A semi-automatic generation method for visually-plausible virtual clouds

Joo-Young Do
Kyungpook Nat'l Univ.
Daegu, Korea
djy@naver.com

Nakhoon Baek *†
Kyungpook Nat'l Univ.
Daegu, Korea
oceancru@gmail.com
(corresponding author)

Chang-Woo Lee
KOG Inc.
Daegu, Korea

Kwan-Woo Ryu
Kyungpook Nat'l Univ.
Daegu, Korea
kwryu@knu.ac.kr

Abstract

Cloud is one of the most important atmospheric effects in computer graphics applications, and there have been lots of research results on it. Although they have mainly focused on the modeling and rendering of more realistic clouds, nowadays flight simulation games or even full-scale flight simulators often need a quantity of visually plausible clouds rather than realistic ones. Furthermore, we also need to save graphics designer's efforts, especially for small production companies. Contrary to the previous works, we aimed at the mass production of clouds, and represent a new method for modeling various kinds of visually plausible clouds with as little effort as possible. These clouds are displayed in real time with low computing power consumption. Based on the hierarchical particles, our system starts from locating relatively large spherical particles in the space. Using these seed particles, our system automatically generates descendant particles to represent details of the clouds. In this way, the visually plausible clouds can be generated with much less effort, while the designer may build up specifically shaped clouds through controlling the seed particle locations. The particle hierarchy also enables us to naturally implement level-of-detail effects on the cloud rendering. In the final rendering stage, our system renders each particle with alpha-blended billboards to achieve fast real-time processing.

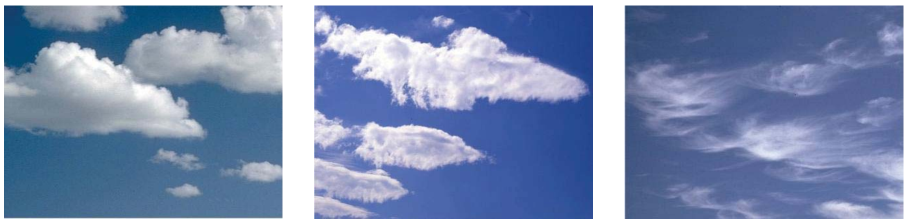
1: Introduction

Clouds are mandatory elements for natural outdoor scenes. Especially, they play important roles to increase user immersions in flight simulation games and/or more serious

* Corresponding author: oceancru@gmail.com
† This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (Grant 2011-0014886).

flight simulation applications. Thus, nowadays, we meet increased needs for effective ways of representing clouds in computer graphics applications[18].

The actual form of cloud depends on the size and density of seed particles, the strength of the uplift, the air stability, etc. Rapidly constructed clouds typically have water droplets of almost the same sizes, while slowly grown clouds have those of somewhat difference sizes. Cloud types mean the shape of the cloud and there are 10 major cloud types[14]. Among them, *cumulus*, *stratus* and *cirrus* are well-known and typical cloud types with distinguishable shapes, as shown in Figure 1.



(a) cumulus (b) stratus (c) cirrus

Figure 1. Three typical cloud types.

In computer graphics, clouds can be physically simulated from the water vapors to its final three-dimensional voluminous shapes, dynamically changing with respect to the atmospheric conditions such as temperature, air pressure, air stability, and so on. These physically simulated clouds may have ambiguous boundaries, and are hard to be effectively simulated. Additionally, we should consider the way of rendering those clouds, to show more realistic clouds. Since the incident lights can either pass through or be scattered by the water droplets, complex calculations are required to get precise colors of the clouds, with respect to their circumstances. Thus, although various modeling and rendering techniques for clouds have been developed, there is not yet a universal solution which can satisfy all the requirements.

