m_{l(P)}) \quad (27)$$

The essential of equation (27) is to calculate emergent light intensity accumulation of all the sampling points in direction of sight line. Thus, the phase function of each sampling point is the same. Each sampling point in the integral path has the same $\vec{V}(c^m_{l(P)})$. Therefore,

$$\vec{V}(c^m_{l(P)}(s_1)) = \vec{V}(c^m_{l(P)}(s_2)) = ... = \vec{V}(c^m_{l(P)}(s_n)) \quad (28)$$

Equation (27) can be depicted as:

$$\begin{aligned} &\sum_l \sum_m c^m_{l(P)} \sum_s c^m_{l(L_{in})}(\vec{x}(s))\alpha(\vec{x}(s)) \\ &= V(c^m_{l(P)}) \cdot \left[V(\sum_s c^m_{l(L_{in})}(\vec{x}(s)) \times \alpha(\vec{x}(s))) \right] \\ &= V(c^m_{l(P)}) \cdot \left[V(\int_s c^m_{l(L_{in})}(\vec{x}(s))\alpha(\vec{x}(s))ds) \right] \end{aligned} \quad (29)$$

where $\alpha(\vec{x}(s)) = \psi(\eta(s))K(\vec{x}(s))$.

From a physical standpoint, the essential of our method is to make the incident light distribution of each particle in the integral path equivalent to the total incident light distribution of a single particle. As shown in Fig. 9.

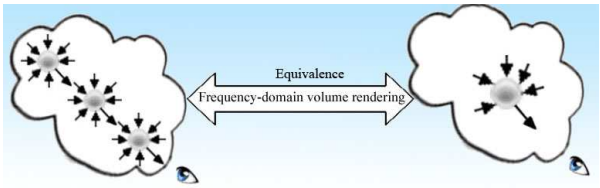
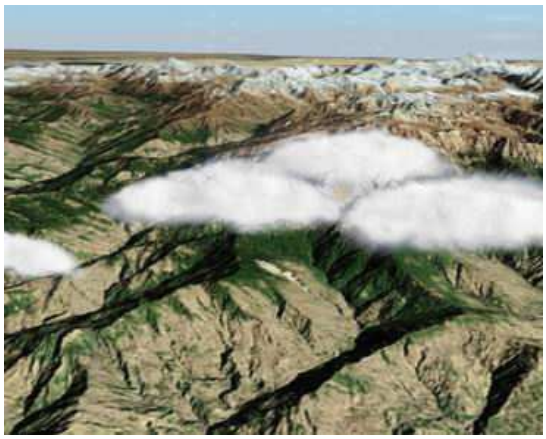


Fig. 9 Physical meaning of phase function-supported frequency domain volume rendering

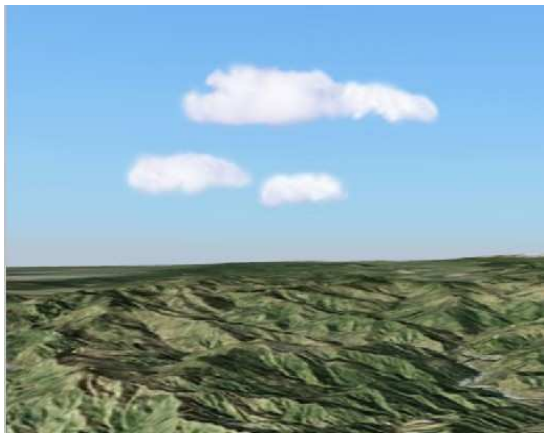
7 Experiments and Discussions

We use OpenGL on C++ platform to implement the method presented in this paper, and carry out related experiments on PC. The computer configuration used for experiments is Windows XP operating system, Intel Pentium 2.6GHz CPU, 2G memory, and NIVDIA 9800GT graphics card.

The performances of our rendering method are shown in Fig.10, which have little perceivable loss in image quality.



