International Journal of Innovative Computing, Information and Control
Volume **8**, Number **6**, June **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 4407–4420

USING INSTANCE FOR LARGE SCALE VOLUMETRIC CLOUDS RENDERING IN REAL-TIME

Gang Wang[1], Zhenzhou Ji[1] and Zexu Zhang[2,*]

[1]School of Computer Science and Technology
[2]Deep Space Exploration Research Center
School of Astronautics
Harbin Institute of Technology
No. 92, West Da-Zhi Street, Harbin 150001, P. R. China
gmkwangg@gmail.com; jizhenzhou@hit.edu.cn
*Correspondence Author: zexuzhang@hit.edu.cn

Received March 2011; revised November 2011

Abstract. *In this paper we discuss instance as a novel approach to render large scale volumetric clouds in real-time. Firstly, a layered geometry model based on LOD (Level of Detail) was proposed. Instance technique was applied to the model, which effectively improves the performance of rendering volumetric cloud. Secondly, the 2D textures were used to modulate of cloud appearance, cloud shadow, city glow, which provides additional flexibility. Thirdly, the lighting such as multiple forward scattering was computed in advance. The results were stored in a 3D texture, which simplifies the lighting computation at runtime. Finally, the algorithm was implemented by means of GPU (graphics processing unit). The algorithm proposed is suitable for rendering the cloud covering part or all of the sky, which was applied to a helicopter flight simulator successfully.*

Keywords: Volumetric rendering, Real-time rendering, Texture mapping, Geometry model, Lighting computation

1. Introduction.

Cloud is such a type of natural phenomena which is computationally expensive to render, especially for clouds cover part or all of the sky. To render such a large scale volumetric cloud in real-time, the following issues must be addressed: the first is simulation, which includes cloud dynamics, shading and illumination, such issues are always computationally intensive; the second is modeling, a variety of geometry models have been proposed which often involved a large number of vertices and primitives; the third is rendering, the methods must be flexible enough to simulate detail animation of clouds. Furthermore, special effects related to clouds should be taken into account. Therefore, simple and efficient methods which maintain the visually convincing results are required. This paper proposes such methods rendering vivid large scale volumetric cloud in real-time which address most of the above issues. We assume that cloud particles are static relative to each other even when the wind blows them. These assumptions enable us to simplify the computation of cloud lighting.

This paper is focused on rendering of large scale volumetric clouds in real-time. The rest of the paper is organized as follows. In Section 2, we describe related works. Section 3 describes the shading of clouds. In the next three sections, we describe cloud modeling, rendering details and relative special effects. The last two sections are the results and conclusions.

2. Related Works.

Much research work has been done on the simulation of volumetric clouds. [1] has divided the research results into three categories: (1) simplifying physics of cloud simulation and modeling, (2) photorealistic or special effects rendering, (3) efficient or real-time rendering. The brief review on the simulation, geometry modeling, and rendering of clouds is given as follows.

There are two categories to simulate the gaseous motion like clouds. One is to simulate the physical process of fluid dynamics. The other is a heuristic approach.

Physical methods use numerical solutions of computational fluid dynamics equations. Generally, the result of numerical simulation is cloud density distribution in 3D space, which can be rendered in various ways. In the work of [2-5], a staggered grid discrimination of the velocity and pressure equations is proposed. In the above papers, pressure, temperature and water content are defined at the center of voxels while velocity was defined on the faces of the voxels. Stam [6] developed a fast simulation method by simplifying fluid dynamics and demonstrated the real-time animation of smoke on a high-end workstation, but the rendering of clouds was not discussed. Moreover, the phenomenon known as phase transition was not taken into account. Kajiya et al. [7] developed a method of the phase transition effects with digital solution of computational dynamics and ray tracing algorithm which was so complex as to require of computation time. Dobashi et al. [8] proposed a simplified method based on cellular automaton. However, it can only simulate cumulus due to its extremely simplified physical model. Physics-based cloud simulation has the virtue of visual convincing, but it was not suitable to cloud rendering in real-time for its expensive in computation.

Most of heuristic approaches include fractals [8-11], procedure modeling [12-17] and stochastic modeling [18] and so on. The methods are computation inexpensive and much easier to implement. However, users have to adjust many parameters by trial. A fast and simple method of large scale volumetric cloud rendering was proposed and applied in a flight games in [19]. The rendering of cloud in real-time was presented by means of the partial differential equations of fluid motion, buoyant forces, water phase transitions and "flat 3D textures" simulations on the GPU [5]. Although the above two algorithms can deal with large scale clouds in real time, it is hard to simulate detail animation of clouds.

Since the cloud lacks of well defined boundary, it is fairly difficult to build a static geometry model. Therefore, geometry model of cloud is one of the important issues in the process of clouds rendering. The particle system and impostors [5,19], volume with meatball [8,20-23] and image based modeling [22,23] are three main methods.

The simplest rendering method is to use texturing algorithms to realize the rendering of clouds [10]. To be realistic, the scattering and absorbing effects within clouds, multiple scattering of light and skylight must be taken into account [8]. Ray tracing [7], photons map [24] and procedural texturing [10,25-27] are common techniques for such purposes. However, it is difficult for them to satisfy real-time clouds rendering.

In this paper, a novel layered geometry model based on LOD is proposed. Compared with the particle system and impostors, this model has the advantage of greatly reducing amount of the vertices. Then, the 2D texture is used to modulation of cloud appearance, cloud shadow and city glow. The lighting such as multiple forward scattering and skylight is computed in advance. The results computed are stored in a 3D texture, which simplifies the lighting computation at runtime. Moreover, coverage, density, wind and special effects such as shadow, city glow and lightning were also discussed.

3. **Lighting Model.**

The light passes through a cloud will be absorbed and scattered by the particles in clouds. The lighting model involves the computation of the transport and change of the light in the cloud. Lighting model used in this paper was based on the algorithm in [5]. We divided the algorithm into two steps in this study. The first step is pre-computation of illumination, and the computed results are stored in a 3D texture.

