Procedia Computer Science

Volume 66, 2015, Pages 697–704

YSC 2015. 4th International Young Scientists Conference on Computational Science

The Method for Real-time Cloud Rendering

Ksenia Mukhina and Alexey Bezgodov
ITMO University, Saint-Petersburg, Russia
mukhinaks@gmail.com, demiurghg@gmail.com

Abstract
Modeling of realistic clouds always was one of the most important problems in creating any virtual scenes outside. It has always been an extremely valuable feature for great variety of applications: from flight simulators or meteorological software to computer games especially with an open world. In this work the algorithm of rendering flat clouds in real-time is presented. The hemispherical grid was designed to fill natural placement of clouds. Tools for high-quality visualization of stratocumulus clouds were created. The model of light scattering through clouds is described. This approach allows rendering realistic clouds evolving through time at high frame rates.

Keywords: cloud rendering, light scattering, real-time rendering, cloud animation, cloud modeling

1 Introduction

The vast majority of all information the person receives through a visual communication channel that is why there is no doubt that visualization has become the one of the most powerful and effective way for data representation. In the modern world, the good graphical representation is a necessary part of any scientific project in different areas from liberal arts to computational science (Zudilova-Seinstra & Adriaansen, 2009). For the last two decades, the technologies made a huge qualitative leap forward, and scientific visualization was not an exception. However, nowadays there are still a few challenges for scientists from all over the world. One of such issues is rendering realistic clouds and a meteorological environment. For varied kinds of applications: from flight simulators to computer games it has always been a valuable and one of the most desired features.

With fast improvement of technologies and particularly the hardware, previous methods are not capable to ensure the necessary level of cloud visualization. They spend unacceptably much time on computations, or their execution requires specific hardware. Either way still there is not a single solution for high quality real-time visualization of clouds. To address this issue, this paper proposes a new method for modeling clouds. The basic idea of the suggested method is to create stratocumulus clouds by generated noise texture and to compute all light effects using shaders on the graphics processing unit (GPU).

Selection and peer-review under responsibility of the Scientific Programme Committee of YSC 2015

doi:10.1016/j.procs.2015.11.079

2 Related works

The one of the first attempts to visualize fuzzy objects was taken place by Reeves (Reeves, 1983) in 1983. He suggested using particle systems to represent the difficult structures like smoke, water, fire, and clouds. Particles systems are a compact volume representation and directly support many cloud modeling techniques.

All approaches to cloud modeling may be classified as simulation-based or procedural. The procedural methods are generally used to model a cloud's density field. There are few processes to fill the cloud volumes: use volumetric implicit functions (Heinzlreiter, Kurka, & Volkert, 2002), fractals (Hu, Xiao, & Hu, 2012) or noise. The most common way to create a cloud environment is the Perlin noise (Perlin, 1985). It allows to get a realistic sky with soft transitions between clouds and to avoid repetitive patterns in generated textures. The Perlin noise is used in various of cases such as landscape generation, textures of fire, smoke, wood and much more.

In 2001, Harris and Lastra (Harris & Lastra, 2001) presented an algorithm for shading and rendering realistic clouds using fluid simulation techniques. Their method simulates multiple scattering and anisotropic single scattering in the light direction and the view direction, respectively. Harris continued his work (Harris & Baxter, 2003) in 2003 by implementing this algorithm on GPU. At the same time, Schpok and Simons (Schpok, Simons, Ebert, & Hansen, 2003) developed an interactive system for rendering and animation of clouds. They used the combination of different noise filters to obtain a high-quality visualization. However, their application was designed for central processing unit (CPU) and worked with low frame rates only.

A new model for light transport was proposed by Bouthors and Neyret (Bouthors, Neyret, Max, Bruneton, & Crassin, 2008) in their work devoted to anisotropic scattering in clouds. Their suggestion consists of estimating the energy transport from the illuminated cloud surface to the rendered cloud pixel for each separate order of multiple scattering.