(a)

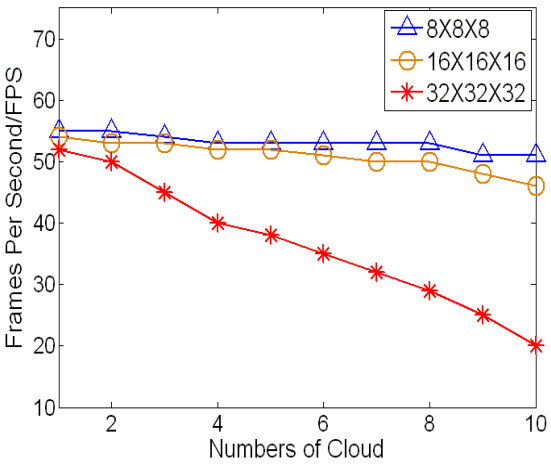


(b)

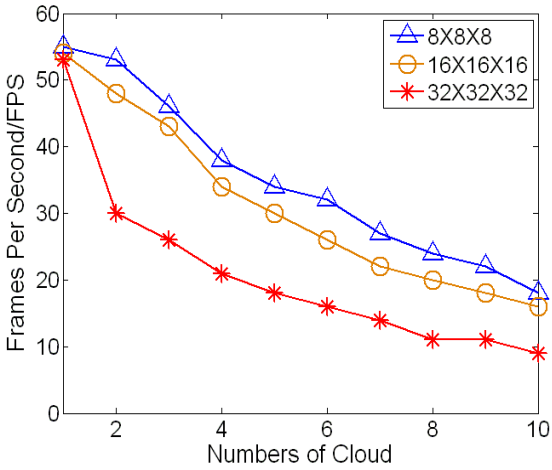
Fig. 10 The snapshots of cloud simulation

To testify the influences of lattice number and interpolation algorithm on the rendering efficiency, we carried out a series of pressure tests. Firstly, we constructed three scenes. The lattice number in above-mentioned three scenes are set to $8\times8\times8$, $16\times16\times16$ and $32\times32\times32$ respectively. Based on tri-linear interpolation and high order interpolation, we analyze rendering efficiency. The fly-through path is predefined. We sampled 2000 frames in each scene respectively, and achieved the average frame rate. Fig.11 shows the result.

For tri-linear interpolation, when the number of lattice is $8\times8\times8$ and $16\times16\times16$, with the increase of cloud number, the rendering efficiency decreased not evidently. When the cloud number is 10, the rendering speed is 50fps and 45fps, respectively. With the increase of lattice point, the rendering efficiency decreased. Benefit from frequency domain volume rendering, the computational cost can be controlled effectively. When the number of lattice is $32\times32\times32$ and the number of rendering cloud is 10, the rendering speed is still 20fps.



(a) Rendering efficiency with tri-linear interpolation



(b) Rendering efficiency with high order interpolation

Fig. 11 Rendering efficiency with different cloud resolution and interpolation algorithm

For high order interpolation, with the increase of lattice point, the rendering efficiency decreased quickly. When the number of lattice is $32\times32\times32$ and the number of rendering cloud is 4, the rendering speed is still 20fps.

8 Conclusion

Realistic simulation of cloud is a challenging problem in the field of natural phenomena simulation with wide applications. In this paper, we have proposed an effective method for simulating 3D cloud. The interactions between light and cloud are considered, and achieved by using a series of spherical harmonics and spherical harmonic coefficients to represent incident-light distribution. To improve rendering speed, a frequency domain volume-rendering algorithm combined with spherical harmonics is presented.

Obviously, the method proposed in this paper is not considering flying through or inside the clouds. Furthermore, our method is not very suitable for animation of clouds. In the near future, we would like to extend our algorithm to handle the dynamic simulation of cloud.

Acknowledgement:

The authors wish to thank the anonymous reviewers for their thorough review and highly appreciate the comments and suggestions, which significantly contributed to improving the quality of this paper.

This work is supported by National High Technology Research and Development Program of China (No. 2012AA011503), Project on the Integration of Industry, Education and Research of

Guangdong Province (No. 2012B090600008) and the pre-research project (No. 51306050102).

References:

[1] H. Qiu, K. Yang, Y. Chen, Survey on realistic simulation of cloud, *Journal of Computer Science*, Vol.38, No.6, 2011, pp. 14-19. (in Chinese)

[2] J. Schopik, J. Simons, D. S. Ebert, C. Hansen, A real-time cloud modeling, rendering, and animations system, *In Proc. Eurographics'03*, Leuven, Belgium, 2003, pp. 160-166.