We focused on the visually-plausible real-time display and its convenient generation. Our paper does not aim at a more realistic cloud modeling technique, as in most previous methods. Our goal is to represent a visually plausible method, with the following characteristics:

- **real-time display:** The rendering time for the finally constructed clouds should be reduced as much as possible. Typical flight simulators or even casual flight simulation games need much processing time for realistic motions of aircraft and others. Thus, the per-frame simulation costs for the clouds, as minor roles, should be minimized, not to disturb major calculations.
- **convenient generation:** The virtual cloud designers would want to see the final cloud appearance as soon as possible. Especially, novice designers tend to get the final output interactively, and we should satisfy their needs. When designers specify rough shapes of the clouds, our system would automatically generate their details.
- **user-controllable construction:** Designers may need specifically shaped clouds, and our system would support them. As skywriting[17] or sky-typing techniques[16] in the real world, designers can locate clouds whose shapes are specific alphabets, company logos, or game characters in the virtual sky of flight simulation games, for gaming purposes or advertising purposes.

Our cloud modeling method with the above characteristics has somewhat different goals in comparison with previous methods. Most previous methods aimed at the generation of realistic clouds, as natural as possible, while our goal is visually plausible clouds, with minimized design costs and rendering time.

In this paper, to achieve the above goals, target cloud types are roughly classified as cumulus, stratus and cirrus. At the first step, a user will locate relatively large spherical particles in the space, to intuitively describe the rough shape of a cloud. Then, our system automatically generates relatively small particles around those seed particles, in a hierarchical manner. Users can control the numbers, locations, and sizes of the descendant particles, to finally construct detailed ones from the original rough shapes.

At the rendering stage, particles are approximately rendered with DirectX billboards, to satisfy the fast real-time requirement. We use the concept of texture palettes as used in Wang's work[18], to plausibly display several cloud types in a single framework. Additionally, using hardware blending and renderable textures, the silver lining effects on the cloud boundaries are also supported. Conclusively, our method efficiently constructs visually plausible clouds, and displays them fast enough.

Previous works in cloud modeling and rendering are presented in Section 2. Section 3 represents our cloud modeling and rendering method. Experimental results from our prototype implementations are shown in Section 4. Conclusions and future work are followed in Section 5.

2: Previous Works

We can classify previous cloud modeling methods into two categories: *procedural methods* and *physically based methods*. Procedural methods focus on the appearance of clouds, and enable relatively fast modeling with somewhat reduced reality. Users may control cloud shapes with some intuitive parameters. In contrast, physically based methods are based on the physical simulation of the droplets in the cloud. For large-scale clouds, these methods are hard to achieve real-time processing capability, due to the massive calculations. Since the cloud shapes should be controlled indirectly through the physical parameters, they are also hard to make specifically shaped clouds.

Ebert[6] and later Schpok et al.[15] presented procedural cloud modeling methods, respectively, in which overall cloud shapes are constructed by implicit functions and their details are added by turbulence and noise factors. Since they used slice-based volume rendering techniques, all the textures should be updated for every viewpoint change, and thus they are hard to achieve real-time renderings.

Nishita et al.[13] focused on the rendering methods rather than modeling, to show more natural and realistic clouds, and represented a cloud shading method, which reflects light scatterings. Later, they also used GPU for cloud rendering[5] and extended their method for some special purposes[3, 2].

Harris et al.[7, 8] proposed a more advanced real-time cloud rendering method, which can handle multiple scattering effects due to multiple directional light sources and anisotropic scattering effects due to the change of viewpoints. When the viewpoint changes, they still need slice-based recalculations, though.

In physically based modeling area, there have been computational fluid dynamics solutions for modeling and animation of fluids including clouds. Kajiya and von Herzen[10] solved partial differential equations derived from fluid dynamics, for rendering clouds with ray tracing methods. Later, GPUs are actively used for parallel processing of massive data including cloud simulation. However, physically based simulation of clouds still requires heavy computation.

Dobashi et al.[4] used a cellular automaton to model cloud shapes and their motions. They simplified overall physical process of cloud construction to accelerate the cloud modeling and animation. Thus, it is difficult to generate various types of clouds. Additionally, it requires memory spaces for three-dimensional volume data, and also even much computation. At least at this time, fluid dynamics-based methods are hard to achieve real-time processing, due to their heavy computation.

