An Efficient Approach to Real-Time Sky Simulation

Kun Yang[1,2], Qicheng Li[1,2,3], Zhangjin Huang[1,2], Jia Wang[4], Guoping Wang[1,2]
[1]*Dep. of Computer Science & Technology, Peking University, Beijing, China*
[2]*Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, China*
[3]*IBM China Research Lab, Beijing, China*
{yangk, lqc, hzj, wgp }@graphics.pku.edu.cn
[4]*Network Center, DaLian Polytechnic University, Dalian, China*
wangjia@dlpu.edu.cn

Abstract

Real-Time sky rendering plays a great role on visualization in many 3D applications. It is difficult to simulate a realistic sky without considering the effect of clouds and the scattering effect of the atmosphere. In this paper, we use a physical scattering model to simulate the sky, and a Perlin Noise texture to create highly realistic clouds. Different effects of the sky are also generated by combining the lighting model with various atmospheric constituents.

1. Introduction

Rendering realistic sky is one of the major challenges in the simulation of outdoor scenes. Although many of the simulation methods have considered the light effects, they usually neglect the influence of the environment itself, which means the atmosphere, clouds, fog, smoke and other gaseous constituents.

Two approaches to sky rendering have traditionally been used in games and similar applications. In the static approach, one or more photographs are textured onto a 3D model of the sky. For example, a cloud texture can be overlaid on top of a blue sky texture. The cloud texture can then be rendered offset over time to produce an animation. While the resulting sky can be very convincing given a high quality photograph, the lack of dynamics is a significant drawback. The second approach involves creating clouds procedurally. Procedural cloud generation can be divided into two categories, volumetric and planar, based on how the clouds are represented. In this paper, we propose an efficient approach which combines the light scattering model with the simulation of clouds. In the end of this paper we give some examples generated by our algorithm.

978 -1-4244-1579-3/07/$25.00 © 2007 IEEE

2. Related work

The realistic sky simulation has been an active area of research for many years. Numerous approaches have been proposed for realistic sky simulation. Nishita [1] brought out a rough experience-based sun light model which separated the sky area into strap light sources and using different CIE models in different environments. Inakge [2] and Irwin [3] considered the physical model which is based on the Rayleigh and Mie scattering. Preetham [4] presented a day light analysis model which has considered the atmosphere effect on the light. With the rapid development of GPU, Dobashi [5] and Hoffman [6] started to render the sky model using GPU.

One of the most important sky scene simulations is the cloud. The appropriate clouds can increase the realistic feeling of the sky scene dramatically. The usual approaches of cloud simulation include the following two models:

1) **The physical model**, which uses the true physical and mathematical methods to simulate the cloud. The typical approach is proposed by Nishita [7]. Also Nelson Max [8] adopted volumetric rendering to enhance the effect of simulation. Their methods can produce highly realistic clouds, but the calculation is too complex to be real-time.

2) **The texture model**, which employs one or more textures to combine and rotate to simulate the flow of cloud. The typical methods include Gardner [9], Perlin [10] and Ebert [11]. The texture model is widely used in the real application of virtual environment because of the real-time rendering requirement.

3. Sky model

This section describes a high realistic sky dome model. To increase the realistic feeling of the sky scene, the flowing clouds effect is considered. A rendering method based on the texture map combined with a light scattering model is proposed to create a highly realistic sky scene.

3.1. Sky dome model

The sky dome model [14, 15] is the skeleton of this sky model, which will create a hemi-sphere to cover the whole scene (see Figure 1).

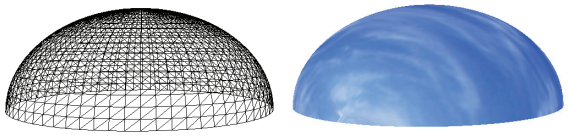


Figure 1. The sky dome model

We define the sky dome model by the polar system with the center of sphere as the origin and r as the radius (see Figure 2):

$$x = r\cos(\phi)\sin(\theta)$$
$$y = r\sin(\phi)$$
$$z = r\cos(\phi)\cos(\theta)$$

Here, ϕ is the latitude and θ is the longitude: $0 \le \phi \le \pi/2, 0 \le \theta \le 2\pi$. Then we can get every point's coordinates (x, y, z) according to its latitude and longitude.

