Realistic Simulation for Rainy Scene

Cheng Wang[1], Meng Yang[1, 2*], Xuemin Liu[1], Gang Yang[1]

[1] School of Information Science & Technology, Beijing Forestry University, Beijing 100083, China.
[2] The State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China.

* Corresponding author. Tel.: +86-10-62336392; email: yangmeng@bjfu.edu.cn
Manuscript submitted March 26, 2014; accepted September 8, 2014.

Abstract: Study of rain dynamic simulation is an important research topic in computer graphics. This paper proposes a dynamic simulation of realistic rain scene based on particle system and raindrops texture. This method dissects the principle of atomization phenomenon, and fuses the raindrops' depth texture and scene background to simulate a hazy atomization. In addition, it also introduces wind field and lightening in the rain scene so as to enhance sense of reality. Finally, the paper renders the whole scene by constructing the rain scene and applying ray tracing algorithm. The experimental results show that: the present method can simulate the rain dynamic realistically and efficiently, which is of a high practical value.

Key words: Particle system, raindrop texture, atomization effect, lightning simulation, ray tracing.

1. Introduction

Natural scene simulation has been one of the focuses and difficulties in computer graphics. Rain simulation is widely used in defense and military, television, advertising, meteorology, games and other fields. In military field, all-weather weapon tests must be conducted in a photo-realistic rain. For scene simulation in television and advertising fields, rain simulation plays an important role in realistic effects. In meteorology field, fast and realistic rain simulation is essential for weather forecasts and analyses. In the field of games, rain simulation must strike a balance between photorealistic and real-time generation.

However, on rainy days, large amount of raindrops with hazy surface would have some light reflection and refraction, which makes it quite hard to simulate the realistic rain scene. The enormous quantities of raindrops would be a heavy burden for computer calculation. In addition, the raindrop has many properties, such as size, mass, density, velocity and acceleration. All of these above make it quite hard to simulate rain scene in a realistic and efficient way. Therefore, how to improve the computer's running speed and reduce the resource consumption is a challenging work.

This paper aims to find an optimum agreement of reality, real-time performances and resource consumption. It simulates complex rain scene on the basis of the method of particle system. In the meantime, dynamic raindrops are simulated by studying the motion law of raindrops. In addition, it introduces wind field and lightning to simulating the rain scene, and also adopts the classic ray tracing algorithm to render rain scenes under different lighting conditions. The simulating process is sped up by making use of raindrop texture mapping. With low requirements for equipment, fast simulation speed and high-reality of graphing results, this method is very efficient.

In this paper, Section 2 focuses on introducing the raindrops simulation algorithm. Section 3 elaborates an

algorithm of raindrop dynamic simulation, including the raindrop's geometry structure, the calculation method of rain line length, the setting of raindrops texture, the real-time updating algorithm of particle, the processing of atomization, and the introduction of wind field and lightning. Section 4 introduces the construction and rendering method of rain scene. Section 5 describes and analyzes the experimental results in detail. At last, Section 6 summarizes the methods and makes a prospect of the future work.

2. Related Work

Recently, research on rain simulation mainly focuses on two aspects: one is the raindrop dynamics simulation, the other is the rendering of rain scene.

2.1. Raindrop Dynamics

Nowadays, there are many algorithms of dynamic raindrops simulation. Rousseau *et al.* [1] proposed a real-time method by using programmable graphics hardware to simulate the effect of the refraction inside the raindrops, and mapped the raindrops' continuous changes caused by optical properties to texture. Based on the physical characteristics of raindrops and visual principle, Wang *et al*. [2] focused on simulating the shape and intensity of moving raindrops in different situations. Choudhury [3] proposed a real-time frame to simulate optically-active raindrops, yet there existed few differences between them. Niniane [4] mapped textures onto a double cone centered about the camera. Wang *et al.* [5] proposed a hybrid approach that capitalized on a real footage of raindrops to extract their realistic appearance. By applying the method of Pre-computed Radiance Transfer (namely PRT) to particle system, they gained a more realistic rain scene rendering with regard to environment lighting. Although those methods of raindrop dynamics simulation could achieve real-time effect, the raindrop models are relatively simple.