In the article (Elek, Ritschel, Wilkie, & Seidel, 2012) dedicated to interactive cloud rendering Elek and Ritschel presented a novel algorithm for simulation of light transport in clouds. They also introduced a technique to store and reconstruct the angular illumination information by exploiting properties of the standard Henyey–Greenstein function, namely its ability to express anisotropic angular distributions with a single dominating direction. Later they expanded (Elek, Ritschel, Dachsbacher, & Seidel, 2014) light transport simulation on the heterogeneous participating media. The steps of the proposed method are physically plausible. The employed empirical heuristics introduce a certain bias but allow to make design decisions that result in a real-time performance on contemporary graphics hardware.

In 2014 Egor Yusov (Yusov, 2014) developed a high-performance algorithm for real-time rendering of clouds which are evolving through time. To obtain the high quality visualization and to keep calculation time acceptable he used the four-dimensional look-up table of precompute values of optical depth and single and multiple scattering for each camera's rotation angle and light direction. However, this implementation relies on peculiar properties of Intel's graphic devices, and there are significant quality losses on the hardware with different configurations.

All the works mentioned above deal with a three-dimensional model of cloud. The rendering algorithms are extremely time-consuming and require many resources for computations and as a result this leads to reducing of quality for such visualizations in order to keep appropriate rendering time. That is why we concentrate on a two-dimensional approach of cloud representation to demonstrate the results of our method. It allowed us to achieve the high quality of visualization and keep the appropriate time of computations at the same time. Proposed algorithm does not depend on the chosen way of representation and can be extended on a three-dimensional model. However, the number of calculations increases sharply in this case. Thus, our method meets several certain requirements. First, the rendering time should be no longer than 1 or 2 milliseconds to keep the high level of frame rate. All computations are performed on GPU to ensure that. And second, we consider that camera is

located on zero-level above the ground and cannot be moved in the vertical direction. It is a good approach for a wide range of applications from naval simulators or meteorological applications (Haase, et al., 2000) to computer games.

3 Method overview

In broad meteorological terms, a cloud can be defined as a visible mass of liquid droplets or crystals of ice suspended in the atmosphere above the ground. Due to its nature, visualization of realistic atmosphere entails the number of difficulties. There are four major types of clouds which are also divided into a number of subtypes, and different cloud types require different volume representations, lighting and rendering techniques. Computation of all mentioned points, especially lightning, requires a lot of time. As far as every cloud consists of billions of particles we should use a model for approximate calculations to provide an appropriate result and do not overload system at the same time.

Despite the fact that nowadays more than ten types of clouds are distinguished depending on their shape and altitude (Houze Jr, 2014) and this number can be even increased by scientific society, in our approach we focused on rendering stratocumulus type of clouds only. We choose this way due to the fact that stratocumulus is the most widespread type on the low level across the Europe (Warren, Hahn, London, Chervin, & Jenne, 1986) and Russian Federation particularly. It has the greatest average coverage among any other low cloud types; average annual amount of this type is 12 per cent (Warren, Eastman, & Hahn, 2007).

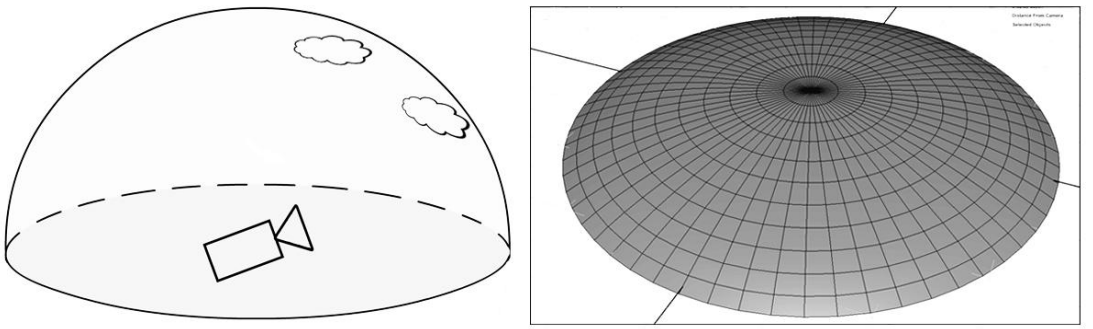

Figure 1. Key idea (left); three-dimensional model of skydome (right).