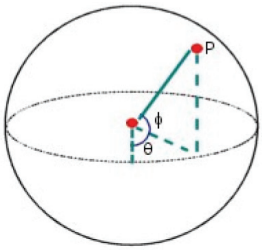


Figure 2. The polar system

After construction of the mathematical model of the sky dome, we will sample points on the hemi-sphere, using the discrete sampling on the latitude and longitude coordinates. The distance of latitude sampling is $\Delta\phi$ and the longitude one is $\Delta\theta$. Then the total number of the sample points is $(360/\Delta\theta)(90/\Delta\phi)$. Sampling more points will produce a higher rendering precision and higher realistic feeling. However, too many sampling points cost much more spending in the computer resource, and the improvement of effects is not so obvious. We choose the sampling distances as $\Delta\theta = \Delta\varphi = 15$, which means only 144 vertex points will be used to describe our sky dome model.

3.2. The simulation of clouds

After building the geometry skeleton, we use sphere texture mapping to mask the cloud texture on the sky dome, which will increase the realistic feeling of sky.

It's difficult to simulate clouds in 3D environments because the light will be reflected, refracted and scattered when it go through the clouds. We use texture distribution method [12] to realize the flow of clouds. The core idea of this method is to scroll sky texture at a random speed in both u and v directions (see Figure 3). To make it more realistic, we build multi-layer cloud textures and different layer moves at a different speed, then we mix these layer into one texture.

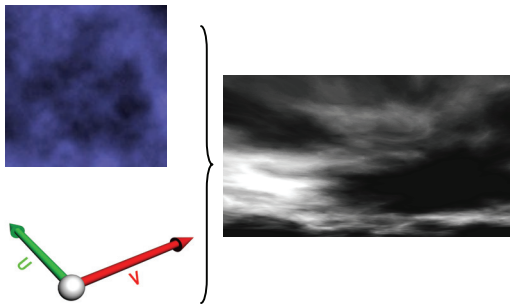


Figure 3. The sketch map for the simulation of clouds

Now we use the Perlin Noise [10] to generate the cloud texture. Also Pallister [13] proposed an idea of combining four noise textures to create clouds. To build a Perlin Noise, we need a noise function and an interpolation function. The noise function is a random number generator which will generate a random number according to an integer number. The interpolation function is always selected from the linear interpolation, cosine interpolation and cubic interpolation.

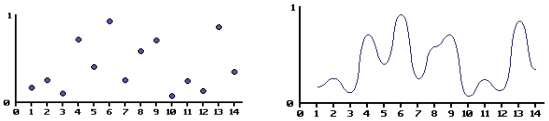


Figure 4. The distribution of noise

To create a smooth noise function, we will construct some new noise functions by changing the amplitude and frequency as follows:

$$\begin{cases} frequency = 2^i \\ amplitude = persistence * e^i \end{cases}$$

Here, *persistence* is a factor which means the amplitude will change with the frequency. The final function is generated by summing up all of these noise functions.

The final 2D result of Perlin Noise is illustrated in Figure 5.

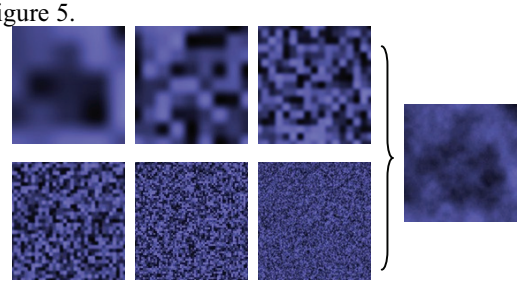


Figure 5. The generation of texture using the Perlin Noise

3.3. The scattering model

Now the cloud effect is only the gray effect (see Figure 6). Some simulations use the color table map to simulate the sky scene, but the effect is some kind of robotistic and not so realistic. We will use a precise scattering model to enhance the realistic feeling of the sky scene.