Reeves *et al.* [6] proposed a particle system with numerous irregular particles randomly distributed to efficiently simulate complex scenes. The distribution of particles in natural rain scenes is similar to the normal distribution. Inspired by the randomness of cloud model proposed by Li *et al.* [7], a method using Ultra-entropy is introduced to modify the rain drops distribution to better simulate natural rain. In the particle swarm algorithm field where particle motion is more irregular, Qiu *et al.* [8] got a good result by giving each particle a fractal evolutionary process. This paper achieves a real-time effect as well as reality by applying particle system and texture mapping method to rain scene simulation.

2.2. Rain Scene Rendering

In the early days, based on the background color and the geometrical properties of raindrop, Starik *et al.* [9] added a rain mask to each video frame before being adaptively added to the background. Nakamae *et al.* [10] collected data of road surfaces, and proposed two models to simulate the wetting effect. One is a reflection model which factors weather condition in; the other is a light model with refraction and diffraction taken into consideration. To realize the real-time rendering of photorealistic rain scene, Garg *et al.* [11] used ray-tracing algorithm to generate a raindrop texture database in advance, saving a great amount of time at the expense of memory. All the methods described above did not consider the misty effect of rain scene.

Whitted *et al.* [12] proposed the ray-tracing algorithm to produce high fidelity images. Based on the ray-tracing method, Cook *et al.* [13] proposed a distributed approach to improve the classic ray tracing. Jiang *et al.* [14] rendered rain streaks with Monte Carlo ray tracing method, and then sped up the rendering process by adaptive sampling of raindrops. Gregory *et al.* [15] adopted a bidirectional path tracking algorithm. Lv [16] improved the photon mapping method. Since the ray-tracing method was time-consuming, many methods were applied to speed up the process, including GPU-based method [17], bounding-box-based method [18], KD-tree method [19], [20] and Octree method [21]. This paper rendered the rain scene with ray-tracing algorithm.

To generate realistic lightning, Kim [22] used adaptive grid to render the lightning, and used the jitter to improve the visual effect. La Pointe *et al.* [23] adopted volumetric data structure and L-system to generate lightning, and rendered the volume of lightning at the cost of huge memory.

3. Raindrop Dynamic Simulation

3.1. Geometrical Construction of Raindrop

A rain scene in nature has a high complexity. It could be billions of raindrops, each with different radius and shape under different-direction forces, which would also change subtly with the raindrop and the surrounding environment, and thus lead to acceleration variation in raindrop movement whose velocity, as a result, can hardly be described in one movement formula.

Inspired by the work of Reeves [6] and Li *et al.* [7], this paper abstracts rain scene into a system consisting of numerous simple voxels. That is, use particle system method to create rain scene. Besides, it also defines the properties of a single raindrop, including position, velocity, acceleration, life span, demise velocity, particle size and particle color, etc. The type of raindrops is defined based on the following pseudo code:

```
Struct Raindrop{
        Vector3   position;
        Vector3   velocity;
        Vector3   acceleration;
        float     lifetime;
        float     vanishing;
        float     size;
        float     color[3];
}
```

where *Vector3* is expressed as a user-defined class of three dimensional vectors, *position* is the raindrop coordinate in three-dimensional space, *velocity* is the velocity, acceleration is the *acceleration*, *lifetime* is the life span which marks a particle's time during its falling process. *Vanishing* is the velocity of a raindrop's decay, *size* is the size of a raindrop, *color[3]* is its color in RGB channels.

3.2. Raindrop Dynamic

In order to improve the efficiency of rain scene generation, we need to generate raindrops within certain depth of the field of view, update the three-dimensional positions of current raindrops and realize the dynamic effect of the rain scene. By updating the coordinate position, speed, life cycle and other properties of a single rain particle in particle system, we can realize the real-time updating of raindrops. As shown in the following formula (1):

$$\begin{cases} p = p' + \int_0^t v dt \\ v = v' + \int_0^t a dt \end{cases} \tag{1}$$

where p is the current position of a raindrop, p' is the initial position of the raindrop. v is the current speed. v' is the initial speed of the raindrop, a is the acceleration. t is the time period.

