Asian Journal of Information Technology 5(5) : 555-565, 2006

Particles Cloud Modeling Algorithm for Virtual Environment

Muhammad Azam Rana, Mohd Shahrizal Sunar and Siti Mariyam Shamsuddin
Department of Computer Graphics and Multimedia,
Faculty of Computer Science and Information System,
University Technology Malaysia, 81310 Skudai, Johor Darul Takzim, Malaysia

Abstract: Clouds are an integral part in earth's weather system, a strong indicator of weather patterns and changes and are an important component of the visual simulation of any outdoor scene. The complexity of cloud formation, dynamics and light interaction makes real time cloud modeling difficult for virtual reality applications. A fundamental problem of any modeling system is the creation of shape of clouds as they consist of small particles and therefore it is difficult to define their shape. This study proposes an efficient algorithm based on randomized method to create detailed volumetric shapes of clouds using particles system. This research consists of two phases. In the first phase, cloud shape modeling algorithm based on randomized method has been developed and tested. This algorithm creates particles data and places these particles randomly in cloud volume space so that they look like a cloud. Second phase of this research presents an intuitive and interactive editing environment-namely cloud macrostructure editor for designing cloud shapes. This bi-level technique provides integrated environment for modeling volumetric clouds with particle system. This randomized based algorithm has been tested and is able to create good shapes of clouds. It is simple to use, efficient and gives on-the-fly control over size of particles and number of particles to be used.

Key words: Cloud, rendering, virtual reality, flight simulator, billboard

INTRODUCTION

Clouds are very important in generation of images for outdoor scenes, interactive flight simulator applications, games and so on. The color and shape of clouds is dependent on the position of sun and position of the viewer as well[1]. This makes it important to display clouds in three-dimensional space so that changes in their shape and color with varying positions of sun and viewer may be observed.

Therefore, a lot of methods have been methods developed to create photo-realistic images. These methods can be classified into two groups. Methods in first group[2-34] use procedural modeling techniques. Methods in the second group model phenomena by simulating physical process[1,35-41]. Modeling phenomena such as clouds, smoke, water and fire have proved difficult with the existing techniques of computer image synthesis. These fuzzy objects do not have smooth, well defined and shiny surfaces; instead their surfaces are irregular, complex and ill defined[29].

Clouds, like other amorphous phenomena, elude traditional modeling techniques with their peculiar patterns of intricate, ever-changing volume-filling microstructures[2]. This challenge has lead us create an interactive system that allows artists to easily design and interactively render visually convincing clouds. This will accelerate rendering to real-time and extend applications from static data visualization and movies to interactive exploration and video games. In addition, the use of responsive, interactive system aids comprehension of synthetic environments and increases the productivity of artists.

We have developed an interactive, multi-level, cloud modeling and rendering system using intuitive controls. Our system is composed of a high level modeling system, the renderer and the user interface. The design of an intuitive, multi-level interface for the system makes it easy to use for both novice and expert users. The designer interactively outlines the general shape of the clouds using cubes. The cloud details are described by selecting a 2D texture for cubes from a pool of pre-computed 2D textures. The clouds are then rendered using textured billboarding technique. This detailed modeling and rendering is performed using the graphics processor through the use of textures and texture transformations.

We begin with a brief survey of current cloud modeling techniques and then introduce our framework

Corresponding Author: Muhammad Azam Rana, Department of Computer Graphics and Multimedia, Faculty of Computer Science and Information System, University Technology Malaysia, 81310 Skudai, Johor Darul Takzim, Malaysia

for cloud formation and rendering. Next, we present our test results. Finally, we present conclusions and our planned future work.

PREVIOUS WORK

Approaches to cloud modeling may be classified as simulation-based or procedural[13]. Simulation methods produce realistic images by approximating the physical processes within a cloud. Computational fluid simulations produce some of the most realistic images and movement of gaseous phenomena. However, despite the recent breakthroughs in real-time fluid simulation[36], large-scale high quality simulation still exhausts commodity computational resources. In contrast, procedural methods rely on mathematical primitives, such as volumetric implicit functions[42], fractals[31,42], Fourier synthesis[27] and noise[32] to create the basic structure of the clouds. This approach can easily approximate phenomenological modeling techniques by generalizing high-level formation and evolution characteristics. These procedural modeling systems often produce more controllable results than true simulation in much less time. Many approaches use volumetric ellipsoids to roughly shape the clouds and then add detail procedurally[12,34,42].

Correct atmospheric illumination and shading is another difficult problem that must be addressed to produce convincing cloud images and animations. Early work discussed accurately modeling light diffusion through clouds, accounting for reflection, absorption and scattering[43]. Further research has explored accurate volumetric light scattering and inter reflection for clouds[7,38,42,44,45]. While full self-shadowing, translucent volume rendering is approaching interactive rates[34], we elect to implement a visually convincing approximate volumetric lighting model utilizing graphics acceleration to achieve true interactive performance.

Two areas of previous work, cloud modeling and cloud rendering are important to our work. Cloud modeling is concerned with the generation of cloud data and how to use and organize this data to represent clouds in computer. Reeves introduced particle system as an approach to model clouds and other fuzzy phenomena[29]. Voxels are another choice for clouds, which provide a uniform sampling of the volume and can be rendered with both forward and backward methods. Procedural noise techniques are also important to cloud modeling as a way to generate random and continuous density data to fill cloud volumes[17,32,46]. We model our clouds by making use of texture-mapping techniques. We make use of Graphics Hardware accelerated API-OpenGL to implement this technique at a very fast rate.