Wang suggests a good approximation method for cloud modeling and rendering, especially for flight simulation games[18]. A cloud is modeled with a set of rectangular boxes and later rendered with alpha-blended textures. Wang gave considerations on the expression power for the designers. After designing the final virtual clouds in a rendering program such as 3D Studio Max, a plug-in program exports the final clouds into the flight simulation game. In contrast, our system generates all the details from the roughly located seed particles, through adding descendant particles with their own hierarchies. Thus, although designers are hard to control the fine details, this automatic generation capability in our system enables us to achieve mass production of visually plausible clouds. Though it is a common point to use a kind of texture palette at the rendering stage, our system dynamically changes the texture set according to the cloud types, for more efficient cloud generation on a single framework. More details of our cloud modeling and rendering method will be represented in the following sections.

3: Cloud Modeling and Rendering

In this section, we start from our cloud modeling method, and represent its corresponding real-time rendering method. As already mentioned, our final goal is fast modeling of visually plausible clouds, rather than realistic clouds.

3.1: Modeling of clouds

The main idea in our cloud modeling method is that a cloud is modeled with a set of spherical particles, with their own hierarchies. At the first stage, users locate relatively large particles, which become *seed particles* or the upper-most level particles. According to the desired cloud types, we apply some constraints on the relative positions of these seed particles. For example, since most cirrus clouds are characterized by thin and wispy shapes spreading horizontally, the locations of seed particles for cirrus clouds are vertically bounded to a specific range. Other cloud types also have their corresponding constraints, and users should satisfy these constraints to make clouds of a specific type. If necessary, our system can also generate these initial seed particle locations automatically. In this case, the clouds can be full-automatically generated.

After locating the seed particles, the next level particles with smaller sizes are positioned around their upper level ones. This hierarchical generation is used to reflect the randomness on the fine details of cloud boundaries. Up to the pre-specified number of levels, the child particles are located around the parent particles, the grandchild particles are located around the child particles, and so on, as shown in Figure 2.

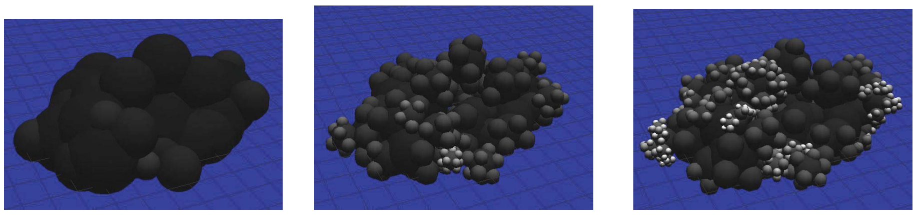

(a) the seed particles (level 0) (b) child particles (level 1) (c) grandchild particles (level 2)

Figure 2. Hierarchical particles for a cloud.

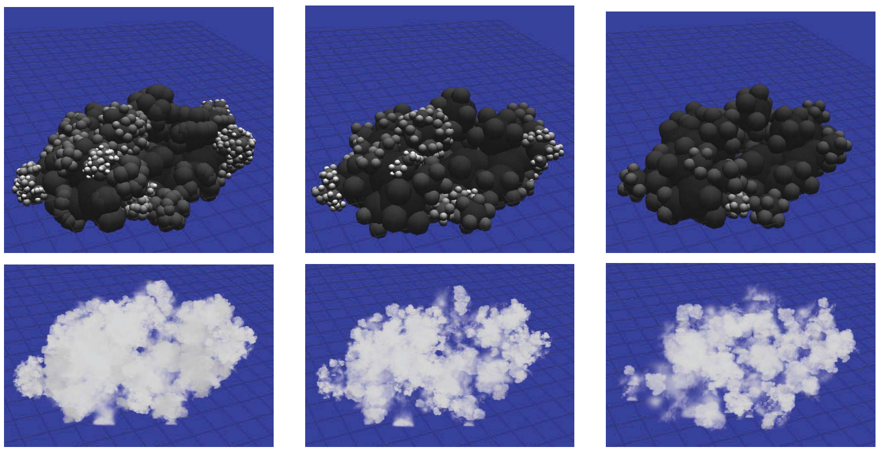

(a) cumulus (b) stratus (c) cirrus

Figure 3. Various cloud shapes.