Due to the persistence of vision, the rain streak seems like a line with a certain length when falling down fast. During time period t, the displacement S of raindrops can be defined as the following formula (2):

$$S = V_0 \times t + a \times t^2 / 2 \tag{2}$$

where V_0 is the initial velocity of the raindrop, depending on its radius; t is the time of vision persistence; a is the acceleration of the raindrop. Considering the raindrop size, the length L of a rain streak can be defined by formula (3):

$$L = S + 2R = V_0 \times t + a \times t^2 / 2 + 2R \tag{3}$$

where R is the raindrop's radius.

In this paper, we mainly consider the impacts of wind and gravity. The acceleration is calculated by the following formula:

$$a = \frac{F_w + G}{m} \tag{4}$$

where a is acceleration, F_w is the wind power size, m is the quality of raindrop, G is the size of gravity.

Wind power has a direct influence on the horizontal movement of the raindrop. In this paper, Zioma [24] wind field model is adopted to simulate the wind in disorder, and the mathematical function expression is as follow:

$$w = \cos(\pi t) \times \cos(3\pi t) \times \cos(5\pi t) \times \cos(7\pi t) \times -0.1 \times \sin(25\pi t) \tag{5}$$

where t is the time, w is the wind-force, π is the pi, or ratio of the circumference of a circle to its diameter.

3.3. Simulation of Raindrops Merging

In natural rain scene, the diameters of raindrops are between 0.5 mm to 7 mm. On its way down, tiny raindrops gradually merge, when their diameters increase to 6 mm to 7 mm, big raindrops will break into tiny ones.

We simulate this process through the changes of raindrop diameters. The raindrops generated in upper air are set with smaller diameters which gradually increase on the way down so as to simulate raindrop expansion after raindrops merge. Meanwhile, the life value of raindrops is gradually decreasing to 0 when raindrops disappear so as to simulate raindrop disappearance after the fusion. The diameters of raindrops are set to decrease after increasing to the critical value, so as to simulate raindrop breaking.

3.4. Misty Simulation

In natural rain scene, fogs often rise especially in drizzling rain (hence fog is indispensable for realistic rain scene). Some research suggested that fogs appear because of the evaporation of rainwater [25]. Generally speaking, when raindrops fall down to the ground, their temperature rises. The temperature of near-surface condensation nuclei is low, therefore raindrops will evaporate and shrink due to larger pressure of raindrops surface. When the process reaches a critical state, condensation nuclei will condense and form fogs [26], [27].

Compared with raindrop particles, fog particles are smaller and therefore seem much mistier. A raindrop particle is normally surrounded by many fog particles. It would be quite time-consuming and costly if this misty effect was simulated by using numerous small fog particles. This paper proposes a more efficient method to solve this problem. We mix the raindrop texture with opaque background, and under the approach of weighted linear interpolation we got the center brightness of a raindrop and the blurring effect around it. So the atomization effect can be simulated.

Realistic rendering is based on texture mapping technique. The primary task is to capture the texture of rain. On its way down to the earth, the raindrop would not only change its trajectory but also change its shape due to the influences of different forces. When all the forces on a raindrop are balanced, its shape seems like a sphere; otherwise, the shape will appear flat in the bottom-front side and semi-sphere in the top-back side. We took a series of different raindrop images to make rain textures with different shapes. For instance, smaller raindrops look like little balls while larger ones look like mini-yurts which are hemispheric. For the middle states, we use interpolation technique to generate the mid-frame with the shape change from sphere to hemispheric.

After the raindrop textures were captured, each of them would be bound to the raindrop's structure according to its change of shape. Then we generated a mixed texture between background image and a texture of a raindrop, and adjusted its final misty effect by altering its transparency of an alpha-channel factor. Experimental results show that this approach can achieve realistic and efficient misty effects.