Cloud rendering is much difficult, as realistic shading requires the integration of the effects for optical properties through the cloud volume and incorporation of complex light scattering with the cloud volume. Previous work has attempted to approximate the physical characteristics of clouds at various levels of accuracy and complexity and then to use these approximate models to render images of clouds. Blinn[43] introduced the use of density models for image synthesis in, where he presented a low albedo, single scattering approximation for illumination in a uniform medium. Kajiya and Von Herzen[38] extended this research with methods to ray trace volume data exhibiting both single and multiple scattering. Max[44] provided an excellent survey in which he summarized the spectrum of optical models used in volume rendering and derived their integral equations from physical models. David Ebert has done much work in modeling solid spaces, including offline computation of realistic images of smoke, steam and clouds[1,21]. Nishita *et al.* introduced approximations and rendering techniques for global illumination of clouds accounting for multiple anisotropic scattering and skylight[20].

Most of the physical methods need a large amount of computation time. The simulation of clouds covering a large area in the sky is impossible by these methods on today's desktop PCs. Controlling cloud shapes is also difficult by using the methods in this category. Procedural methods such as presented by Dobashi *et al.*[11] and Harris[3] produce good cloud images taking into account light scattering and shadows, still these techniques require a lot of processing power. For the applications that are not mission critical and where the effects of light can be ignored, our approach can produce very fast images requiring less computing power. The viewer even can go very close to clouds and this produces no visual artifacts, the only limitation is that viewer cannot pass through the clouds as clouds are modeled as 2D textures.

SHAPE MODELING ALGORITHM

This section presents the general model formulation of cloud shape modeling algorithm based on randomized method. Part of the reason that particle clouds look any good is that it incorporates randomness in a controlled way. Each particle is assumed to have a roughly spherical volume in which a Gaussian distribution governs the density falloff from the center of the particle. Each particle has attributes of a center, radius, density and color. For a good approximation, particles of varying sizes and densities are used build a cloud space.

Input data: Data required for the cloud shape modeling algorithm consists of the attributes of the particles making

clouds and coordinates of 3-D space in virtual environment that contains particle clouds. The data inputs that are used are described below:

Cloud center: Cloud center consists of coordinates in 3D space. It consists of X-Coordinate, Y-Coordinate, Z-Coordinate values representing the center of cloud bounding volume.

Number of particles: It represents the total number of particles that are randomly distributed in the cloud bounding volume to make shape of a cloud. Each particle is consists of a number of attributes such as center, radius, density and color.

Radius: It represents the maximum value for radius of cloud particles. In order to get good approximation of cloud, particles of varying sizes are used to make cloud. The radius of each particle is randomly calculated that is les than or equal to this value.

Data file format: In the following, the organization of the input data files is discussed. This data file serves as input data files for the proposed cloud shape modeling algorithm. Table 1 shows the format of a data file. On the first line, it has total number of particles in the system. On the second line, it has total number of cubes in the system. Rest of each line contains the information about each cube such as particles in a cube, cube dimension, cube center and radius of particles in that cube.

The description of each and every piece of data and its type is as described below:

Total particles: It represents the total number of particles in the system. The *integer* data type is used for this data.

Total cubes: It represents the total number of cubes used in the system to build the apparent shape of the clouds. The *integer* data type is used for this data.

Particles: It represents the total number of particles to be distributed in a particular cube. The *integer* data type is used for this data.

Length: It represents length of a particular cube. The *float* data type is used for this data.

Height: It represents height of a particular cube. The *float* data type is used for this data.

Width: It represents width of a particular cube. The *float* data type is used for this data.

Table 1: Format of input data file

Line	Input Data
1	<Total Particles>
2	<Total cubes>
3	<Particles, Cube(height, width, length), Center(x,y,z), Radius>

Model formulation: The model formulation for each step of the cloud shape modeling process is presented using the following notation[48].

N = Total number of particles.
X_L = Length of cloud bounding volume along x-axis.
Y_L = Length of cloud bounding volume along y-axis.
Z_L = Length of cloud bounding volume along z-axis.
R_{max} = Maximum value for radius of particles.
X_C = X-coordinate for center of cloud space.
Y_C = Y-coordinate for center of cloud space.
Z_C = Z-coordinate for center of cloud space.
R(i) = Radius of particle i.

The model formulation is as follows.

Step 1: For each particle, evaluate the location of particles in cloud bounding volume. Let X(i), Y(i) and Z(i) represent the X-coordinate, Y-coordinate and Z-coordinate respectively for the location of particle i. Then these values are calculated as follows:

$$X(i) = random(s)$$
$$Y(i) = random(s)$$
$$Z(i) = random(s) \quad (1)$$

where,

$$i = 0,1,2, \ldots, N-1,$$

s is seed for random-number generator and sets starting point for generating a series of pseudorandom numbers and
random(s) is a function takes seed as an argument and generates a random number.

The constraints on the values are

$$X(i) \leq X_L, \quad \text{for all i}$$
$$Y(i) \leq Y_L, \quad \text{for all i}$$
$$Z(i) \leq Z_L, \quad \text{for all i} \quad (2)$$

In order to transform X(i), Y(i) and Z(i), for all values of i, to meet above constraints, we compute as;

$$X(i) = X(i)\% X_L$$
$$Y(i) = Y(i)\% Y_L$$
$$Z(i) = Z(i)\% Z_L \quad (3)$$

where% is the modulus operator.

To add randomness, we add phase shifts in X(i), Y(i) and Z(i). Let P_x, P_y and P_z be the phase shifts, then their values are calculated according to the following relationships[27]:

$$P_x = \pi/2 \text{ Sin}(0.5\ Y(i))$$
$$P_y = \pi/2 \text{ Sin}(0.5\ X(i))$$
$$P_z = \pi/2 \text{ Sin}(0.5\ Z(i)) \quad (4)$$

The phase shifts produce a controlled pseudo-random effect by shifting the X component as a function of Y and Y component as a function of X component. To provide three dimensional variations, the phase shifts are augmented by added sine variations with Z component[27]:

$$P_x = P_x + \pi \text{ Sin}(P_x * Z(i)/2)$$
$$P_y = P_y + \pi \text{ Sin}(P_x * Z(i)/2) \quad (5)$$

Then the phase shift values calculated above added to X(i), Y(i) and Z(i) to get their new values as listed below:

$$X(i) = X(i) + P_x$$
$$Y(i) = Y(i) + P_y$$
$$Z(i) = Z(i) + P_z \quad (6)$$

Finally, the values of coordinates for the location of particle that fits within the cloud bounding volume in virtual environment with respect to cloud center are calculated by adding the value of cloud center (X_c, Y_c, Z_c), as shown by the equations listed below:

$$X(i) = X_c + X(i)/2$$
$$Y(i) = Y_c + Z(i)/2$$
$$Z(i) = Z_c + Z(i)/2 \quad (7)$$

Step 2: Calculate the radius of each particle. Each particle has a different value of radius which is less than or equal to a preset value R_m. Let R(i) represents radius of particle i, then the value for radius of the particle is calculated as follows:

$$R(i) = \text{random(seed)} \quad (8)$$

where i = 0,1,2, …, N-1.
and random(seed) is a function takes seed as an argument and generates a random number.