The 3D texture storing the lighting results can reduce the complexity of illumination computation of clouds at runtime. This texture can be stored on disk. Applications can load it before start to render. Thus this step can be offline. The second step is to use the results of first step to render the cloud. Lighting computation is one of the main bottlenecks of volumetric cloud rendering in real-time. Using pre-computed results can significantly improve the runtime performance.

Scattering illumination models simulate the emission and absorption of light by a medium as well as scattering through the medium. Our lighting model includes scattering model and skylight. Although we chose a LOD based layered geometry model representation for our clouds, it is important to note that both our shading algorithm and our fast rendering system are independent of the cloud representation, and can be used with any model composed of discrete density samples in space.

3.1. **Scattering model.** According to the direction of scattering, the scattering model can be divided into single scattering and multiple scattering models. The single scattering models simulate scattering through the medium in a single direction which usually points to the view point. Simulations based on the multiple scattering models usually sample N directions on a sphere. Each additional order of scattering will multiply N to the number of simulated paths. It is shown in [28] that the contribution of most of these paths is insignificant except the first and second order. [5] approximated multiple scattering only in the light direction (multiple forward scattering) and anisotropic single scattering in eye direction. We use of multiple forward scattering model instead of multiple scattering model.

3.1.1. *Multiple forward scattering.* Given the amount of incident light at each position P from direction l, the light $I(P,l)$ is composed of all direct light from direction l that is not absorbed by intervening particles, plus light scattered to P from other particles

$$I(P,\omega) = I_0(\omega)\cdot\exp\left(-\int_0^{D_P}\tau(t)dt\right) + \int_0^{D_P} g(s,\omega)\cdot\exp\left(-\int_s^{D_P}\tau(t)dt\right)ds \tag{1}$$

where $I_0(\omega)$ is the illumination intensity of light in direction of ω outside the clouds (i.e., skylight). s is path length of the incident light. $\tau(t)$ (In units of 1/length) is the extinction coefficient of the cloud at depth t. D_P is the depth of P in the cloud along the light direction.

$$g(x,\omega) = \int_{4\pi} r(x,\omega,\omega')I(x,\omega')d\omega' \tag{2}$$

represents the light from all directions ω' scattered into direction ω at the point x. $r(x,\omega,\omega')$ is the bi-directional scattering distribution function, which determines the percentage of light incident on x from direction ω' that is scattered in direction ω. The $r(x,\omega,\omega')$ can also expand to as follows:

$$r(x,\omega,\omega') = a(x)\cdot\tau(x)\cdot F(\omega,\omega') \tag{3}$$

where $a(x)$ is the albedo of the medium at x, and $F(\omega,\omega')$ is the phase function indicating the directional characteristics of scattering.

A full multiple scattering algorithm must compute this quantity for a sampling of all light flow directions. We simplify the approximation to compute only multiple forward scattering in the light direction, so $\omega = l$ and $\omega' = -l$ in Equation (3). There is only small solid angle γ around the forward direction. The value of γ is small enough to suppose that r and I are constant over γ. Therefore, Equation (2) reduces to

$$g(x,l) = r(x,l,-l) \cdot I(x,-l) \cdot \frac{\gamma}{4\pi} \tag{4}$$

If split the light path from 0 to D_P into discrete segments s_j $(j = 1, \ldots, N)$, where N is the number of cloud particles along the light direction from 0 to D_P. By approximating the integrals with Riemann Sums, we got the light intensity I_p at position P.

$$I_p = I_0 \cdot \prod_{j=1}^{N} e^{-\tau_j} + \sum_{j=1}^{N} g_j \prod_{k=j+1}^{N} e^{-\tau_k} \tag{5}$$

where I_0 is the intensity of light incident on the edge of the cloud. In discrete form of $g(x,l)$.

$$g_j = a_j \cdot \tau_j \cdot F(l,-l) \cdot I_j \cdot \frac{\gamma}{4\pi} \tag{6}$$

We assume that albedo and optical depth are represented at discrete samples (particles) along the path of light. In order to easily transform (5) into an algorithm that can be implemented in graphics hardware, we cast it as a recurrence

$$I_k = \begin{cases} g_{k-1} + T_{k-1} \cdot I_{k-1}, & 2 \le k \le N \\ I_0, & k = 1 \end{cases} \tag{7}$$

where $T_k = e^{-\tau_k}$ is the transparency of particle p_k. This equation simply says that starting outside of the cloud, as we trace along the light direction the light incident on any particle p_k is equal to the intensity of light scattered to p_k from p_{k-1} plus the intensity transmitted through p_{k-1} (as determined by its transparency, T_{k-1}).

3.1.2. *Eye scattering.* Eye scattering is a type of single scattering towards the viewer as in [8]. The recurrence for it is subtly different as follows:

$$E_k = S_k + T_k \cdot E_{k-1}, \quad 1 \le k \le N \tag{8}$$

This equation represents the energy of light E_k exiting any particle p_k is equal to the light incident on it that it does not absorb, $T_k \cdot E_{k-1}$, plus the light that it scatters, S_k, which can be replaced by $S_k = a_k \cdot \tau_k \cdot F(\omega, -l) \cdot \frac{I_k}{4\pi}$, where ω is the view direction.