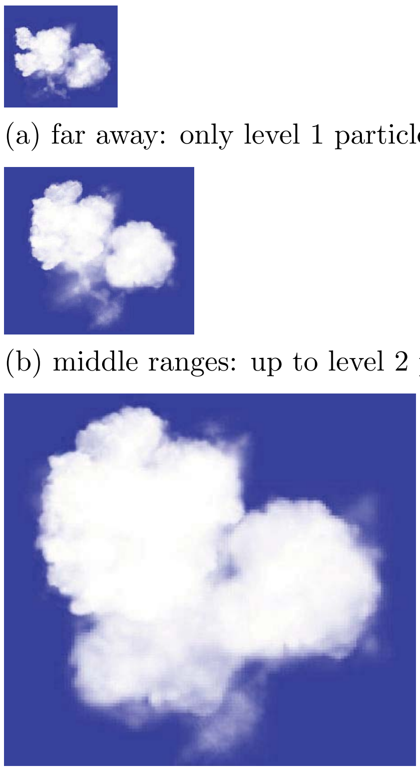
To generate child particles, we use a modified version of Levet's particle sampling method[11], which is originally developed to approximate an implicit surface with a set of particles. Originally, Levet's method generates a dense set of particles with the same sizes. In contrast, we need to control the denseness according to the cloud types, and also, particle sizes had better to have somewhat distributed range, for more plausible cloud boundaries.

Each particle has a set of its own attributes including the radius, the center position, and the repulsion radius. The repulsion radii are varied with respect to the cloud types. In the case of cumulus clouds, their shapes are somewhat rounded and need to show some smooth feeling, and thus, we decrease the repulsion radius to densely position relatively many particles. Stratus clouds seem to be blown off and their relatively large repulsion radii give us sparsely located particles for final thin and wispy shapes.

For a given particle, the center positions of the next level particles are located on the surface of their parent particles, while the child particles are apart more than the repulsion radius from one another. More details on the positioning child particles are described in Levet's paper[11].

The radii of child particles are reduced in proportion to that of its parent particle. It is more natural to vary their radii in a specific range. Thus, we set the radius r_{i+1} at $(i+1)$-th level in a statistical way, as follows:

$$r_{i+1} = (s + d \cdot R) \cdot r_i, \tag{1}$$



(a) far away: only level 1 particles.

(b) middle ranges: up to level 2 particles.

(c) near distance: up to level 3 particles.

Figure 4. Clouds rendered to different levels for the level-of-detail technique.

where r_i is the radius of its parent particle at i-th level, and s and d are user-controllable parameters for the role of average and standard deviation, respectively. The R term is a randomized real number in the range of $[-1, +1]$. Since the repulsion radii for child particles are arranged to be proportional to its radius, the overall child particles are irregularly located. If necessary, we can modify the above equation to show a normalized distribution of the child radii.

Figure 3 represents a set of hierarchical particles, generated by the above procedures. To generate various cloud shapes, users can control the number of particle levels, the ratio between the particle radius and its repulsion radius, the s and d values in Equation (1), and so on. Of course, we can minimize the manual selections through using the system default values for each specific cloud type.

3.2: Rendering of clouds

Since we use hierarchical particles for cloud modeling, we also perform particle-based rendering. Depending on the cloud types such as cumulus, stratus and cirrus, we prepare separated texture sets and each particle is rendered with a randomly chosen texture from the specific texture set. Although it looks similar to that of Wang's work[18], our rendering strategy would be explicitly different on the use of different textures for different cloud types. Furthermore, our texture sets can be dynamically changed in real-time when a user selects different cloud types.