The constraints on the values are

$$R(i) \leq R, \quad \text{for all } i \quad (9)$$

```
get no_of_clouds
for each cloud
    get cloud_center
    get no_of_particles
    get cloud_bounding_volume
    for n=1 to no_of_clouds
        get center(x,y,z) of a particle
        if (center lies in cloud_bounding_volume)
            add particle to cloud space
        else    get a new particle
        end if
    next
next
```

Fig. 1: Pseudo code for shape modeling algorithm

In order to transform R(i), for all values of i, to meet above constraint, we compute as;

$$R(i) = R(i)\%\ R \quad (10)$$

where% is the modulus operator.

In order to get randomness in the values of radius, some noise is added to it as listed below:

$$R(i) = R(i) + \pi/2 \text{ Sin } (R(i)) \quad (11)$$

Step 3: Add the particle to cloud and repeat the process of generating data for new particles according to step 1 and step 2 till total number of particles (N) is reached.

Pseudo code of the algorithm: The Fig. 1 presents the pseudo code of the proposed algorithm for cloud shape modeling process.

IMPLEMENTATION OF THE ALGORITHM IN CLOUD EDITOR

The evolving structure of clouds is modeled using a two-level approach. For controlling the general shape of clouds, cubes have been used. Various cubes can be sized and arranged in order to create a desired shape of clouds. For low-level cloud details particles system has been used to ill he cloud volume with particles.

In order to evaluate the shapes modeled by the proposed algorithm, we make use of an interactive editing application-cloud macrostructure editor and use it to design the apparent shape of clouds. The snapshot of cloud macrostructure editor is shown in Fig. 2 and the further details about this editor can found in[49].

To design cloud shapes, we can add cubes by pressing Add Cube button from the control group named

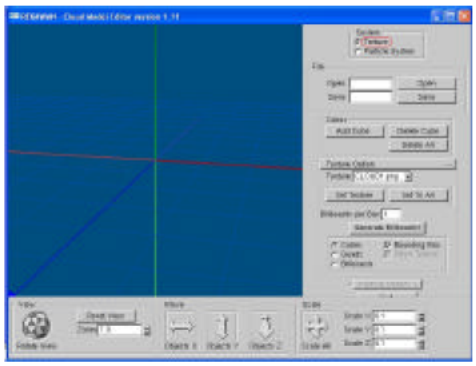

Fig. 2: Snapshot of Cloud Macrostructure Editor

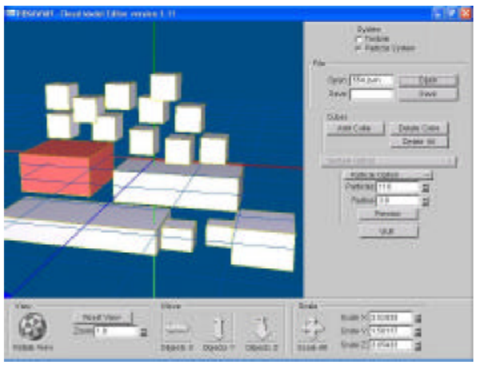

Fig. 3: Using 17 cubes for clouds apparent shape

Table 2: Data file generated by snapshot of Fig. 3

Line	Input Data
1	554
2	17
3	200, 7.39642, 1.07048, 2.89284,-3.4,-3.3,-0.600002, 2.1
4	10, 1.12383, 0.893671, 0.750589, 2.8,-3.1,-0.4, 2.2
5	5, 1.29472, 0.981624, 1, 1.1,-3.2, 0, 2
6	100, 4.99599, 1.30149, 2.33467, 1.1,-0.899999, 0, 3
7	110, 3.02839, 1.58117, 3.05433,-3.8,-0.499999, 0, 3
8	10, 1, 1, 1, 2.5, 0.600001, 0, 2
9	8, 1, 1, 1,-0.399999, 0.800001, 0, 3
10	7, 1, 1, 1, 1.1, 0.500001, 0, 2.2
11	5, 1, 1, 1, 1.5, 1.8, 0, 2.4
12	8, 1, 1, 1,-2, 1.3, 0, 2.123
13	9, 1, 1, 1, 6.12438e-007, 2.1, 0, 2
14	50, 2.36429, 1.50285, 2.29693 4.6,-3, 0, 2.2345
15	8, 1, 1, 1,-4, 2.5, 0, 2
16	0, 1, 1, 1, 0.500001, 3, 0, 2.5
17	7, 1, 1, 1,-4, 1.3, 0, 2
18	7, 1, 1, 1,-2.8, 2, 0, 2
19	10, 1, 1, 1,-1.4, 2.6, 0, 2

as Cubes. The attached parameters with each cube are its length, width, breadth, center and number of particles and radius of particles. We can use any number of cubes to design cloud apparent shape.