[3] T. Nishita, Y. Dbashi, E. Nakamae, Display of clouds taking into account multiple anisotropic scattering and sky light, *In Proc. SIGGRAPH'96*, New Orleans, USA, 1996, pp. 379-386.

[4] N. Wang, Realistic and fast cloud rendering, *Journal of Graphics Tools*, Vol.9, No. 3, 2004, pp. 21-40.

[5] X. Y. Hu, B. Sun, X. H. Liang, An improved cloud rendering method, *In Proc. Image and GRAPHICS*, Xi'an, China, 2009, pp. 835-858.

[6] Y. Dobashi, K. Kaneda, T. Okita, T. Nishita, A simple, efficient method for realistic animation of clouds, *In Proc. Eurographics'00*, Brno, Czech Republic, 2000, pp. 19-28.

[7] R. Miyazaki, S. Yoshida, Y. Dobashi, A method for modeling clouds based on atmospheric fluid dynamics, *In Proc. Pacific Graphics'01*, Tokyo, Japan, 2001, pp. 363-372.

[8] M. J. Harris, W. V. Baxter, T. Scheuermann, Simulation of cloud dynamics on graphics hardware, *In Proc. Eurographics'03*, Sarajevo, Yugoslavia, 2003, pp. 92-101.

[9] M. J. Harris, Real-time cloud simulation and rendering, *PhD Dissertation*, University of North Carolina, 2003.

[10] J. Stam, Stable fluids, *In Proc. SIGGRAPH'99*, Los Angeles, USA, 1999, PP. 121-128.

[11] G. Y. Gardner, Visual simulation of cloud, *Computer Graphics*, Vol.19, No.3, 1985, pp. 279-303.

[12] M. J. Harris, A. Lastra, Real-time cloud rendering, *Computer Graphics Forum*, Vol.19, No.3, 2001, pp. 76-84.

[13] R. Miyazaki, Y. Dobashi, T. Nishita, A fast rendering method of clouds using shadow-view slices, *In Proc. CGIM'04*, Hawaii, USA, 2004, pp. 93-98..

[14] A. Bouthors, F. Neyret. S. Lefebvre, Real-time realistic illumination and shading of stratiform clouds, *In Proc. Eurographics Workshop on Natural Phenomena*, Vienna, Austria, 2006, pp. 1-10.

[15] A. Bouthors, F. Neyret, N. Max, E. Bruneton, C. Crassin, Interactive multiple anisotropic scattering in clouds, *In Proc. I3D'08*, Redwood City, USA, 2008, pp. 173-182.

[16] O. Elek, T. Ritschel, A. Wilkie, H. P. Seidel, Interactive cloud rendering using temporally-coherent photon mapping, *In Proc. Graphics Interface*, Toronto, Canada, 2012, pp. 141-148.

[17] K. Riley, D. Ebert, M. Kraus, Efficient rendering of atmospheric phenomena, *In Proc. Eurographics Symposium on Rendering*, Norkoping, Sweden, 2004, pp. 375-386.

[18] R. Miyazaki, Y. Dobashi, T. Nishita, Simulation of cumuliform clouds based on computational fluid dynamics, *In Proc. Eurographics'02*, Saarbruecken, Germany, 2002, pp. 405-410.

[19] Y. Dobashi, T. Yamamoto, T. Nishita, Interactive rendering of atmospheric scattering effects using graphics hardware, *In Proc. ACM Siggraph/Eurographics on Graphics Hardware*, Saarbruecken, Germany, 2002, pp. 99-107.

[20] R. Green, Spherical harmonics lighting: the gritty details, *In Proc. Game Developers*, San Jose, USA, 2003, pp. 1-47.

[21] T. Malzbender, Fourier volume rendering, *ACM Transactions on Graphics*, Vol.12, No.3, 1993, pp. 233-250.

[22] T. Totsuka, M. Levoy, Frequency domain volume rendering, *In Proc. SIGGRAPH'93*, Anaheim, USA, 1993, pp. 271-278.