3.5. Lightening

Lightening often appears in thunderstorms. This paper simulates lightning on the basis of self-similarity between structures. The main part of the lightning is a skeleton structure. In this article, the L-System [28], which is commonly used in the plant growth simulation, is used to simulate the skeleton structure of lightning. The basic idea can be interpreted like this: the backbone of lightning gives out new branches, which sends out the secondary branches, and at last, the trunk and branches of different levels will form the skeleton structure of lightning. The law of lightning generation represents the Fibonacci sequence. Remember the sequence of n as F_n, the recursive relation is as follows:

$$F_{n+2} = F_{n+1} + F_n (n = 1, 2, 3...) \tag{6}$$

According to the basic ideas of L-System, this paper uses the line and bifurcation as the fundamental structure to establish the mathematical model of lightning: starting from the lightning backbone, it advances in decreasing step length, and gradually extends from the trunk to connected branches, and then this process will be repeated in a recursive manner without stopping till it comes to the last branch. In this model, the trunk and each branch has a cylindrical structure, and all the branches will be scaled, rotated and translated to the new position of their superior branch. In this way, the structure of lightning can be rendered after a finite repeat.

Fig. 1. The rain scene with lightning.

4. Structure and Rendering of Rain Scene

In this paper, we get an ideal test result by selecting to test the hazy sense and transparency of the rain scene in the night and observing a comparison of light and shade in the streetlights. We adopted night scene

to test the haziness and transparency of rain and observed the contrast between light and dark. The results are satisfying.

And we apply the ray tracing technique to rendering rain scene with misty effect. From the start point of light source, which is set as the viewpoint, a light ray traverses each pixel on the visual plane to determine whether it intersects other raindrops. Then the final image color will be calculated according to the Phong illumination model.

When intersecting a light ray, the raindrop depth can be calculated by the alpha-channel factor. In addition, the ray refraction will contribute to the final image color. Meanwhile, according to the Lambert Beer's law [29], we decompose a light ray into three parts when it runs through the raining range: one part of the light is absorbed by raindrops, one part is penetrating raindrops and the rest is reflected by the raindrop surfaces. Therefore, the farther the raindrops are apart from the light source, the darker their rendering colors will be.

5. Results

This paper adopts three-dimensional system of coordination in which X-axis points to the right, Y-axis points to above and Z-axis points to the internal direction according to the computer screen. Therefore, every property of a raindrop is initialized as follow: its initial position is located within the X-Z plane with random values along X-axis and Z-axis directions; initial velocity is of the value of 0 both in X and Z axes and a random float value in Y-axis; the velocity of a particle' demise is initialized as a user-defined parameter; the size of a raindrop is generated also as a random value which is in the interval [0.6mm, 1.4mm].

According to experimental data, the residence time t is in the interval between 0.05 seconds to 0.2 seconds[30]. Therefore we made it 0.1 seconds in our performance. The classic Marshall-Palmer model suggests that the density of raindrops takes up a trend of decrease accompanied with the increase of raindrop's radius and that the density of raindrops will reduce sharply when the radius exceed 1 mm. In our performance the radius R is set to be 1.5 mm in order to gain a realistic effect for the rain streak.

For a typical rain scene, our rain animation system runs on a 2.4 GHz Core (TM) with 4 GB RAM, and GeForce GT540M graphic cards. Table1 gives the data of dynamic FPS in the conditions of different number of particles in rain scene. If the number of particles is 2000, the frame per second (FPS) of dynamic rain simulation can reach a very high ratio up to about 478; if the number is 15000, the FPS is about 76 which is also in real-time.

Table 1. Statistics of Dynamic FPS with Different Number of Particles in Rain Scene

Num. of Particles	2000	5000	10000	15000
FPS	478	278	162	76

Fig. 1 shows the rain scene with lightning effect. Fig. 2 shows the rain scene without fog effect. Fig. 3 displays the rendering misty effect with a smaller depth of the fog, while Fig. 4 shows the rendering misty effect with a larger depth of the fog. Their comparison shows that the misty effect can strengthen the realistic result of tiny rain scene with fog. The larger the atomization depth is, the more realistic the hazy effect will be.