3.1.3. *Phase function.* The characteristics of scattering are dependent on the size of particles and it is a powerful forward scattering. This means that the intensity of scattering is only 10% of that in 0° scattering angle when the scattering angle is greater than 10°. The phase function $F(\omega, \omega')$ mentioned above determines the distribution of scattering for a given incident light direction. The phase function is a sum of the phase function of particles in every scale is as

$$F(\theta) = \sum_{i=1}^{K} w_i \cdot F_i(\theta) \tag{9}$$

where K is the number of types of functions and w_i is the weight for phase function i. θ is the scattering angle. [29] improved the function of Henyey-Greenstein and the results were closed to the physics, which express as

$$F(\theta, f) = \frac{3(1-f^2)}{2(2+f^2)} \cdot \frac{(1+\cos^2\theta)}{(1+f^2-2f\cos\theta)^{3/2}} \tag{10}$$

where θ is the scattering angle. f is an asymmetry factor which is determined by the cloud condition and the wave length. Equation (10) considers Rayleigh scattering and Mie scattering uniformly. If $f = 0$, this function is equivalent to the Rayleigh scattering.

For scattering due to clouds the spectrum of scattering is not much influenced compared with that of air molecules. Thus color of clouds depends on spectrum of incident light.

3.2. **Scattering results.** Armed with recurrences (7) and (8) and a standard graphic API such as OpenGL, computation of cloud illumination is straightforward. Our algorithm is similar to the one presented by [5]. The key to the implementation is the use of hardware blending and pixel read back. Blending operates by computing a weighted average of the frame buffer contents (the destination) and an incoming fragment (the source), and storing the result back in the frame buffer. This can display as

$$C_{result} = f_{src} \cdot C_{src} + f_{dest} \cdot C_{dest} \tag{11}$$

If we let $C_{result} = I_k$, $f_{src} = 1$, $C_{src} = g_{k-1}$, $f_{dest} = T_{k-1}$ and $C_{dest} = I_{k-1}$, then we see that (7) and (11) are equivalent if the contents of the frame buffer before blending represent I_0. To solve the recurrence for a particle p_k, we must know how much light is incident on particle p_{k-1} beforehand. To do this, we employ pixel read back.

The procedure described by the pseudo code in Figure 1 is used to compute (7) and (8). As shown in the pseudo code, the light incident on p_k from a small solid angle γ is found by reading back the color of a small square of pixels centered on the projection of p_k before it is rendered. The number of pixels needed to sample γ is computed using the distance of p_k from the projection plane of the camera. I_k is computed by multiplying this result by the light intensity, and is used to compute multiple forward scattering in (7) and eye scattering in (8).

The results of the pseudo code are not used directly. Normal vectors with the length of which set to the results of the pseudo code is used as the final results. These vectors are easily created through similar methods creating bump mapping texture. This additional step allows us to approximately simulate directional light and time of day effects at runtime by directly using the results. In this paper, an 8 bit 3D texture with RGBA channels is used to store the results. The resolution of the texture is $256 \times 256 \times 32$. The negative values of the normal vectors are stored as $\mathbf{n}' = \frac{\mathbf{n}}{2} + 0.5$, where $\mathbf{n}'$ is the result stored in the 3D texture, $\mathbf{n}$ is the result of our algorithm. In fragment shader the vectors is further used to compute the intensity of a light source. Figure 2 shows slices of the final results stored in the 3D texture. The result of farthest layer of clouds from light source is stored in the first slice and up vector is (0, 0, 1).

```
source_blend_factor = 1;
dest_blend_factor = 1 − src_alpha;
texture_mode = modulate;
$l$ = direction from light;
$\omega = -l$;
view cloud from light source;
clear frame buffer to white;
sort particles in ascending order by distance to light;
$\gamma$ = solid angle of pixels to read.
foreach layer
  foreach particle $p_k$
    [$p_k$ has extinction $\tau_k$, albedo $a_k$, and alpha]
    compute # pixels $n_p$ to cover solid angle $\gamma$;
    read $n_p$ pixels in square around projected center of $p_k$;
    $i_k$ = Average intensity of $n_p$ pix;
    $\omega$ = $p_k$.position − view_position;
    $p_k$.color = $a_k \cdot \tau_k \cdot i_k \cdot \gamma/4\pi \cdot phase(\omega, l)$;
    $p_k$.alpha = $1 - e^{-\tau_k}$;
    render $p_k$;
  endfor
  read frame buffer back;
endfor
```

Figure 1. Pseudo code for cloud shading

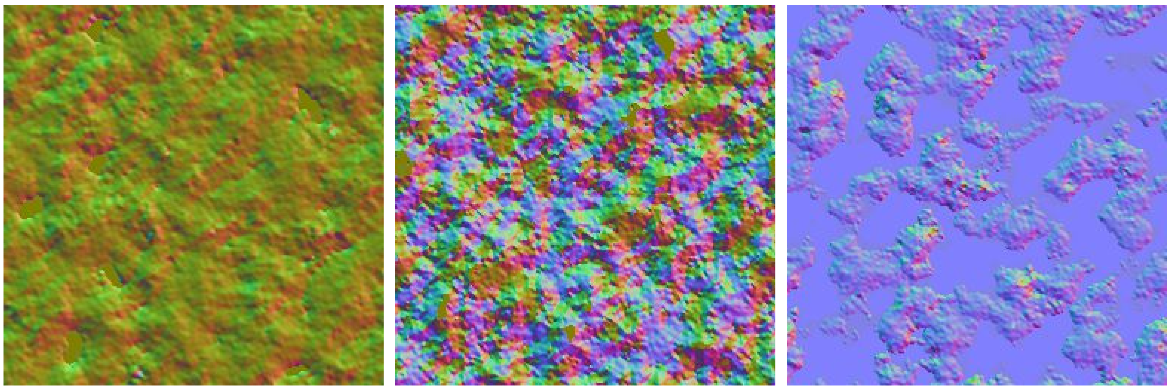

(a) The 4th slice (b) The 15th slice (c) The 27th slice

FIGURE 2. Lighting results stored in the 3D texture

Only intensity of the light source is stored, the color properties of which are set at runtime. This choice is important for runtime tuning these properties to get desired lighting effects such as skylight described bellow. Further more, this algorithm can be implemented offline.