Another benefit of our hierarchical particle-based method is straight-forward implemen-

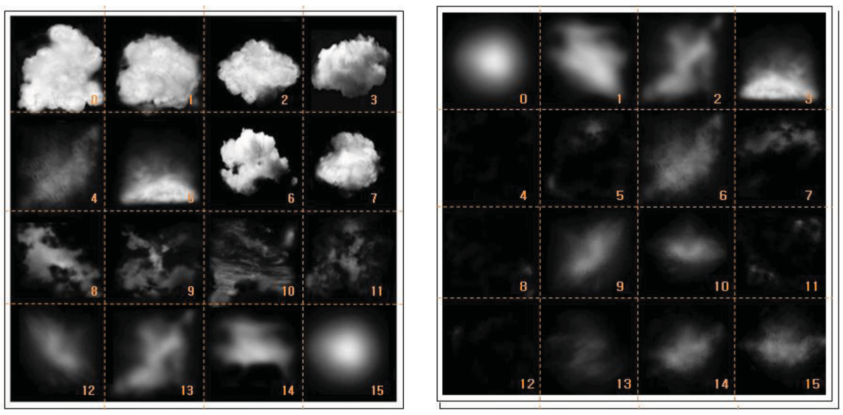


Figure 5. Textures used for cumulus(left) and cirrus(right) clouds.

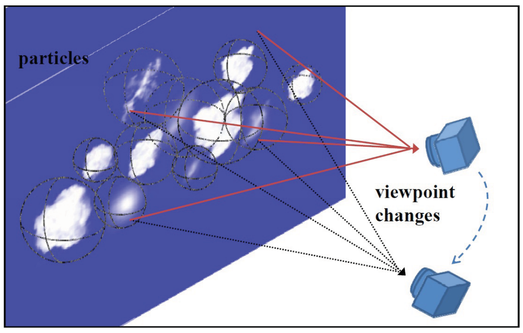
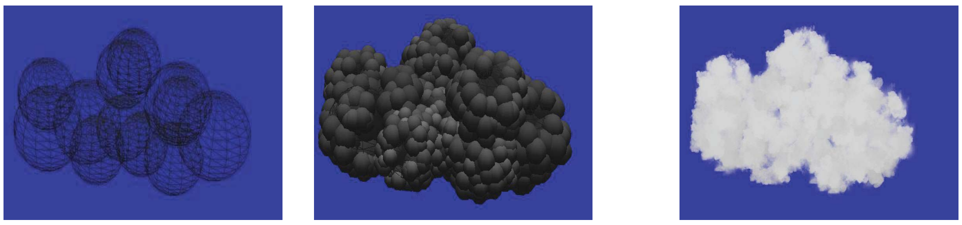


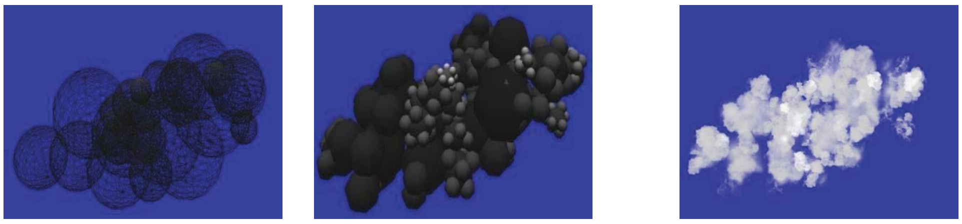
Figure 6. Billboard representations of cloud textures.

tation of the level-of-detail technique. For clouds at long distances, we need to render only seed particles. As clouds approaching to the camera, we adjust the particle levels to be rendered. Finally, the closest clouds will be drawn to its maximum level particles. Figure 4 shows examples of clouds for the level-of-detail demonstration.

Texture images used in our system are 32bit full color images with alpha values, as shown in Figure 5. We use different sets for different cloud types. Each set of textures consists of 16 texture images with 256 $\times$ 256 resolutions, to use totally 4M byte video RAM for each set of textures.

At the texture mapping stage, particles are displayed with *DirectX billboards*, as shown in Figure 6. Since we use spherical particles, the billboards are always rendered to be perpendicular with respect to the camera positions. This is another advantage of spherical particles, since other shapes, as an example, such as rectangular boxes used in Wang's work[18] require re-positioning calculations for camera position changes. Even for rapid viewpoint changes, our spherical particle-based method is easy to maintain the overall cloud shapes. Additionally, we used hardware blending and renderable texture features in DirectX, to more naturally express the cloud boundaries and to approximate the multiple scattering and anisotropic scattering effects, as did in Harris's work[9].