After finishing the design of the cloud apparent shape, the data related with these cubes is used by the proposed algorithm and it models the microstructure of

Fig. 4: Cloud modeled with 554 particles

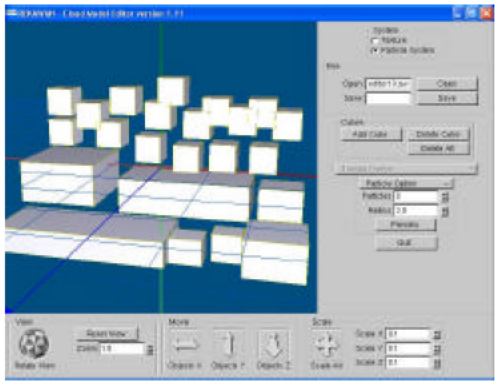

Fig. 5: Using 23 cubes for clouds apparent shape

Fig. 6: Cloud modeled with 615 particles

the cloud by filling the cloud space with particles of varying sizes at random position.

In the following Fig. 3 shows the snapshot of cloud editor showing a number of cubes placed at different location and having different number of particles and size of particle radius. The cube in red color is active one and its attributes such as size, location, number of particles and size of radius can be changed by using various controls.

Table 3: Data file generated by snapshot of Fig. 5

Line	Input Data
1	615
2	23
3	200, 7.39642, 1.07048, 2.89284,-3.4,-3.3,-0.600002, 2.1
4	10, 1.21266, 1.14308, 0.963248, 2.7,-3.1,-0.4, 2.2
5	5, 1.36688, 1.05379, 1.07216, 1.1,-3.2, 0, 2
6	100, 4.99599, 1.30149, 2.33467, 1.1,-0.899999, 0, 3
7	110, 3.02839, 1.58117, 3.05433,-3.8,-0.499999, 0, 3
8	10, 1, 1, 1, 2.5, 0.600001, 0, 2
9	8, 1, 1, 1,-0.399999, 0.800001, 0, 3
10	7, 1, 1, 1, 1.1, 0.500001, 0, 2.2
11	5, 1, 1, 1, 1.5, 1.8, 0, 2.4
12	8, 1, 1, 1,-2, 1.3, 0, 2.123
13	9, 1, 1, 1, 6.12438e-007, 2.1, 0, 2
14	50, 2.36429, 1.50285, 2.29693, 4.6,-3, 0, 2.2345
15	8, 1, 1, 1,-4, 2.5, 0, 2
16	10, 1, 1, 1, 0.500001, 3, 0, 2.5
17	7, 1, 1, 1,-4, 1.3, 0, 2
18	7, 1, 1, 1,-2.8, 2, 0, 2
19	10, 1, 1, 1,-1.4, 2.6, 0, 2
20	10, 1, 1, 1, 4.8, 1.9, 0, 2
21	7, 1, 1, 1, 3.2, 2, 0, 2
22	10, 1, 1, 1, 4, 0.600001, 0, 2.1
23	9, 1, 1, 1, 2.2, 2.3, 0, 2
24	5, 1, 1, 1, 3.9, 2.3, 0, 2.123
25	10, 1.62329, 1.19516, 1.31986, 4.7,-0.999999, 0, 3

In the following, Table 2 shows the data file generated by cloud macrostructure editor using 17 cubes as shown in Fig. 3. The total number of particles used to model cloud in this system is 554. T

In the following, Fig. 4 shows the snapshot of cloud modeled by 554 particles according to data generated by the cloud macrostructure editor shown in Fig. 3. The data used by the proposed algorithm to model cloud is shown in Table 2. By making use of data of Table 2, the cloud shape modeling algorithm modeled cloud data. This cloud modeled data was rendered in the renderer presented by[50], Fig. 4 shows this rendered image.

Figure 5 shows another snapshot of cloud editor showing 23 cubes placed at different locations to represent apparent shape of cloud.

Table 3 shows the data file generated by cloud macrostructure editor using 23 cubes as shown in Fig. 5. The total number of particles used to model cloud in this system is 615.

Similarly, the Fig. 6 shows the snapshot of cloud modeled by 615 particles according to data generated by the cloud macrostructure editor shown in Fig. 5. The data used by the proposed algorithm to model cloud is also shown in Table 3. The image shown in Fig. 6 has also been rendered by the renderer presented by[50].

EFFICIENCY OF THE RANDOMIZED ALGORITHM

We have performed test for efficiency of the randomized algorithm. For this purpose, process time for a Pentium® III 797 MHz Processor having 128 MB