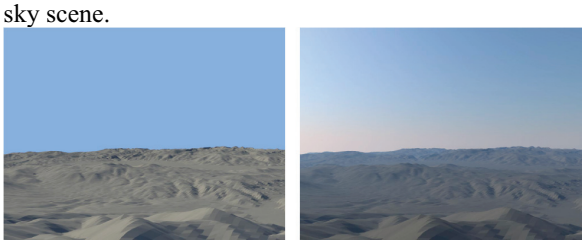


Figure 6. Comparison of not using and using scattering models [4]

The traditional outdoor rendering method always uses the following fog model in hardware to simulate the effect:

$$L = L_0(1-f) + C_{fog} f \tag{1}$$

where L_0 is the intrinsic color which is always a light blue color, C_{fog} is the color of fog, and f is the fog factor. This model has an obvious shortcut: the simple formula can't express the real complex physical model and the intensity does't change in different view directions. We import the model provided by Hoffman [6] which is more physically accurate to this work.

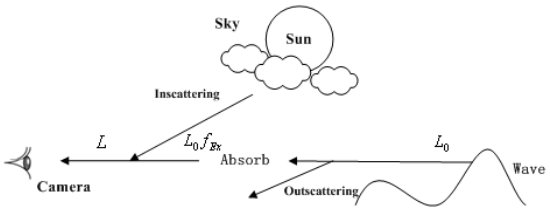


Figure 7. Sketch map of the scattering model

Based on the physical model, we can get the equation as follows:

$$L = L_0 f_{Ex} + L_{In} \tag{2}$$

where f_{Ex} is the term of attenuation, L_{In} is the incidence ray from the sky. As shown in Figure 7, one can see that L_0, which is the reflex ray with an intrinsic color, is absorbed by the atmosphere partly and deviates from primary direction as a result of outscattering. These two parts compose the attenuation of ray. However, the ray from sky and other directions can radiate along the direction of the reflex ray, and this is the inscattering ray.

The equation of absorption and outscattering is

$$L(x) = L_0 e^{-\beta_{Ab} x} e^{-\beta_{Sc} x} = L_0 e^{-(\beta_{Ab} + \beta_{Sc}) x} \tag{3}$$

where β_{Ab} is the factor of absorption, β_{Sc} is the factor of scattering. β_{Ab} and β_{Sc} are based on the wavelength of ray, and can expressed as a triple vector in RGB system. In practice, β_{Ab} is set to the value suggested in [4], and β_{Sc} is the sum of the Rayleigh coefficient and Mie coefficient.

However, there is also a phenomenon which can add light to a ray. Such case is in-scattering, where the light, which was originally headed in a different direction, is scattered into the path of a light ray and adds to the ray's radiance. To compute the inscattering ray, we define a scattering phase function $\phi(\theta)$, which gives the probability of scattered light going in the direction θ, and θ is the angle between the incidence ray and the scattering ray. Since $\phi(\theta)$ is a probability function, when integrated over the entire sphere of directions the result is 1. Hence we have the following equation:

$$\frac{\Delta L}{\Delta x} = -\beta_{Ex} \cdot L + E_{Sun} \cdot \phi(\theta) \cdot \beta_{Sc} \tag{4}$$

where $\beta_{Ex} = \beta_{Ab} + \beta_{Sc}$, and E_{Sun} is a scalar factor which represents the energy of sun. From (4), we have

$$L(x) = E_{Sun}(L_0 e^{-\beta_{Ex} x} + \frac{\beta_{Sc}\phi(\theta)}{\beta_{Ex}}(1 - e^{-\beta_{Ex} x})) \quad (5)$$

Since β_{Sc} is the sum of the Rayleigh coefficient and Mie coefficient, the function $\phi(\theta)$ depends on the particles doing the scattering. For particles much smaller than the wavelength of light ($r < 0.05\lambda$), $\phi(\theta) = \phi_R(\theta)$ is based on the Rayleigh coefficient. For larger particles, $\phi(\theta) = \phi_{HG}(\theta)$ is based on the Henyey / Greenstein function which is an approximation to the Mie coefficient.

$$\phi_R(\theta) = \frac{3}{\pi}(1 + \cos^2\theta) \quad (6)$$

$$\phi_{HG}(\theta) = \frac{(1-g)^2}{4\pi(1 + g^2 - 2g\cos\theta)^{3/2}} \quad (7)$$

Here, g is the eccentricity of the Mie coefficient. We use an approximation to express it, which scales with $1/\lambda^2$. Hence,

$$\beta_{Sc}\phi(\theta) = \beta_R \frac{3}{\pi}(1+\cos^2\theta) + \beta_M \frac{(1-g)^2}{4\pi(1+g^2-2g\cos\theta)^{3/2}} \quad (8)$$

where β_R and β_M are the Rayleigh coefficient and Mie coefficient, respectively. Substituting (8) into (5), we can work out the scattering ray.