(a) a cumulus cloud: 12 seed particles, 946 descendant particles and the image.


(b) a stratus cloud: 12 seed particles, 193 descendant particles and the image.

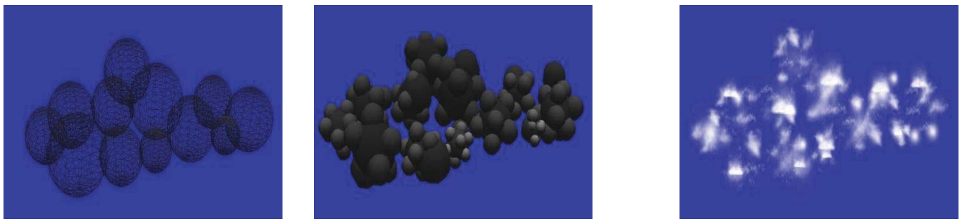

(c) a cirrus cloud: 12 seed particles, 128 descendant particles and the image.

Figure 7. Clouds generated by our prototype system.

4: Experimental Results

Using techniques described in the previous sections, we developed a prototype system for editing, modeling, and rendering of clouds. In this section, we present the experimental results, executed on a Microsoft Windows system with a single core CPU at 3.2GHz, 1G byte main memory and nVIDIA GeForce 6600 graphics card.

Figure 7 shows the cumulus, stratus and cirrus clouds generated on our system, with different seed particle locations and other parameters for each cloud type. Cumulus clouds are modeled with relatively thick seed particle layers and more distinct and voluminous textures. In the case of stratus clouds, we use cloud textures with somewhat indistinct boundaries and let the particles a little scattered. Most thin particle layers and more scattered particles are used for cirrus clouds. Additionally, more transparent and scattered cloud textures are used.

For clouds shown in Figure 7, we generated up to level 2 particles, starting from 12 to 23 seed particles. Control parameters such as particle radii, repulsion radii, and so on are set to appropriate values for each cloud type. Total numbers of particles are ranged from 128 to 946, depending on the cloud types.

Figure 8 shows some examples of composing our final alpha-blended cloud images with landscape images. We used at most 1,961 and 2,253 particles for the upper and lower cumulus clouds, respectively. Stratus and cirrus clouds need fewer particles.

Due to the particle-based modeling, the rendering speed of our system depends mainly

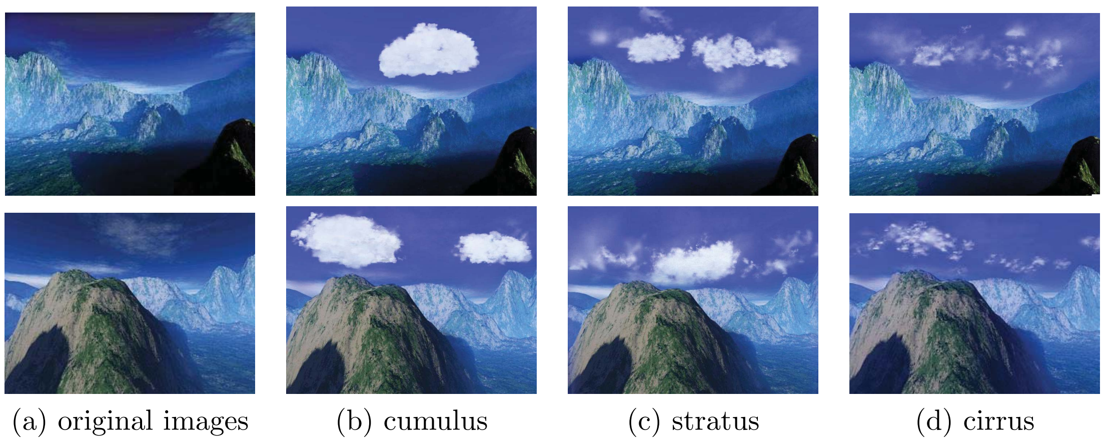


Figure 8. Composition of our clouds with landscape images.

Table 1. Experimental results with respect to the number of particles.