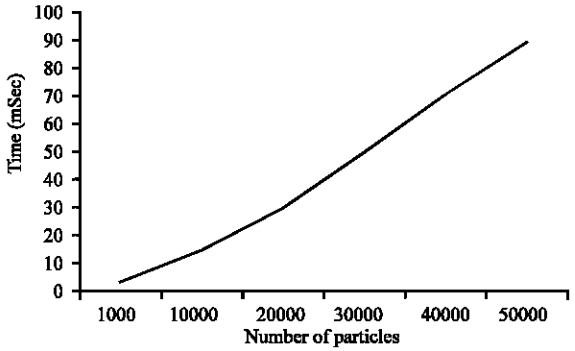

Fig. 7: Test result for efficiency of the algorithm

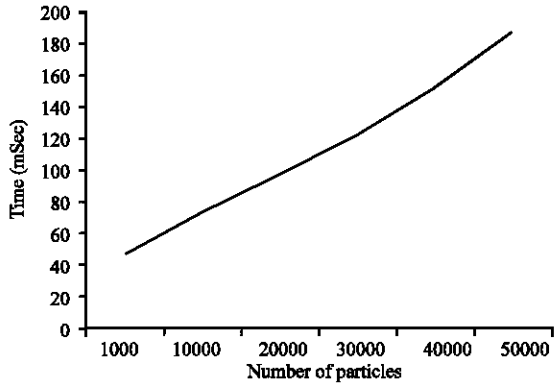

Fig. 8: Test result for efficiency of the algorithm

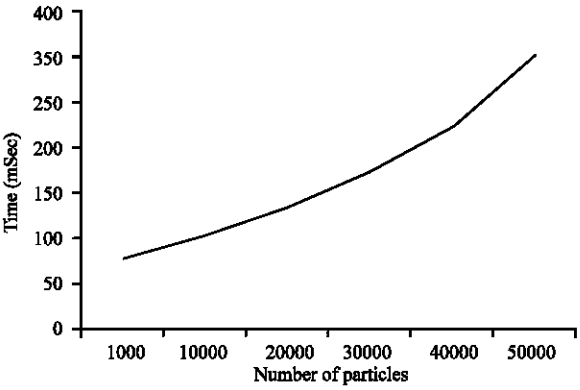

Fig. 9: Test result for efficiency of the algorithm

taken by the algorithm to model the cloud data has been calculated for a number of particles. The machine used for this purpose is Intel Pentium® IV 3.2 GHz having 1 GB RAM. Figure 7 through Fig. 9 show the graphs for these tests. Figure 7 shows resulting graph for a Pentium® IV 3.2 GHz Processor having 1 GB RAM. Figure 8 shows resulting graph for a Pentium® IV 2.0 GHz Processor having 248 MB RAM. Figure 9 shows resulting graph

RAM. It is obvious that as the number of particles is increased, the process time taken by the algorithm to model cloud data is almost increasing linearly.

RENDERING PERFORMANCE TEST RESULTS

We have implemented our cloud shape modeling system using OpenGL. The rendering of the clouds has been done using the method described[50,51].

We have performed several performance tests for rendered images of clouds modeled by our proposed randomized cloud shape modeling algorithm. Our first test machine was Pentium® IV processor; the details of this system-*System1* are listed in the following Table 4.

In this test, we created the data for cloud particles using our proposed randomized algorithm, distributed the cloud particles randomly in the cloud bounding volume and then rendered the resulting cloud images. Figure 5 shows the results of the performance test for a resolution of 800×600, 960×600 and 1024×876. The tests have been performed for a maximum number of 50000 particles.

Figure 10 shows the test results for the system having Intel Pentium IV 3.2 GHz, 1GB RAM and AGP as nVADIA GeForce FX 5950 Ultra. It is observed that for a less number of particles such as 500 particles, the frame rate is more than 120 frames per second whereas for a large number of particles we have frames rate around 20 frames per second. It has been observed that a small piece of cloud can be modeled by using a number of particles around 500 to 1000. This shows that even for scenes consisting of several thousands particles we can achieve interactive frame rates and the proposed method is applicable to design cloud shapes interactively.

Table 4: Specifications of test system-system1

Processor	Intel Pentium IV 3.2 GHz
System RAM	1 GB
Graphics Card	nVADIA GeForce FX 5950 Ultra
RAM on GPU	256 MB

Table 5: Specifications of test system-system2

Processor	Intel Pentium IV 1.5 GHz
System RAM	384 MB
Graphics Card	ATI Radeon 9600 XT
RAM on GPU	128 MB

Table 6: Specifications of test system-system3

Processor	Intel Pentium IV 1.4 GHz
System RAM	256 MB
Graphics Card	nVADIA GeForce FX 5200
RAM on GPU	128 MB

Table 7: Specifications of test system-system4

Processor	Intel Pentium IV 2.8 GHz
System RAM	512 MB
Graphics Card	nVADIA GeForce FX 5200
RAM on GPU	128 MB

Table 8: Specifications of test system-system5

Processor	AMD Athlon™ X 1600+1.4 GHz
System RAM	256 MB
Graphics Card	nVADIA GeForce MX
RAM on GPU	64 MB

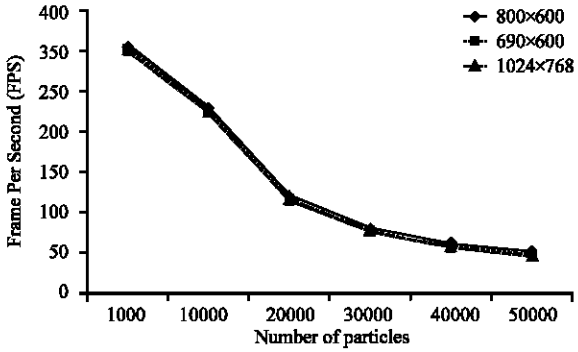

Fig. 10: Performance Test results for System1

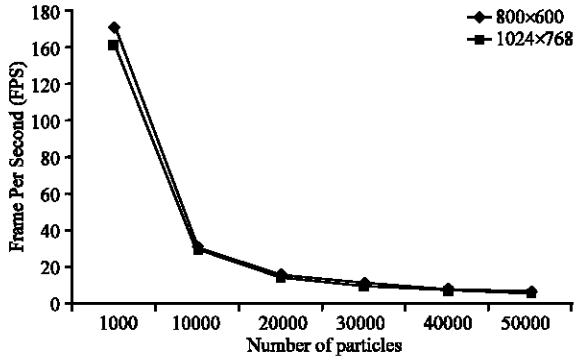

Fig. 11: Performance Test results for System2

In order to further test performance of proposed algorithm, a number of performance tests were conducted on other machines having different specifications and with a different variety of AGP card. The specifications of these test machines are listed in the following tables from Table 5 to Table 8. These systems have been labeled as System2, System3, System4 and System5.

Figure 11 shows the results of the performance test for *System2* for a resolution of 800×600 and 1024×876. The hardware on this system only supported these two mentioned resolution modes and the other resolution modes were not available for testing.

Figure 11 shows the test results for a test system having processor as Pentium IV 1.5 GHz, 384 MB RAM and AGP card as ATI Radeon 9600 XT. As shown in the graph, for less number of particles, the frame rate is very good. Between 500 particles and 10000 particles, the frame rate drops quickly and then it drops down linearly. The AGP card used in this system is ATI Radeon 9600 XT.