3.1 Key aspects

To obtain a natural look of the sky we consider the number of key ideas. First, clouds are approximately projected on the surface of the hemispherical disc instead of a plane. The grid is represented by three-dimensional FBX model of oblate in vertical direction unit sphere (Figure 1, right). The model is loaded to the scene; the center of the sphere is fixed in origin point. Then, the hemisphere is scaled according to user-defined size and clouds are placed on it. After that, any other objects are loading in conformity with their positions. Thus, with camera movements we can observe the different clouds in different parts of sky and sky covers the whole scene from all possible camera's points of view.

Second, we store all necessary data in texture to ease access to it and keep the fast speed of read-out. For instance, the normal vectors of cloud shape are extracted from noise texture, after that they are recalculated and rotated according scene coordinates.

Figure 2. High-quality images generated by our method with three layers of clouds (left) and one layer (right)

Third, the cloud environment consists of multiple layers of clouds. In a single iteration of grid rendering, we can draw maximum three layers of clouds: one for every color channel. If for some reasons it will be decided to display more layers, it is easy to duplicate the grid several times. However, it should be noticed that rendering even a small number of cloud layers results in high-quality photorealistic images and for the majority of cases with cloudiness from 0 oktas to 6 oktas sufficient quality of visualization could be achieved with three layers of clouds (Figure 2). Okta is a unit used in meteorology as measure of cloud cover, equal to one eighth of the sky area (an overcast sky is 8 oktas, and a half-obscured sky is 4 oktas).

Fourth, in our approach clouds are evolving through time. Different layers can move with different velocities. Cloud's motion considered as uniform with constant velocity. Every particle of cloud simply shifts in space according to given offset of noise texture. The value of velocity for every single layer of clouds can be defined in a compute shader.

4 Cloud rendering

Directly the rendering of clouds consists of several stages. Primarily, the color of the whole sky is computed depending on the time of the day. Perez method (Perez, Seals, & Michalsky, 1993) was used for sky modeling; in this approach the color of the sky is changed according to sun position from lowest at sunset or sunrise to the highest at midday. Sky is divided into several sky elements and color of each part is calculated independently. After that, the color of the whole sky is computed by using previous calculations. This approach allows us to obtain a smooth transition between the colors and natural look of the sky.

$$lv = f(\xi, \gamma) = \left[1 + a \cdot exp\left(\frac{b}{\cos(\xi)}\right)\right] \times [1 + c \cdot exp(d \cdot \gamma) + e \cdot \cos^2(\gamma)], \quad (1)$$

where lv - relative luminance, ξ is the zenith angle of the given sky element, γ is the angle between the sun position and considered sky element. The coefficients a, b, c, d, and e are user-defined parameters. The parameter a is responsible for a blackening ($a > 0$) and a brightening ($a < 0$) of the horizon area. The coefficient b adjusts the luminance gradient close to the horizon. The coefficient c is related to intensity of the solar aureole. The coefficient d modulates the size of the area near the sun and the parameter e accounts for the relative intensity of backscattered light. The luminance of the considered sky element, Lv may be obtained by using this equation

$$Lv = Lvz\frac{f(\xi,\gamma)}{f(0,Z)}, \quad (2)$$

where Lvz - zenith luminance and Z is the zenith angle of the sun (solar zenith angle).

The initial noise texture matching the cloud shape is generated by using two-dimensional Perlin noise (Perlin, 1985). All necessary information (e.g. normal vectors, binormal vectors, tangent vectors) are packed in texture and send to GPU. After that, the texture can be modified to satisfy the different weather conditions from clear sky to completely grey. The different level of cloudiness (Figure 3) could be achieved by changing parameter in the pixel shader; it can be varied from 0 for the cloudless sky to 1 for overcast. To obtain the high overcast of clouds we take more information from initial noise texture, for clear sky only the brightest parts of the texture are used.

Figure 3. Sky with different level of overcast.