3.4. Implementation and results

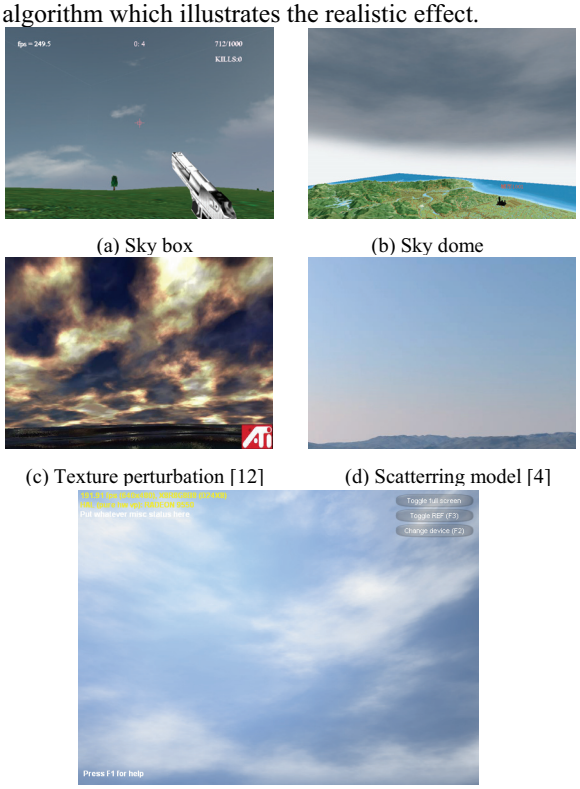
The algorithm is mainly composed by the following two steps:

(1) **Pre-Process step** will use Perlin Noise to generate the clouds texture. We don't need to consider too much on the time-consuming of pre-process, since we can export a little larger texture. The final texture resolution is 512×512, and it spends us about one minute.

(2) **Real-time Render step** includes the modeling of sky dome, texture mapping and disturbances and scattering calculation. The texture turbulences and scattering result calculation will take advantage of GPU because GPU has powerful computing ability in vector and matrix calculation.

Using a computer with AMD Sempron™ 2500+, 512M RAM memory and ATI 9600 Series (with 128M RAM), the frame refresh rate is more than 200 at the resolution 640*480.

We use several pictures to compare the effect of different approaches. Figure 8a is the sky box model which can't generate the flow of clouds easily. Figure 8b is a simple sky dome model which has clumsy and robotistic flow effect. Figure 8c is generated by ATI Research[12] which was using the texture perturbation. The color of sky is simply generated by the specified color so it's not so real. Figure 8d is the result of the scattering model in [4], which mainly focused on the simulation of scattering itself without considering the true environment. And Figure 8e is the result of our algorithm which illustrates the realistic effect.



(a) Sky box (b) Sky dome

(c) Texture perturbation [12] (d) Scatterring model [4]

(e) Our algorithm

Figure 8. Comparison of different approaches

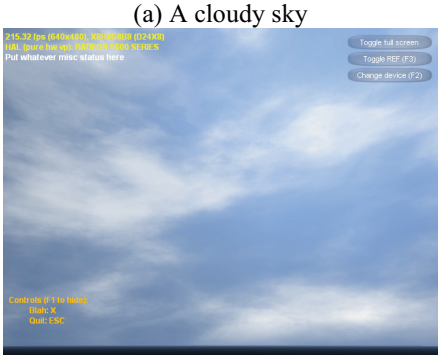
The thickness of clouds, the intensity of light and others parameters are also provided in our algorithm to generate different environment effects as shown in Figure 9.