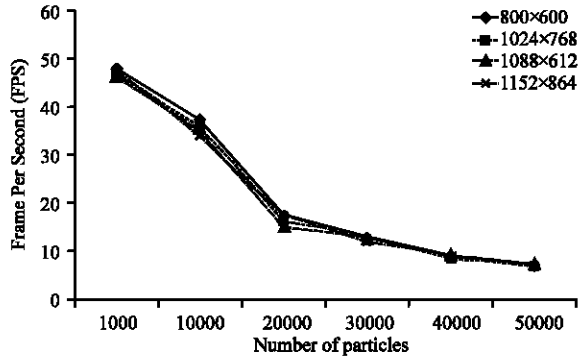

Fig. 12: Performance Test results for System3

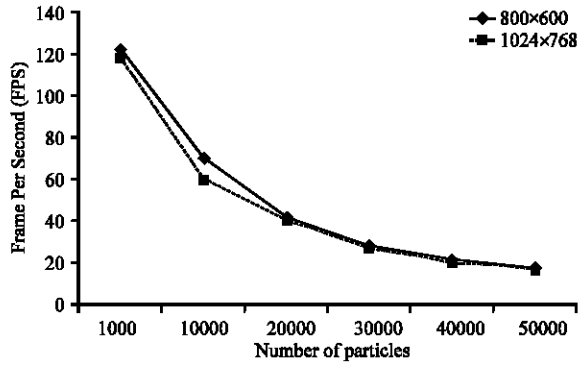

Fig. 13: Performance Test results for System4

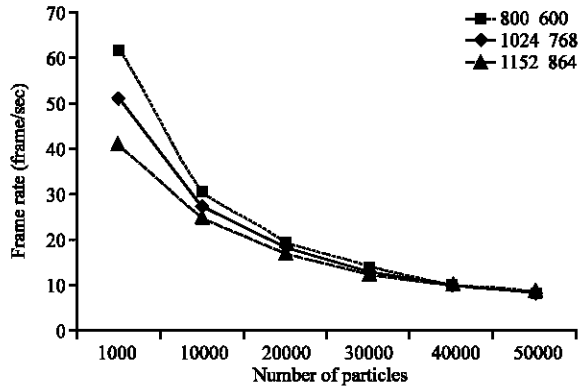

Fig. 14: Performance Test results for System5

This type of behavior looks due to limitation of computational power of AGP card. For higher number of particles, the frame rate drops below 20 frames per second, but even then this is quite reasonable to design clouds interactively on low power PCs.

Figure 12 shows the results of the performance test for *System3* for a resolution of 800×600, 1024×876, 1088×612 and 1152×864.

Figure 12 shows the test results for a test system having processor as Pentium IV 1.4 GHz, 256 MB RAM and AGP card as nVADIA GeForce FX 5200. As shown in the graph, for less number of particles, the frame rate is closer to 50 frames per second. For higher number of particles, the frame rate drops closer to 10 frames per second. In the series of tests, this is the low power machine and looks reasonable to design cloud images interactively using the proposed interactive environment proposed by this research.

Figure 13 shows the results of the performance test for *System4* for a resolution of 800×600 and 1024×876. The hardware on this system only supported these two mentioned resolution modes and the other resolution modes were not available for testing.

Figure 13 shows the test results for a test system having processor as Pentium IV 2.8 GHz, 512 MB RAM and AGP card as nVADIA GeForce FX 5200. As shown in the graph, for less number of particles, the frame rate is closer to 120 frames per second. For higher number of particles, the frame rate drops closer to 20 frames per second. As seen on the graph, the difference in the frame rate for the two mentioned resolution is very close and is insignificant.

Figure 14 shows the results of the performance test for *System5* for a resolution of 800×600, 1024×876 and 1152×864.

Figure 14 shows the test results for a test system having processor as AMD Athlon™ X 1600+1.4 GHz, 256 MB RAM and AGP card as nVADIA GeForce MX. As shown in the graph, for less number of particles, the frame rate is closer to 120 frames per second. For higher number of particles, the frame rate drops closer to 10 frames per second. As seen on the graph, the difference in the frame rate for les number of particles is considerable but this difference is insignificant for higher number of particles in the system.

In the Fig. 15-17 show the comparison of rendering performance test for 800×600, 1024×768 and 1152×864 resolutions.

Figure 15 and Fig 17 show comparison of frame rate per second for all systems for the resolution of 800×600 and 1024×768 respectively. For the number of particles up to 10000, the frame rate is more than required for real time rendering. For particles more than 20000, the frame rate is close to real time. For the System1 and System4, frame rate is close to real time even for 50,000 particles. System2, System3 and System5 have slightly less frame rate than required for real time as the number of particles increases from 20000; it is due the les power of Graphical Processing Units (GPU). The frame rate greater than real time shows