3.3. **Skylight.** When it is on sunrise or sunset, the most awe-inspiring images of clouds would spread in the sky. These clouds are often illuminated by skylight – sunlight that is scattered by the atmosphere. This scattering is stronger than that of daytime. Algorithms described before are not well adaptable for our dynamic lighting. Since illumination has additive effects, this scene can be easily achieved through adding a light source and modifying the lighting model slightly. This additional light source is used to compute the illumination of clouds. This method is better than an ambient contribution, since it is directional and therefore would be more realistic.

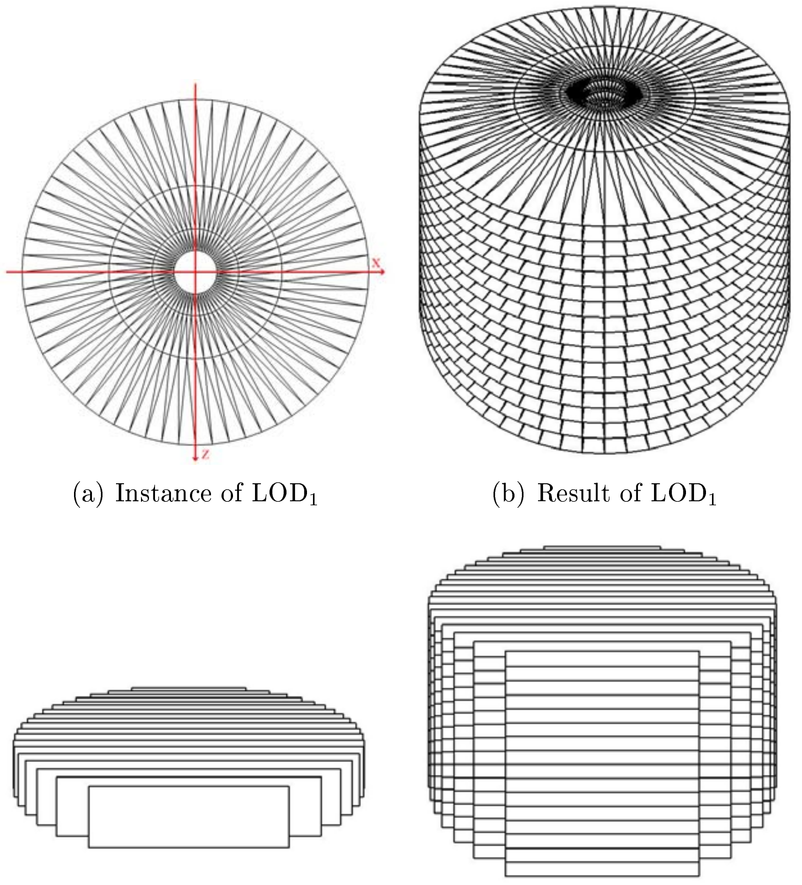
4. **Geometry Model.**

The geometry model is an important issue for rendering of volumetric cloud. The results and the performance of rendering are heavily depend on the choice of the geometry model. Topological structure and amount of vertices in the geometry model hold the consumptions of the CPU, memory and GPU. The rendering of volumetric cloud are applications consume resources critically. Thus real-time performance should take advantage of the geometry model as simple as possible on the premise of the realistic visual effects.

A novel layered geometry model based on LOD and particle system is proposed in this paper. This geometry model is very simple as described later and no further works need to be done at runtime compared with that of impostors ones.

A common use case in modern rendering engines is to be able to draw the same object, or groups of similar objects that share vertex data, primitive count and type, multiple times. Instance technique of modern GPU provides a means of accelerating such use cases while restricting the number of API calls, and keeping the amount of duplicate data to a minimum. Each of these objects has a constant unique instance ID in GPU shader. Our layered model especially adapted this technique. Taking advantage of instance, the model need only to store the data of one layer, data of other layers can be the duplication of this through instance.

A two level LOD geometry model was used in this paper. LOD_0 is at the center of LOD_1. Viewpoint is always at center of the geometry model on the horizontal. This means no LOD switch which decreases the management consumption of runtime.



(a) Instance of LOD$_1$ (b) Result of LOD$_1$

(c) Instance of LOD$_0$ (d) Result of LOD$_0$

FIGURE 3. Geometry model

LOD$_1$ is the low level of LOD which was a multilayer annulus model. The number of layers is related to the thickness of clouds and has the effects on the results and efficiency of rendering. The more layers the model has, the better the visual effects is. Though lesser layers improve the performance of the rendering, gaps will appear when the viewing angle is small. If the number of layers is N, the base height of the cloud is H_{base} and the $Thickness$ stands for the thickness of clouds, then, the height H_i of the ith layer is

$$H_i = H_{base} + i \cdot \frac{Thickness}{N-1} \quad i = 0, 1, \ldots, N-1 \tag{12}$$

Particle system or billboards was adopted in LOD$_0$. Differing from the methods in [5,19], the particles (billboards) are uniformly distributed in the area of LOD$_0$. The width W_p and the height H_p of a given particle are

$$\begin{cases} W_p = \begin{cases} 2r, & p_d = 0 \\ 2\sqrt{r^2 - (p_d - 0.5 \times p_s)^2}, & p_d > 0 \end{cases} \\ H_p = \max\left(\dfrac{Thickness}{N-1}, p_s\right) \end{cases} \tag{13}$$

where r is the radius of LOD$_0$, p_d is the distance of the particle center to the layer center of LOD$_0$, p_s is the space between two adjacent particles, $Thickness$ is the thickness of the cloud, and N is the number of layers.

The properties except the illumination and position of each particle are approximately constant thus are easy to implement on GPU [30]. According to the above geometry model and lighting model, the rendering results are shown in Figure 4.