4. Conclusion and future work

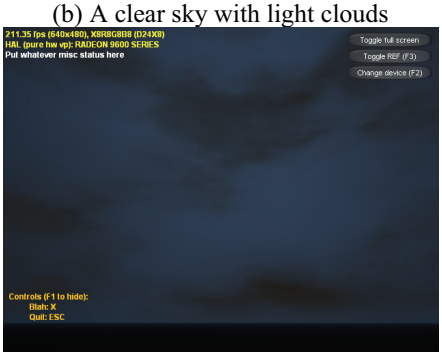
We present an approach to the high realistic sky simulation. The models of Perlin Noise cloud texture and physical scattering are imported into our algorithm, and good effects are generated.

There is still much room for improvement. Firstly, the impact of the clouds on the light can be modeled more accurately. Secondly, the reflection effect can be introduced in the physical model to make the environment more realistic especially considering different time and different environment of a day.

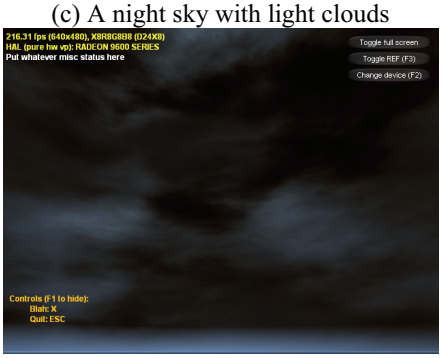
(a) A cloudy sky



(b) A clear sky with light clouds



(c) A night sky with light clouds



(d) An overcast sky with heavy clouds

Figure 9. The different environment effects generated by our algorithm

Acknowledgements

The research was supported by the 973 Program of China (2004CB719403), 863 Program of China (2006AA01Z334), NSF of China (60573151), and China Postdoctoral Science Foundation(20060390359).

References

[1] Nishita T, Nakamae E., Continuous tone representation of three-dimensional objects illuminated by sky light. In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, 1986, 20(3): 125-132.

[2] Inakage M., Volume tracing of atmospheric environments. *The Visual Computer*, 1991, 7(2): 104-113.

[3] Irwin J., Full-spectral rendering of the earth's atmosphere using a physical model of Rayleigh scattering. In *Proceeding of 1996 Eurographics UK Conference*, London, 1996, pp. 103-115.

[4] A.J. Preetham, P. Shirley, B.E. Smit., A Practical Analytic Model for Daylight. In *Proceedings of ACM SIGGRAPH 99*, 1999, pp. 91-100.

[5] Dobashi Y, Yamamoto T, Nishita T., Interactive rendering of atmospheric scattering effect s using graphics hardware. In *Proceedings of the ACM SIGGRAPH / EUROGRAPHICS Conference on Graphics Hardware*, Barcelona, Spain, 2002, pp. 99-107.

[6] Naty Hoffman, Nathaniel, Arcot J. Preetham., Rendering Outdoor Light Scattering in Real Time. In *Proceedings of Game Developer Conference 2002*, California, USA, 2002, pp. 85-89.

[7] Nishita T, Dobashi Y, Kaneda K. et al., Display method of the sky color taking into account multiple scattering. In *Proceedings of Pacific Graphics'96*, Pohang, South Korea, 1996, pp. 117-132.

[8] Max Nelson, Optical Models for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1995, 1(2): 99-108.

[9] Geoffrey Gardner Y., Visual Simulation of Clouds. In *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, 1985, 19(3): 297-303.

[10] Perlin, K., An Image Synthesizer. In *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, 1985, 19(3): 287-296.

[11] David E S., Volumetric Modeling with Implicit Functions: A Cloud is Born. In *Proceedings of ACM SIGGRAPH 97*, 1997, pp. 147-153.

[12] John Isidoro, Guennadi Riguer, Texture Perturbation Effects. In *ShaderX: Vertex and Pixel Shader Tips and Tricks*, WordWare Publishing, Minnesota, 2002, pp. 91-108.

[13] Pallister, K., Generating Procedural Clouds Using 3D Hardware. In *Game Programming Gems 2*, Mark Deloura ed., Charles River Media, 2001, pp. 463-473.

[14] Timothy Roden, Ian Parberry, Clouds and stars: efficient real-time procedural sky rendering using 3D hardware. In *Proceedings of the 2005 ACM SIGCHI International Conference*, Valencia, Spain, Jun. 2005, pp. 434-437.

[15] Georg Zotti, A Sky Dome Visualisation for Identification of Astronomical Orientations. *Information Visualization*, 2006, 5(2): 152-166.