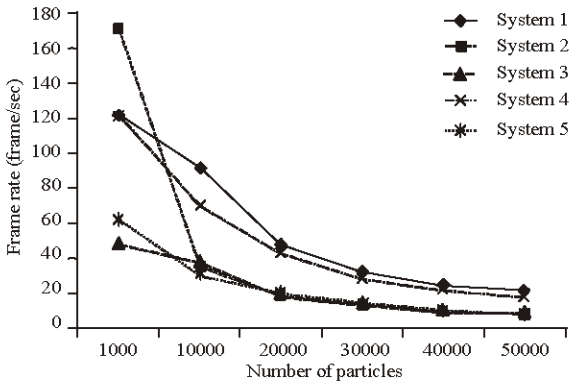

Fig. 15: Comparison of Performance Test for 800×600 resolution

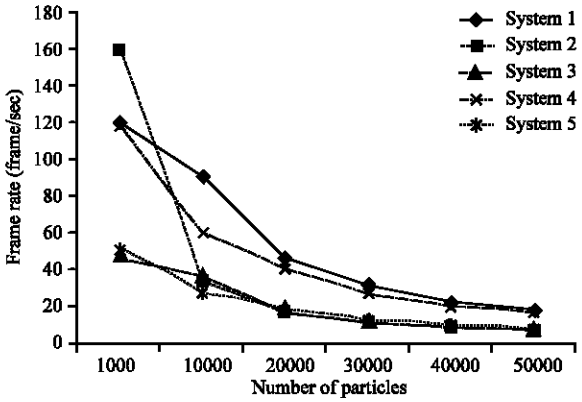

Fig. 16: Comparison of Performance Test for 1024×768 resolution

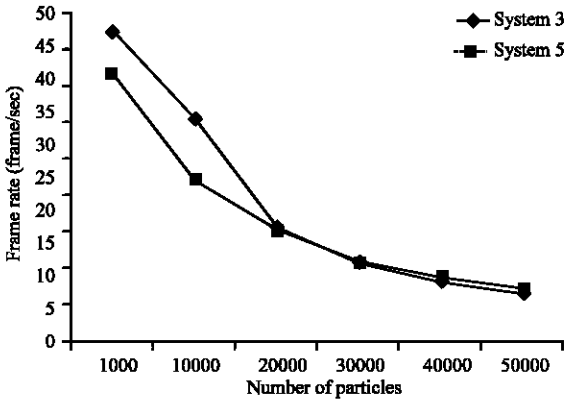

Fig. 17: Comparison of Performance Test for 1152×864 resolution

that the modeled clouds can be used in a virtual environment (VE), as clouds are one small part of a VE and should get less time for their processing. Figure 5 shows comparison of frame rate at 1152×864 for System 3 and System5 only, as the GPUs on other test machines don't support these resolution modes. The frame rate is dropped as the resolution is increased as expected. From above ures, we can deduce that the resolution mode of 600×800 is most sited for the efficient rendering of the clouds.

DISCUSSION

An interactive system for artistically designing cloud shapes has been demonstrated, that can be used for modeling visually convincing clouds using low-cost PC graphics hardware. Using a procedural-based two-level approach, we are able to create a more intuitive system for artists and animators and allow the designers to interact with their full cloud models at interactive rates.

This system is suitable for real-time simulation of cloud shapes for Virtual Environments (VE) applications such as flight simulators and games. These applications require realism, but cannot afford to sacrifice speed to achieve it. Controlling shapes of clouds is a difficult task as they are amorphous objects. Clouds can be built by filling a volume by particles or by using an editing application that allows users to place particles, have control over number of particles and have control over size that can build clouds interactively. The randomized method is a good way to get a quick field of clouds. A combination of randomized method and an editing application serves a good choice for designing cloud for Virtual Reality (VR) applications as they have a range of designed levels and need more control over details of scenes. Providing editing applications for such Virtual reality applications can produce beautiful cloud shapes tailored to the need of the Virtual Environment (VE). The resulting images of clouds shapes, modeled by the proposed algorithm have been demonstrated. It shows these images are promising, easy to design using the proposed framework and almost any type of clouds can be modeled. The proposed framework and algorithm are developed with these requirements in mind.

The efficiency of the randomized particles algorithm shows that the time taken for particles up to 1000 particles is about 1millisecond. Even for 50000 particles, the time taken by the algorithm is around 90 milliseconds. For modeling a reasonable good scene of clouds, a number of particles around 10000 to 20000 is enough, which shows that algorithm can model cloud data very quickly from stage to stage as needed by the VE applications, without interrupting the user.

The tests for the performance of the proposed cloud shape modeling algorithm has been performed on various low cost PCs having a variety of processors such as Intel Pentium IV having CPUs varying from 1.4 GHz to 3.2 GHz and of from different and an AMD Athlon 1.4 GHz

processor. These test machines have different ordinary Accelerated Graphics Port (AGP) cards. The complete specifications of these test machines are listed in Efficiency of the Randomized Algorithm. As shown in Fig. 5.2-5.9, for less number of particles the achieved frame rates are more than hundred frames per second. This frame rate is more than required for real time application. The reason is as we are testing clouds only, whereas in a VE clouds are just a small part and should demand a less processing time. This shows the suitability of the modeled clouds for VE. Even for larger number of particles, the frame rates are in the range of 20-10 frames per second which looks reasonable for synthesizing cloud shapes interactively. This shows that the cloud shape modeling framework proposed by this research is reasonable good for synthesizing cloud shapes interactively for virtual reality applications on low-cost PCs.