Fig. 5 shows the close-up effect of a single raindrop with ray tracing algorithm method. It is quite obvious to find the light refraction and shade effects of raindrops. Fig. 6 shows the effect of rain streaks which generated by controlling the raindrops' location from which they generate. Fig. 7 shows the rendering effect of a rain scene when 3000 particles are set. Fig. 8 shows the scene of heavy rain when 10000 particles are set. Comparison of Fig. 7 and Fig. 8 shows the different realistic rain effects can be achieved by setting different parameters.

Fig. 2 Rendering result without fog.

Fig. 3 Misty effect with a little fog.

Fig. 4 Misty effect with heavy fog.

Fig. 5 Result of single raindrop by ray tracing.

Fig. 6. Result of rain streaks.

Fig. 7. Result of light rain.

Fig. 8. Result of heavy rain.

6. Conclusions and Future Work

This paper proposed an image-based method of real-time rendering realistic rain streaks and rain scene animation, simulates raindrop particle by using the single voxel of particle system, and controlled the generation of different rain scene. The length of a single raindrop at the generation time, which was determined by its radium, was calculated according to physical equation. Based on Marshall-Palmer model, the distribution density of different raindrops was calculated and the depth of rain scene was improved, which was more aligned with the characteristics of natural rain scene. According to Newton's law, we simulated the state of raindrops motion, studied the atomization effect, introduced the wind field and lightning, and enhanced the reality of rain scene simulation. Ray tracing was used to draw the final rain particles to simulate the details such as raindrop intensity. Testing this method in ordinary PC will get an ideal real-time rain scene, which can fully satisfy the requirement of the game for reality and drawing speed.

In the future, we will simulate more complex rain scenes considering the forces of raindrops in different directions. Meanwhile, we will carry on a further research on realistic stimulation and speeding up the rendering of rain scene.

Acknowledgment

The authors wish to thank the anonymous reviewers for their suggestion to this paper. The work has been supported by the Fundamental Research Funds for the Central Universities (No.BLX2012049, No.YX2013-28), National Natural Science Foundation of China (No.61402038, No.61100132), the Undergraduate Research and Training Project in Beijing Forestry University (No.X1310022092).

References

[1] Rousseau P., Jolivet V., & Ghazanfarpour D. (2006). Realistic real-time rain rendering. *Computers & Graphics*, *30(4)*, 507-518.

[2] Wang, C., Wang, Z., Zhang, X. *et al*. (2008). Real-time modeling and rendering of raining scenes. *The Visual Computer*, *24(7-9)*, 605-616.

[3] Choudhury, B., Hao, P., & Chandran S. (2009). Real-time droplet modeling using color-space environment matting. *Proceedings of the SIGGRAPH'09: Posters* (p. 78). ACM.

[4] Niniane, W., & Wade, B. (2004). Rendering falling rain and snow. *ACM SIGGRAPH 2004 Sketches* (p. 14).

[5] Wang, L., Lin, Z., Fang, T. *et al.* (2006). Real-time rendering of realistic rain. *ACM SIGGRAPH 2006 Sketches* (p. 156).

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

[7] Li, D., Meng, H., & Shi, X. (1995). Membership clouds and membership cloud generators. *Journal of Computer Research and Development*, *32(6)*, 15-20.

[8] Qiu, X., & Liao, F. (2013). A fractal evolutionary particle swarm optimizer. *Journal of Computers*, *8(5)*, 1303-1308.

[9] Starik, S, & Werman, M. (2003). Simulation of rain in videos. *Proceedings of Texture Workshop* (pp. 406-409).

[10] Nakamae, E, Kaneda, K, Okamoto, T, *et al.* (1990). A lighting model aiming at drive simulators. *Proceedings of ACM SIGGRAPH Computer Graphics, 24* (pp. 395-404).

[11] Garg, K, & Nayar, S. K. (2006). Photorealistic rendering of rain streaks. *ACM Transactions on Graphics (TOG)*, *25*, 996-1002.

[12] Whitted, T. (1979). An improved illumination model for shaded display. *ACM SIGGRAPH Computer Graphics*, *13(2)*, 14.