On the next stage, the process divides on light scattering, self-shadowing, and lighting. Every part is computed independently on GPU. The first step is the computation of the cloud illuminance. This is essential to light up clouds by the sun correctly. It is carried out by using the color of the sun, sun position, and the normal vector of the cloud, which was extracted from the noise texture and after that it was modified according to world coordinates. In the pixel shader, we reconstruct the view and sun rays and compute their dot product.

$$LT = (n, S)L, \tag{3}$$

where n – cloud surface approximate normal, S – normalized sun position, and L – sunlight intensity.

The next stage is the calculation of light scattering through the cloud. As long as we have not true three-dimensional cloud made of particles, information about cloud density is also stored in noise texture. Computation of light scattering is the key factor to obtain natural looks of the cloud. Light scattering is computing by using following equation

$$LS = [k_1 + k_2(v, S)]^m L, \tag{4}$$

where L – sunlight intensity, S – normalized sun position, v – vector of view direction, k_1, k_2, m – user defined constants, where m is responsible for the size of the lighted area, k_1 and k_2 account for transmitted light.

The last step is self-shadowing which computes by this formula:

$$SH = (1 - \sum_i C_i)^m, \tag{5}$$

where C_i – the shadowed element of cloud, m – user defined constant. To get smooth transitions between well-lighted and shadowed parts of the cloud, the sun position is used, and every shadow is radial blurring in that direction.

Finally, we merge results of these calculations together in pixel shader and resulting color of every point of the cloud is obtained by using formula (6).

$$F = (LS + LT)SH, \tag{6}$$

5 Results and discussions

We implemented the described algorithm in C# and HLSL using the Fusion_project Framework (Bezgodov, Karsakov, Zagarskikh, & Karbovskii, 2015). To study the performance of our method, we used several test platforms. The first test platform was laptop powered by Intel(R) Core i7-4510U processor and GeForce GTX 850M integrated with Intel(R) HD. The second test platform was a powerful computer with Intel(R) Core i7-3930K and GeForce GTX 770. The third one is the device with following characteristics: Intel(R) Core i7 and GeForce GTX 470. All images on the test platforms were rendered in three options: standard resolution (800x600), HD (1280x720) and full HD resolutions (1920x1080). For our experiment, we calculated the average time of rendering for 500 frames on every platform. Also, we tested both graphic devices on the laptop.

As can be seen from the graph (Figure 4) the rendering time is increasing with the growth of image resolution. Despite this fact for the majority of devices rendering time is less than 2 milliseconds. The one exception is the less powerful graphic device on the laptop. This device is not capable of performing complicated computations; nevertheless in case of standard resolution rendering time satisfies our requirements (1.73ms). It means that for a huge amount of devices our method works sufficient fast with a high frame rate per seconds. For the laptop there was no big difference between rendering in standard or HD resolution (0.83 and 0.84 ms, respectively), however full HD images require slightly greater than 2 times more time for rendering (1.7 ms). Rendering time for the computer with GTX 470 has an upward trend, and it has the similar pattern for rendering HD and full HD images: for 1920x1080 images time is doubled. In contrast, there is no any trend for the device with GTX 770. The rendering time is almost constant for all kind of resolutions (0.65 ms for standard resolution, 0.83 and 0.87 for HD and full HD resolutions, respectively). It should be noticed that for this device rendering time was less than 1 ms in all cases.