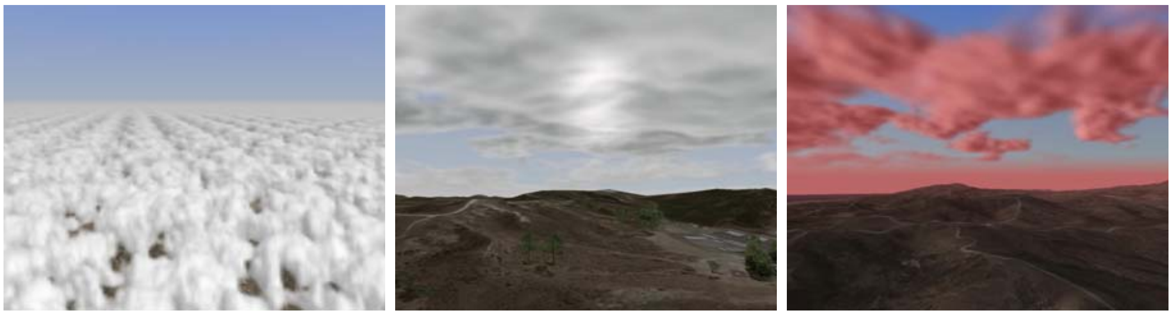

(a) Repeat wrap mode (b) Eye scattering (c) Skylight

FIGURE 4. Rendering results

Figure 4(a) is the results of the proposed methods. Because the wrap mode of the texture is repeating, tiling effect can be seen obviously. To emphasize the tiling effect, tile period of the 3D texture is set to 1600 database units instead of 16000 database units using in actual applications. Figure 4(b) shows the cloud scene of the eye scattering. Where the viewpoint is under the cloud layer and the light source is above the cloud layer. Figure 4(c) shows the cloud layer on sunrise or sunset, and the most awe-inspiring image of clouds are presented. These clouds are illuminated by the skylight or sunlight through the atmosphere.

5. **Cloud Rendering.**

With the lighting intensity stored in the 3D texture and the LOD based layered geometry model, the large scale volumetric clouds rendering is straightforward.

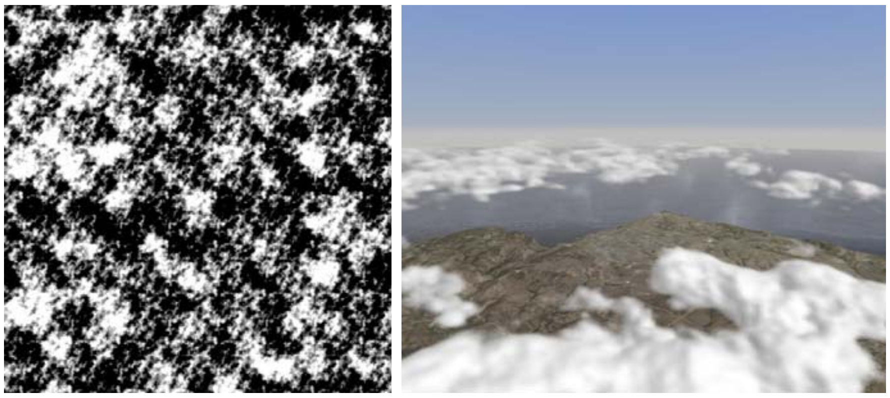
5.1. **Appearance.**

The cloud will have a uniform thickness appearance if only the geometry model and the pre-computed lighting results are applied without other efforts. The alpha channel of the 3D texture can be used to simulate the variation of the cloud thickness or areas where the cloud particles reside in. The tiling effect is inevitable as shown in Figure 4(a) due to the repeat wrap mode of the 3D texture. To solve this problem, a single channel 2D texture is used to modulate the alpha channel of the 3D texture. The result can be express as

$$\alpha = \max(T_{3D}.\alpha - T_{2D}.r, 0.0) \tag{14}$$

where $T_{3D}.\alpha$ is the alpha channel in 3D texture. $T_{2D}.r$ is the color of the single channel 2D texture. The 2D texture and the results modulated are shown in Figure 5(a) and Figure 5(b) respectively. Tile period of the 2D and 3D texture are 400000 and 16000 database units. Comparing Figure 4(a) with Figure 5(b), the tiling effect becomes inconspicuous.

The 2D texture used to modulate the alpha channel of 3D texture eliminating tiling effects acts a crucial role of the flexibility of our methods. Details of clouds such as coverage, density even wind speed effect can be simulated with the 2D texture. Effects related to the appearance of cloud such as cloud shadow can be implemented with this 2D texture too.



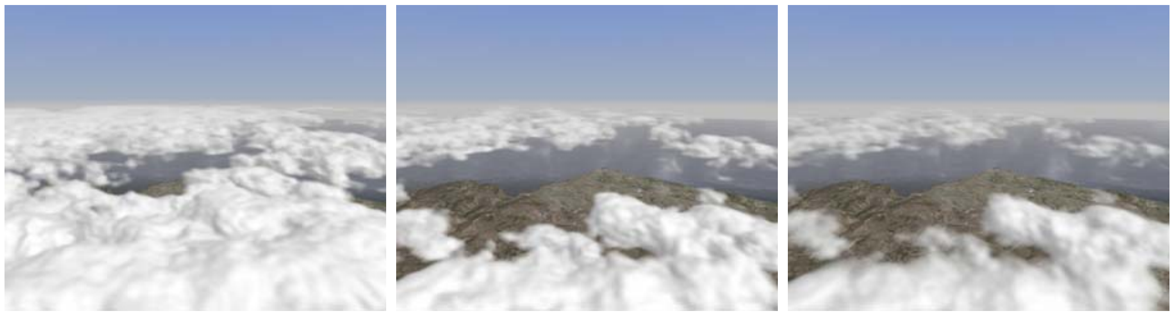
(a) 2D texture for modulation (b) Modulated results

FIGURE 5. Rendering results by 2D texture modulation

5.2. **Coverage.** The sky areas occupied by clouds are always varying. In order to simulate this phenomenon, coverage C is introduced to modulate α further

$$\alpha = \max(T_{3D}.\alpha - (T_{2D}.r + 1 - C), 0.0) \tag{15}$$

It can be seen that Equation (15) is in accord with Equation (14) when $C = 1$. The area covered by cloud will decrease when $C < 1$. The rendering results with $C = 1$ and $C = 0.6$ are shown in Figure 6(a) and Figure 6(b).