[13] Cook, R. L., Porter, T., & Carpenter, L. (1984). Distributed ray tracing. *ACM SIGGRAPH Computer Graphics*, *18*, 137-145.

[14] Jiang, L., & Wu, E. (2013). Photorealistic rendering of rain streaks. *Journal of Image and Graphics, 18(5)*.

[15] Ward, G. J., Rubinstein, F. M., & Clear, R. D. (1988). A ray tracing solution for diffuse antireflection. *ACM SIGGRAPH Computer Graphics*, *22(4)*, 85-92.

[16] Lv, B. (2004). Research and improvement of the photon map algorithm. Master's thesis, Department of Computer Science and Technology, Nanjing University (pp. 3-20), Nanjing, China.

[17] Purcell, T. J. (2004). *Ray Tracing on a Stream Processor*. Stanford University.

[18] Kay, T. L., & Kajiya, J. T. (1986). Ray tracing complex scenes. *ACM SIGGRAPH Computer Graphics*, *20*, 269-278.

[19] Foley, T., & Sugerman, J. (2005). KD-tree acceleration structures for a GPU raytracer. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (pp. 15-22).

[20] Horn, D. R., Sugerman, J., Houston, M., *et al*. (2007). Interactive KD tree GPU raytracing. *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (pp. 167-174).

[21] Wang, W., Xiao, S., Meng, W., *et al.* (2008). Ray tracing algorithm based on octree space partition method. *Journal of Computer Applications*, *28(3)*, 656-658.

[22] Kim, T., & Lin, M C. (2007). Fast animation of lightning using an adaptive mesh. *IEEE Transactions on Visualization and Computer Graphics*, *13(2)*, 390-402.

[23] LaPointe, C., & Stiert, D. (2009). Volume lightning rendering and generation using L-systems. *Advanced Computer Graphics 2009*.

[24] Zioma, R. (2007). Gpu-generated procedural wind animations for trees. *GPU Gems*, *3*, 231-240.

[25] Donaldson, N. R., & Stewart, R. E. (1993). Fog induced by mixed-phase precipitation. *Atmospheric research*, *29(1)*, 9-25.

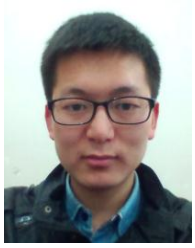
[26] Yan, W., Pu, M., Liu, A., *et al.* (2008). Boundary layer structure and origin of Nanjing winter fog process analysis. *Proceedings of the 15th National Conference on Cloud and Precipitation and Weather Modification*.

[27] Li, Z. (2001). Studies of fog in China over the past 40 years. *Acta Meteorologica Sinica*, *59(5)*, 616-624.

[28] Mishra, J., & Mishra, S. (2007). *L-System Fractals*. Elsevier.

[29] Bohren, C. F., & Huffman, D. R. (2008). *Absorption and Scattering of Light by Small Particles*. John Wiley & Sons.

[30] Garg, K., & Nayar, S. K. (2004). Detection and removal of rain from videos. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition: Vol. 1* (pp. I-528-I-535).



Cheng Wang is currently an undergraduate in the School of Information Science & Technology of the Beijing Forestry University, China. His current research interest includes computer graphics.

Meng Yang received the B.Sc. degree in computer science from Beijing Institute of Technology in 2006, and Ph.D. degree in State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China in 2012. He is a lecturer at the School of Information Science and Technology, Beijing Forestry University, China. His research interests include computer animation, realistic image synthesis and GPU techniques.

Xuemin Liu is currently an undergraduate in the School of Information Science & Technology of the Beijing Forestry University, China. Her current research interest is in photorealistic rendering.

Gang Yang received the B.S. and M.S. degrees in software engineering and computer application and technology from Shandong University, Jinan, China, in 1999 and 2002, respectively, and Ph.D. degree in computer application and technology, from Institute of Software Chinese Academy of Sciences, Beijing, China in 2006. He is an associate professor at the School of Information Science and Technology, Beijing Forestry University, China. His research interests include photorealistic rending, non-photorealistic rendering and texture synthesis.