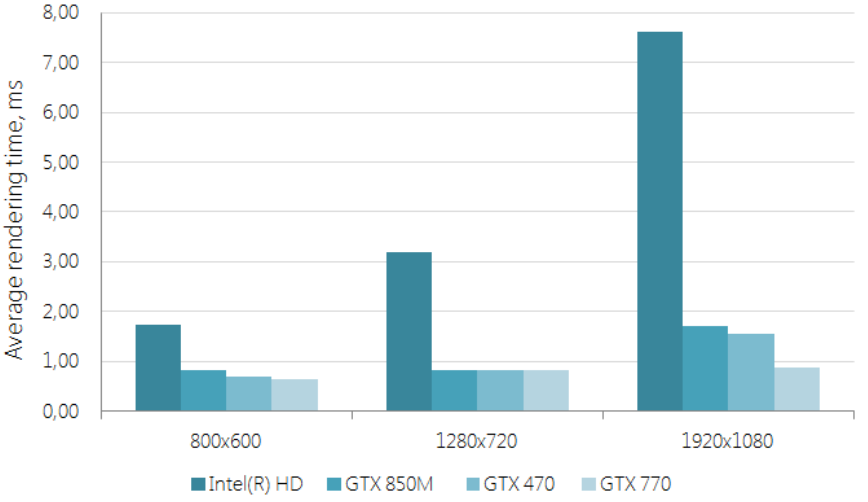

Figure 4. Comparing rendering time on test platforms.

6 Conclusion and future works

In this paper, we presented a new method for rendering realistic stratocumulus clouds. Our method computes light scattering, lighting, and self-shadowing inside the cloud in real-time. It also allows changing the level of cloudiness and satisfies the different sun positions (Figure 5). The clouds are forming and disappearing over time. Proposed algorithm provides a low level of resource consumption

and does not require more than 2 ms for visualization using desktop computer or powerful laptop. However, in some cases (for example, in our test for integrated graphic device) it demands more time for computations. The major reason of that is properties of used GPU. It could be resolved by reducing the resolution of generated images. Photorealistic images of clouds and sky can be received with using proposed method.

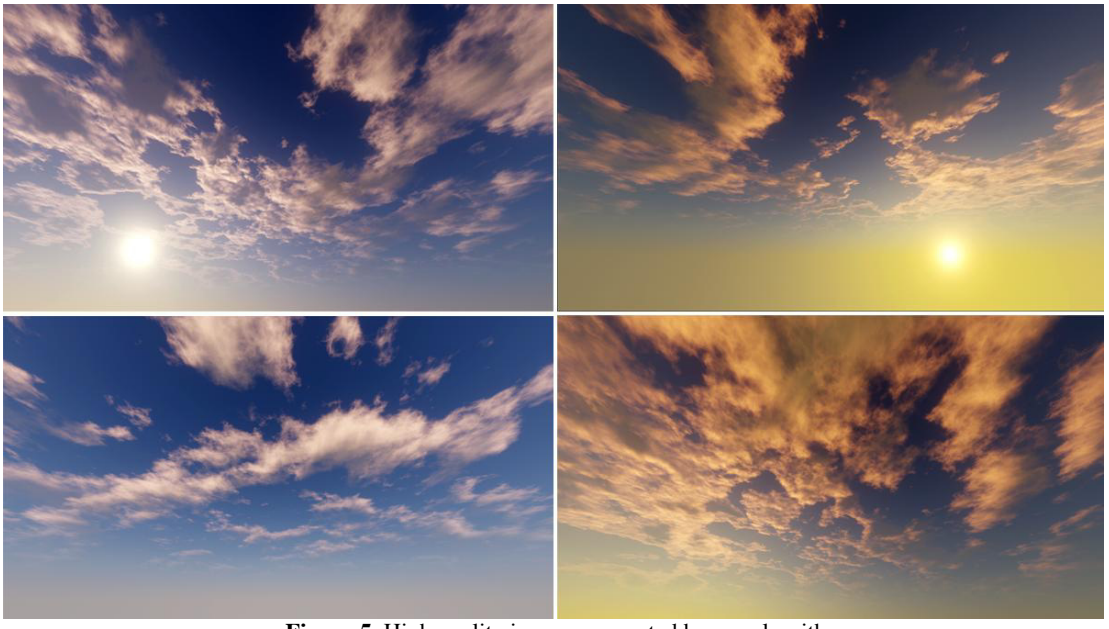

Figure 5. High quality images generated by our algorithm.