(a) $C = 1$ (b) $C = 0.6$ (c) $C = 0.6$ and $D = 0.3$

FIGURE 6. Rendering results with different properties

5.3. **Density.** Density is another important characteristic of clouds. A density factor of D is introduced to describe it. Equation (15) therefore changed into

$$\alpha = D \cdot \max(T_{3D}.\alpha - (T_{2D}.r + 1 - C), 0.0) \tag{16}$$

The density of clouds will be changed if $D < 1$. Figure 6(c) shows that cloud density is decreased than that of Figure 6(b).

5.4. **Eye scattering.** Using the algorithm described in Section 3.1.2 to compute eye scattering is physical accurate. When position of light source or viewpoint is changed, the result should be recomputed. To simplify the computation of eye scattering at runtime, we use the following

$$I_e = I_{e0} \times \max\left(0, \frac{dot(\mathbf{n}_{pe}, l) - \cos\varphi}{1 - \cos\varphi}\right) \times \left(f_e \cdot clamp\left(\frac{\Delta H}{Thickness}, 0, 1\right) + 1.0\right) \tag{17}$$

where I_e is the result of the eye scattering intensity. I_{e0} is the max intensity of the eye scattering. $\mathbf{n}_{pe}$ is the normalized vector of a point in the cloud converted to the eye space. l is the direction of light source. φ is the desired scattering angle. f_e is the proportionality coefficient. ΔH is the height difference of a point in the cloud and the bottom height of the cloud in world space. $Thickness$ stands for the thickness of clouds.

5.5. **Wind effect.** The wind is caused by the atmospheric pressure difference of areas. Therefore, the effects of wind on clouds are various, such as the position, the height and the appearance. Only the motion of winds on the horizontal direction is considered in this study. The height of clouds is kept in constant. Also, only the effects on the 2D and 3D texture coordinate are taken into account. The texture coordinate $T_c(t)$ at a given time t is

$$T_c(t) = T_{c0} + t \times \mathbf{v}_w \tag{18}$$

where T_{c0} is the start value of $T_c(t)$ which is relative to the tile period of textures. $\mathbf{v}_w$ is the velocity vector of wind. Since the 2D and 3D texture coordinate could be computed respectively, the changes of the clouds' position and appearance etc caused by wind can be well simulated.

6. **Special Effects.** The following special effects are not necessities of volumetric cloud rendering. Scenes with these effects are more realistic than the ones without. These effects are treated on ad-hoc ways. As described below, the special effects can be obtained through simple ways. This is further demonstrating the flexibility of methods proposed in this paper.

6.1. **Shadow of clouds.** Most of the sunlight pass through the clouds is absorbed by the particles and cast shadow on terrain. It is more realistic with shadows in simulation scenes. Since the formation of shadows related with the absorbing ability of particles in clouds, a 2D texture is built as shadow texture T_S for simplifying the calculation. This texture is related to the alpha channel of the 3D texture mentioned above. According to the shadow texture and the modulated texture, the factor S for the calculation of shadows is defined as follows:

$$S = 1 - D \cdot \max(T_S.r - (T_{2D}.r + 1 - C), 0.0) \tag{19}$$

The shadow on the terrain would achieve through the multiplication of S to the rendering results of the scene. Moreover, the shadows and the covered area of clouds would have proper location relationship. The rendering results are shown in Figure 7(a).

6.2. **City glow.** The city glow is caused by the particles in fog, cloud or pollution that scattering various illumination of city which often appears in the night. The area yielded the glow can be stored in the other color channels of the 2D modulation texture, such as the green channel. Assume ΔH is the height difference of a point in the cloud and the bottom height of the cloud in world space. The $glow$ of a point can be calculated as follows:

$$glow = T_{2D}.g \times \left(1 - clamp\left(\frac{f_c \cdot \Delta H}{Thickness}, 0.0, 1.0\right)\right) \tag{20}$$

where $clamp$ is a clamp function and f_c is proportionality coefficient. Their values can be chose based on the thickness of clouds. $Thickness$ stands for the thickness of clouds. The rendering result is shown in Figure 7(b).

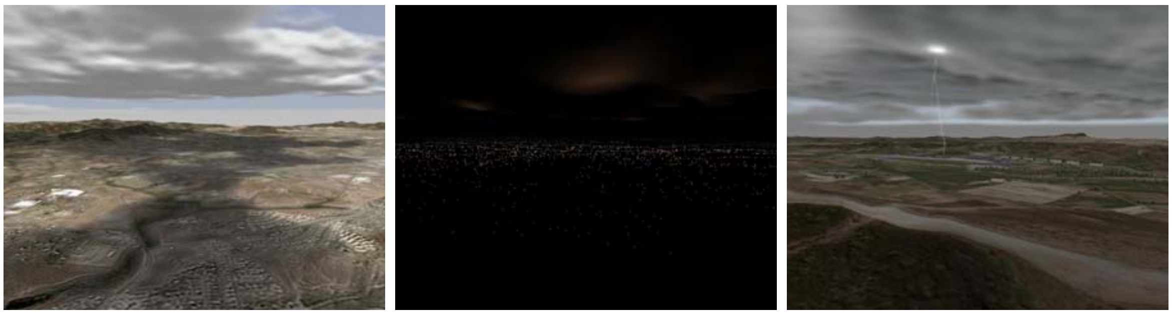

(a) Shadow of clouds (b) City glow (c) Lightning

FIGURE 7. Special effects

6.3. **Lightning.** Lightning is a discharge phenomenon between clouds with different charge and it is a common natural phenomenon in rainy day. Dobashi et al. [28] studied the rendering of lightning taking the scattering into account. Though their result was more physical accurate, the frame rate was too low for real-time rendering. In this paper, the lightning is simply treated as a point light. Suppose the intensity of lightning is I_l and the distance of a point in the cloud to the center of lightning is D_l, I_{P0} is the light intensity of a point P in cloud without lightning, the light intensity I_P at point P with lightning is as the following