REFERENCES

1. Mark, J. Harris and V. Willium Baxter III, 2003. Thorsten scheuermann and anselmo lastra simulation of cloud dynamics on graphics hardware, SIGGRAPH/Eurographics Workshop on Graphics Hardware.
2. Schpok, J., J. Simons, S. David Ebert and C. Hansen, 2003. A real-time cloud modeling, rendering and animation system. Eurographics/SIGGRAPH Symposium on Computer Animation.
3. Mark, J. Harris, 2002. Real-time cloud rendering for games, Game Developer Conference Proceedings.
4. Andrzej Trembilski, 2002. Surface-based efficient cloud visualisation for animation applications, Proceedings WSCG.
5. Andrzej Trembilski andreas BroBler, 2002. Transparency of Polygon Based Cloud Rendering, Proceedings of the ACM Symposium on Applied Computing (ACM SAC 2002), pp:785-790.
6. Paul Heinzlreiter, Gerhard Kurka, 2002. Jens volkert: Real-time visualization of clouds, WSCG'2002-the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002, Czech Republic.
7. Mark J. Harris and Anselmo Lastra 2001 Real-time cloud rendering. Eurographics 2001, 20.
8. Nishita, T. and Y. Dobashi, 2001. Modeling and rendering of various natural phenomena consisting of particles, Proceedings of Computer Graphics International, pp: 149-156
9. Andrzej Trembilski, 2001. Two methods for cloud visualization from weather simulation data, The Visual Computer, 17: 179-184.
10. Andrzej Trembilski, 2001. Two methods for cloud visualization from weather simulation data, The Visual Computer, 17: 179-184.
11. Dobashi, Y., K. Kaneda, H. Yamashita, T. Okita and T. Nishita, 2000. A simple, efficient method for realistic animation of clouds, ACM SIGGRAPG, pp: 18-28.
12. Pantelis Elinas and Wolfgang Stürzlinger, 2000. Real-time rendering of 3D clouds. J. Graphics Tools, 5: 33-45.
13. Nishita, T. And Y. Dobashi, 1999. Modeling and rendering methods of clouds, Pacific Graphics 99, pp: 218-219.
14. Yoshinori Dobashi, Tomoyuki Nishita, Hideo Yamashita and Tsuyoshi Okita 1999. Using metaballs to modeling and animate clouds from satellite images, The Visual Computer 15: 471-482.
15. Nishita, T., Y. Dobashi, Modeling and rendering methods of clouds, Pacific Graphics, 99: 218-219.
16. Westover, L., 1990. Footprint evaluation for volume rendering SIGGRAPH 1990, pp: 367-376.
17. David S. Ebert and Edward Bedwell, 1998. Implicit Modeling with Procedural Techniques, Proceedings Impliict Surfaces '98.
18. Dobashi, Y., T. Nishita and T. Okita 1998. Animation of clouds using cellular automaton. In Proceedings of Computer Graphics and Imaging'98, pp: 251-256.
19. Ebert, D.S., 1997. Volumetric modeling with implicit functions: A cloud is born, Visual Proc. of SIGGRAPH'97, pp: 147.
20. Nishita, T., Y. Dobashi and E. Nakamae, 1996. Display of clouds taking into account multiple anisotropic scattering and sky light, Proc. of SIGGRAPH'96, pp: 379-386.
21. Ebert, D.S. and R.E. Parent, 1990. Rendering and animation of gaseous phenomena by combining fast volume scanline a-buffer techniques, SIGGRAPH pp: 357-366.
22. Lamorlette, A. and N. Foster 2002. Structural modeling of natural flames. ACM Transactions on Graphics, 21.
23. Stam, J., 1994. Stochastic rendering of density fields, Proc. of Graphics Interface'94, pp: 51-58.
24. Nagel, K. and E. Raschke, 1992. Self-organizing criticality in cloud formation?, Phisica A, 182: 519-531.
25. Kaneda, K., T. Okamoto, E. Nakamae and T. Nishita, 1991. Photorealistic image synthesis for outdoor scenery under various atmospheric conditions, The Visual Computer, 7: 247-258.
26. Ebert, D.S. and R.E. Parent, 1990. Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques, Computer Graphics, 24: 357-366.

27. Gardener, G.Y., 1985. Visual simulation of clouds, Computer Graphics, 19: 279-303.
28. Voss, R., 1983. Fourier synthesis of gaussian fractals: 1/f noises, landscapes and flakes, SIGGRAPH'83: Tutorial on State of the Art Image Synthesis, 10.
29. Reeves, W., 1983. Particle systems-a technique for modeling a class of fuzzy objects, SIGGRAPH'83, pp: 359-376.
30. Fournier, A., D. Fussel and L. Carpenter, 1982. Computer rendering of stochastic models. Commun. ACM25, 6: 371-384.
31. Voss, R., 1985. Random Fractal Forgeries. In R.A. Earnshaw, Eds., Fundamental Algorithms for Computer Graphics. Springer-Verlag.
32. Perlin K., 1985. An image synthesizer. In Computer Graphics (Proceedings of SIGGRAPH 85), 19.
33. Dobashi, Y., T. Nishita, H. Yamashita and T. Okita, 1998. Modeling of clouds from satellite images using metaballs. Pac-c Graphics, 98.
34. Kniss, J., S. Premoze, C. Hansen and D. Ebert, 2002. Interactive translucent volume rendering and procedural modeling. In Proceedings of the conference on Visualization '02. IEEE Press.
35. Neyret, F., 1997. Qualitative simulation of convective clouds formation and evolution, EGCAS'97, pp: 113-124.
36. Stam J., 2000. Interacting with smoke and fire in real time. Communications of the ACM, 43.
37. Max, N., 1994. Efficient light propagation for multiple anisotropic volume scattering, Proc. of the 5th Eurographics Workshop on Rendering, pp: 87-104.
38. Kajiya, J.T. and B.P.V. Herzen, 1984. Ray tracing volume densities, Computer Graphics, 18: 165-174.
39. Stam, J. and E. Fiume, 1995. Dipicting fire and other gaseous phenomena using diffusion processes, Proc. of SIGGRAPH'95, pp: 129-136.
41. Kikuchi, T., K. Muraoka and N. Chiba, 1998. Visual simulation of cumulonimbus clouds, J. Institute of Image Electronics and Electronics Engineers of Japan, 27: 317-326 .
42. Ebert, D., F. Musgrave, D. Peachey, K. Perlin and S. Worley, 2002. Texturing and Modeling: A Procedural Approach. Morgan Kaufmann, 3 Edn.
43. Blinn, J., 1982. Light reflection functions for simulation of clouds and dusty surfaces. In Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques.
44. Max, N., 1995. Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics, 1.
45. Stam, J., 1995. Multiple scattering as a diffusion process. Eurographics Rendering Workshop.
46. Levis, J., 1989. Algorithms for solid noise synthesis. SIGGRAPH, pp: 263-270.
47. Rademacher. P., 1999. Glui, A GLUT-BASED USER INTERFACE LIBRARY, 2nd Edn.
48. Gunadi, W.N., 2003. Modified sweep algorithm with fuzzy-based parameters for public bus route selection. Universiti Teknologi Malaysia: Ph.D. Thesis.
49. Muhammad Azam Rana, Mohd Shahrizal Sunar, Mohd Norikhwan Nor Hayat, Sarudin Kari and Abdullah Bade, 2004. Framework for real time cloud rendering. International conference on computer graphics, imaging and visualization (CGIV04). In Proceeding of IEEE Computer Society Press. Penang, Malaysia, pp: 26-29.
50. Harris, M.J., 2002. Real-time cloud rendering for games. Proceedings of Game Developers Conference 2002.
51. Harris, M.J., V.B. Willium, III, S. Thorsten and L. Anselmo, 2003. Simulation of cloud dynamics on graphics hardware. SIGGRAPH/Eurographics Workshop on Graphics Hardware.