Still there are some directions to improve this work. Despite the fact, that two-dimensional approach for cloud rendering is plenty for the vast majority of applications, we can model truly three-dimensional clouds. Our method does not strongly rely on chosen method of cloud representation that means it can be easily implemented on three-dimensional clouds. The one of possible ways to achieve that is to use three-dimensional noise textures. Also, we are going to extend this model by adding other widespread types of clouds such as cirrus and cumulus. It can be obtained by generating textures with different initial parameters. In future research, we are planning to add different atmospheric effects and optical phenomena like rain, snow, rainbow, lightning and crepuscular rays, as well.

Acknowledgements

This paper is supported by the Russian Science Foundation, Project #14613 "Big data management for computationally intensive applications".

References

Bezgodov, A., Karsakov, A., Zagarskikh, A., & Karbovskii, V. (2015). The Framework for Rapid Graphics Application Development: The Multi-scale Problem Visualization. *Procedia Computer Science* (51), 2729-2733.

Bouthors, A., Neyret, F., Max, N., Bruneton, E., & Crassin, C. (2008). Interactive multiple anisotropic scattering in clouds. *Proceedings of the 2008 symposium on Interactive 3D graphics and games - SI3D '08* , 173-.

Elek, O., Ritschel, T., Dachsbacher, C., & Seidel, H.-P. (2014). Principal-Ordinates Propagation for real-time rendering of participating media. *Computers & Graphics* , 28-39.

Elek, O., Ritschel, T., Wilkie, A., & Seidel, H. P. (2012). Interactive cloud rendering using temporally coherent photon mapping. *Computers and Graphics* , 1109-1118.

Haase, H., Bock, M., Hergenröther, E., Knöpfle, C., Koppert, H., Schröder, F., et al. (2000). Meteorology meets computer graphics - a look at a wide range of weather visualizations for diverse audiences. *Computers and Graphics (Pergamon) , 24* (3), 391-397.

Harris, M. J., & Lastra, A. (2001). Real-Time Cloud Rendering. *Computer Graphics Forum* , 76-85.

Harris, M., & Baxter, W. (2003). Simulation of cloud dynamics on graphics hardware. *Graphics hardware* , 92-102.

Heinzlreiter, P., Kurka, G., & Volkert, J. (2002). Real-time Visualization of Clouds. *WSCG '2002: Short Communication Papers: The 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2002, 4.-8. February 2002 Plzeň: Conference proceedings* , 43-50.

Houze Jr, R. A. (2014). *Cloud dynamics* (Vol. 104). Academic press.

Hu, D. B., Xiao, J. B., & Hu, J. H. (2012). Real-Time Simulation of Dynamic Cloud Based on Fractal. *Applied Mechanics and Materials , 220-223*, 2841-2845.

Perez, R., Seals, R., & Michalsky, J. (1993). All-weather model for sky luminance distribution - preliminary configuration and validation. *Atmospheric Sciences , 50*, 235-245.

Perlin, K. (1985). An image synthesizer. *ACM SIGGRAPH Computer Graphics , 19* (3), 287-296.

Reeves, W. T. (1983). Particle systems - a technique for modeling a class of fuzzy objects. *ACM SIGGRAPH Computer Graphics , 17* (3), 359-375.

Schpok, J., Simons, J., Ebert, D. S., & Hansen, C. (2003). A Real-Time Cloud Modeling , Rendering , and Animation System. *Eurographics SIGGRAPH* , 20-27.

Warren, S. G., Eastman, R. M., & Hahn, C. J. (2007). A survey of changes in cloud cover and cloud types over land from surface observations, 1971-96. *Journal of Climate , 20* (4), 717-738.

Warren, S. G., Hahn, C. J., London, J., Chervin, R. M., & Jenne, R. L. (1986). *Global distribution of total cloud cover and cloud type amounts over land*. {Washington Univ., Seattle (USA). Dept. of Atmospheric Sciences; Colorado Univ., Boulder (USA); National Center for Atmospheric Research, Boulder, CO (USA).

Yusov, E. (2014). High-Performance Rendering of Realistic Cumulus Clouds Using Pre-computed Lighting. *High-Performance Graphics 2014, Lyon, France, 2014. Proceedings* , 127-136.

Zudilova-Seinstra, E., & Adriaansen, T. (2009). *Trends in Interactive Visualization - State-of-the-art Survey*.