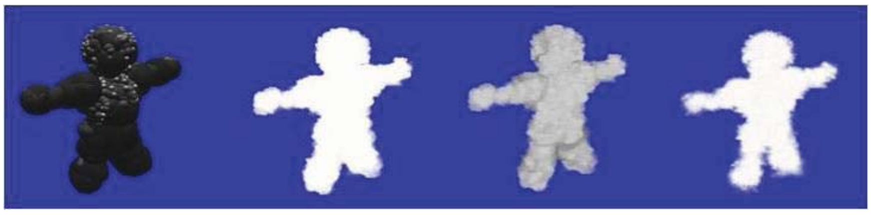
No. of particles	frames per second
100	340.78
1,000	200.30
2,000	130.00
3,000	80.28
4,000	47.59
5,000	25.18
6,000	17.03

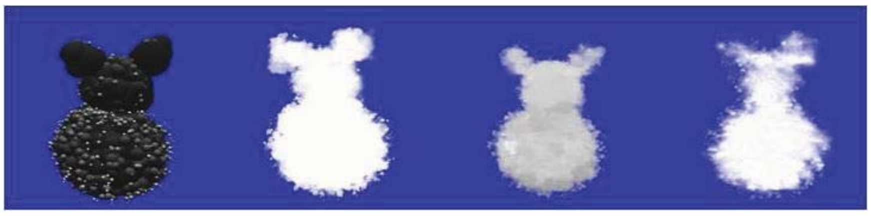
on the number of particles and their radius on the screen. Our benchmark shows more than 140 frames per second for about 3,000 particles with various radii, as shown in Table 1. We used $1,280 \times 1,024$ resolutions for the final images. We turned off the level-of-detail features in the above measurement, and thus, we can additionally accomplish much speed up with the proper level-of-detail processing. Most plausible images can be generated with less than 3,000 particles. If necessary, users can adjust the trade-off between the rendering speed and the final rendering quality, since our system provides user-controllability for the number of particles, their minimum and maximum sizes, ratios between them, and so on.

Figure 9 shows our system's design capability for intentional cloud shapes. As the skywriting[17] or the sky-typing messages[16] are used in the real world, flight simulation applications are also targets for sky-written messages. For this purpose, our users can construct any specific cloud shapes as shown in Figure 9. We also integrated our cloud rendering feature into a flight simulation game, as shown in Figure 10. Clouds in these screen shots are modeled with up to 2,000 particles for each scene.

5: Conclusions and Future Work

In this paper, we presented a cloud modeling and rendering technique to generate visually plausible clouds for typical cloud types including cumulus, stratus and cirrus, even without delicate manual works by expert designers. Users specify the rough cloud shapes through


(a) bubble man: totally 1,998 particles


(a) animation character: totally 1,325 particles

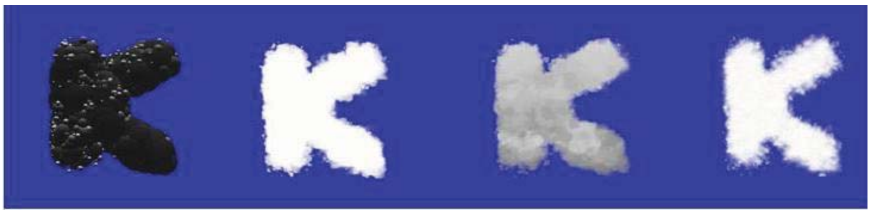

(a) initial K: totally 1,321 particles

Figure 9. Cloud modeling for intentional shapes.

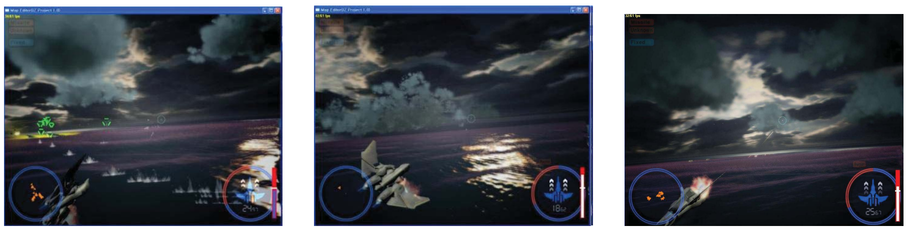


Figure 10. Clouds integrated in a flight simulation game.

locating spherical particles in the space. Then, the details of clouds are automatically generated as hierarchically descendant particles. Using a particle-based modeling technique, we can avoid re-calculations due to viewpoint changes and additionally approximated scattering effects for faster processing. Our proposed method would be suitable for real-time flight simulation applications including casual flight simulation games and mobile platform applications.