$$\begin{aligned} I_P &= I_{P0} + \Delta I_{Pl} \\ &= I_{P0} + \frac{I_l}{D_l^2} \end{aligned} \tag{21}$$

where ΔI_{Pl} is the light intensity change caused by lightning. The rendering result is shown in Figure 7(c).

7. **Results.** The algorithms rendering volumetric clouds are mainly implemented on GPU. The simulating computer is a standard PC with a Pentium D 3.0 GHz CPU, 1GB memory and an NVIDIA GTX260+ display card. The screen resolution is set to 1280×1024.

Vertex program and fragment program are used to implement the rendering. Figure 8 shows the pseudo code of these programs. Instance is used to create the layered geometry model of this paper.

On the simulating computer, the field of view is set to $60° \times 45°$. The tile period of the 3D texture and the 2D texture are set to 16000 and 400000 database units respectively. When the far clip plane and cloud extent are set to 20000 database units, the average frame rate is up to 390 fps. If the far clip plane and cloud extent are adjusted to 100000 and 90000 database units respectively, the average frame rate is still exceeding 350 fps.

The high frame rates are mainly relying on the pre-computation of illuminations and the layered geometry model. The layered model is suitable utilizing instance technique of modern GPUs. Compared with the previous works, normal vectors with which length set to the lighting intensity stored in 3D texture is capable of approximately deal with the position change of light source; LOD based layered geometry model and using of instance technique significantly decrease memory consumption and simplify geometry modeling; Using the 2D texture to modulate alpha channel of the 3D texture further improves the rendering flexibility. The rendering results with different viewpoints' position are shown in Figure 9.

A real-time large scale volumetric cloud rendering in a certain helicopter simulator was realized based on the method proposed above. Using the experimental PC above, with the screen resolution of 1400×1050, the simulation can be at a steady interactive frame rate of 60 fps, in which a complex scene with world wide database, kinds of 3D entities and special effects, etc. were rendered.

8. Conclusion.

The method of rendering large scale volumetric cloud in real-time is studied in this paper. Our contributions include:

(1) The lighting computed in advance was stored in a 3D texture, which significantly simplifies the complexity of lighting computation at runtime. The skylight and eye scattering, etc. were also studied.

(2) A novel layered geometry model based on LOD (Level of Detail) was proposed. This method did not require accurate geometry model of clouds and could improve performance by means of instance technique.

(3) The appearance of clouds was modulated by 2D texture and the tiling effect was eliminated. The 2D texture further provided the flexibility of our algorithm. Moreover, the coverage, density and the effects of wind were also studied. The special effects such as shadows, city glow and lightning were also discussed.

The experimental results indicate that the method proposed can render large scale volumetric cloud with complex scene in the extent of 90km and achieve vivid visual effects.

Methods and algorithms of this paper can be used in flight simulators' visual systems, games and applications which need large scale volumetric clouds rendering in real-time.

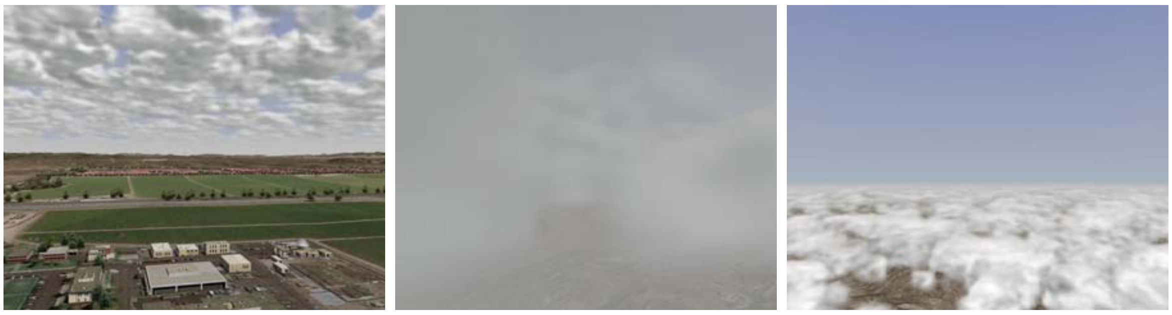
```
// Compute layer height using of InstanceID
   $height = H_{base} + InstanceID \times Distance\ between\ layers$
if lightning
  get vector from lightning position to the vertex $litnvec$;
endif
if cityglow
  compute cityglow parameter;
endif
if eye scattering
  compute eye scattering parameter
endif
compute texture coordinate of 3D texture consider wind;
compute texture coordinate of 2D texture consider wind;
convert vertex position from world space to homogeneous space;
```

(a) Pseudo code of vertex program

```
compute $\alpha$ with Coverage and Density;
// Compute light intensity $lit$
$lit = dot(light\ position, (T_{3d}.rgb - 0.5) * 2)$;
if lightning
  $lit += lightning\ intensity / dot(litnvec, litnvec)$;
endif
if cityglow
  $lit += T_{2D}.g \times cityglow\ parameter$;
endif
if eye scattering
  $lit += I_{e0} \times eye\ scattering\ parameter$;
endif
$color = lit \times light\ color$;
if sky light
  $skylit = dot(sky\ light\ position, (T_{3d}.rgb - 0.5) * 2)$;
  $color += skylit \times skylight\ color$;
endif
```

(b) Pseudo code of fragment program

FIGURE 8. Pseudo code of GPU



(a) Viewpoint under cloud (b) Viewpoint in cloud (c) Viewpoint above cloud

FIGURE 9. Rendering results of large scale scene

Acknowledgements. This work was funded by National Basic Research Program of China (973 Program) under Grants 2012CB720000. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] H. S. Liao, J. H. Chuang and C. C. Lin, Efficient rendering of dynamic clouds, *Proc. of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*, Singapore, pp.19-25, 2004.