On our cloud generation strategy for specifically shaped clouds, we can easily add automatic seed particle generation features from generic mesh models or solid geometries. In this way, users can get a particle-based cloud modeling method, which requires much little user interaction. Since Bradshaw and O'Sullivan[1] and Liu et al.[12] presented algorithms to approximate solid objects with spheres or ellipsoids, we expect we can add these features soon.

We also need to consider dynamic cloud animations. Since our system is based on the hierarchical particles, it is easily possible to make natural and efficient interactions with winds. After setting proper physical quantities such as densities to the particles, a physically-based simulation with the particles and wind factors would generate cloud dispersion and disappearance very naturally. Adding path specification features and wind direction simulation features, we would finally get an integrated cloud processing system.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(Grant 2011-0014886).

References

[1] Gareth Bradshaw and Carol O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph.*, 23(1):1–26, 2004.

[2] Y. Dobashi, Y. Shinzo, and T. Yamamoto. Modeling of clouds from a single photograph. *Computer Graphics Forum*, 29(7):2083–2090, 2010.

[3] Y. Dobashi, T. Yamamoto, and T. Nishita. Interactive and realistic visualization system for earth-scale clodus. In *Proc. Pacific Graphics (poster paper)*, pages 35–38, 2009.

[4] Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tsuyoshi Okita, and Tomoyuki Nishita. A simple, efficient method for realistic animation of clouds. In *SIGGRAPH '00*, pages 19–28, 2000.

[5] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Interactive rendering of atmospheric scattering effects using graphics hardware. In *HWWS '02: Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conf. on Graphics hardware*, pages 99–107, 2002.

[6] David S. Ebert. A cloud is born. In *SIGGRAPH '97*, page 245, 1997.

[7] Mark J. Harris. Real-time cloud rendering. *Computer Graphics Forum*, 20(3):76–84, 2001.

[8] Mark J. Harris, William V. Baxter, Thorsten Scheuermann, and Anselmo Lastra. Simulation of cloud dynamics on graphics hardware. In *HWWS '03: Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conf. on Graphics Hardware*, pages 92–101, 2003.

[9] Mark Jason Harris. Real-time cloud simulation and rendering. Technical Report TR03-040, University of North Carolina, 2003.

[10] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *SIGGRAPH '84*, pages 165–174, 1984.

[11] Florian Levet, Xavier Granier, and Christophe Schlick. Fast sampling of implicit surfaces by particle systems. In *SMI '06: Proc. of the IEEE International Conf. on Shape Modeling and Applications 2006*, page 39, 2006.

[12] Shengjun Liu, Charlie C. L. Wang, Kin-Chuen Hui, Xiaogang Jin, and Hanli Zhao. Ellipsoid-tree construction for solid objects. In *SPM '07: Proc. of the 2007 ACM Symp. on Solid and Physical Modeling*, pages 303–308, 2007.

[13] Tomoyuki Nishita, Yoshinori Dobashi, and Eihachiro Nakamae. Display of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH '96*, pages 379–386, 1996.

[14] R. Rogers. *A Short Course in Cloud Physics*. Pergamon Press, 1988.

[15] Joshua Schpok, Joseph Simons, David S. Ebert, and Charles Hansen. A real-time cloud modeling, rendering, and animation system. In *SCA '03: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, pages 160–166, 2003.

[16] Skytyping. http://www.skytyping.com/.

[17] Skywriting. http://en.wikipedia.org/wiki/Skywriting.

[18] Niniane Wang. Realistic and fast cloud rendering. *Journal of Graphics Tools*, 9(3):21–40, 2004.