[2] N. Foster and D. Metaxas, Modeling the motion of a hot, turbulent gas, *Proc. of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, pp.181-188, 1997.

[3] R. Fedkiw, J. Stam and H. W. Jensen, Visual simulation of smoke, *Proc. of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, pp.15-22, 2001.

[4] M. Griebel, T. Dornseifer and T. Neunhoeffer, Numerical simulation in fluid dynamics: A practical introduction, *Society for Industrial and Applied Mathematics*, Philadelphia, PA, USA, 1998.

[5] M. J. Harris, W. V. Baxter III, T. Scheuermann and A. Lastra, Simulation of cloud dynamics on graphics hardware, *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, San Diego, CA, USA, pp.92-101, 2003.

[6] J. Stam, Stable fluids, *Proc. of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, pp.121-128, 1999.

[7] J. T. Kajiya and B. P. V. Herzen, Ray tracing volume densities, *Proc. of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, Minneapolis, MN, USA, pp.165-174, 1984.

[8] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita and T. Nishita, A simple, efficient method for realistic animation of clouds, *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, New Orleans, LA, USA, pp.19-28, 2000.

[9] R. Voss, Fourier synthesis of Gaussian fractals: 1/f noises, landscapes, and flakes, *Proc. of the 21th Annual Conference on Computer Graphics and Interactive Techniques: Tutorial on State of the Art Image Synthesis*, Detroit, Michigan, 1983.

[10] G. Y. Gardner, Visual simulation of clouds, *Proc. of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, San Francisco, CA, USA, pp.297-304, 1985.

[11] T. Nishita, T. Sirai, K. Tadamura and E. Nakamae, Display of the earth taking into account atmospheric scattering, *Proc. of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, Anaheim, CA, USA, pp.175-182, 1993.

[12] N. L. Max, Light diffusion through clouds and haze, *Graphics and Image Processing*, vol.13, no.3, pp.280-292, 1986.

[13] D. S. Ebert and R. E. Parent, Rendering and animation of gaseous phenomena by combining fast volume and scanline A-buffer techniques, *Computer Graphics*, vol.24, no.4, pp.357-366, 1990.

[14] D. S. Ebert, W. E. Carlson and R. E. Parent, Solid spaces and inverse particle systems for controlling the animation of gases and fluids, *The Visual Computer*, vol.10, no.4, pp.471-483, 1990.

[15] D. S. Ebert, Volumetric modeling with implicit functions: A cloud is born, *ACM SIGGRAPH Visual Proceedings: The Art and Interdisciplinary Programs of SIGGRAPH'97*, pp.147-147, 1997.

[16] D. S. Ebert, Procedural volumetric cloud modeling and animation, *SIGGRAPH'99 Course Notes*, vol.26, no.5, pp.1-52, 1999.

[17] N. Max, R. Crawfis and D. Williams, Visualizing wind velocities by advecting cloud textures, *Proc. of the 3rd Conference on Visualization'92*, Los Alamitos, pp.179-184, 1992.

[18] J. Stam, Stochastic rendering of density fields, *Proc. of Graphics Interface'94*, Banff, Alberta, pp.51-58, 1994.

[19] N. Wang, Realistic and fast cloud rendering in computer games, *Proc. of ACM SIGGRAPH Sketches & Applications*, San Diego, CA, USA, 2003.

[20] J. Stam and E. Fiume, A multiple-scale stochastic modeling primitive, *Proc. of Graphics Interface'91*, Calgary, Alberta, pp.24-31, 1991.

[21] J. Stam and E. Fiume, Depiction of fire and other gaseous phenomena using diffusion processes, *Proc. of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, pp.129-136, 1995.

[22] Y. Dobashi, T. Nishita, H. Yamashita and T. Okita, Using metaballs to modeling and animate clouds from satellite images, *The Visual Computer*, vol.15, no.9, pp.471-482, 1998.

[23] Y. Dobashi, Y. Shinzo and T. Yamamoto, Modeling of clouds from a single photograph, *Computer Graphics Forum*, vol.29, no.7, pp.2083-2090, 2010.

[24] H. W. Jensen and P. H. Christensen, Efficient simulation of light transport in scenes with participating media using photon maps, *Proc. of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, Orlando, FL, USA, pp.311-320, 1998.

[25] D. S. Ebert, Procedural volumetric cloud modeling and animation, *Proc. of ACM SIGGRAPH'00 Course Notes*, vol.25, no.5, pp.1-55, 2000.

[26] K. Perlin, An image synthesizer, *Proc. of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, San Francisco, CA, USA, pp.287-296, 1985.

[27] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin and S. Worley, *Texturing & Modeling*, Academic Press Inc, Orlando, FL, USA, 1998.

[28] Y. Dobashi, T. Yamamoto and T. Nishita, Efficient rendering of lightning taking into account scattering effects due to clouds and atmospheric particles, *Proc. of the 9th Pacific Conference*, pp.390–399, 2001.

[29] W. M. Cornette and J. G. Shanks, Physical reasonable analytic expressions for the single-scattering phase function, *Applied Optics*, vol.31, no.16, pp.3152-3160, 1992.

[30] S. Drone, Real-time particle systems on the GPU in dynamic environments, *ACM SIGGRAPH Course Note*, pp.80-96, 2007.