芸術科学会論文誌 Vol. 6 No. 4 pp. 179 - 196

An Efficient Hybrid Method for Animating the Growth of Large-Scale Cumulus-Type Cloud

Mamat Abdukadir*, Tadahiro Fujimoto*, Mamtimin Geni**, and Norishige Chiba*
*Faculty of Engineering, Iwate University, Japan
**Dept. of Mech. Eng., Xinjiang University, China
E-mail: {mamat, fujimoto, nchiba}@cis.iwate-u.ac.jp

Abstract

We present an efficient method for creating large-scale animations of vertical developing cumulus-type cloud growth. The dynamics of cloud formation, growth, and motion are complex phenomena, and depicting these dynamics remains a significant challenge in the area of simulation and animation of natural phenomena in computer graphics. A novel aspect of this paper is the combination of a physical simulation method and a stochastic simulation method for obtaining an animation of large-scale phenomena with effects that are more natural. The physical simulation method is used to accurately solve fluid dynamics on a relatively small scale and prepare a 3D primitive pattern of a realistic animation of cloud growth. The stochastic simulation method uses $1/f^{\beta}$ noise functions and performs a large-scale simulation of an air current caused by the rising and condensation of water vapor due to the thermal effect. Many copies of the 3D primitive pattern are recursively mapped into the air current to constitute a large-scale continuous cloud growth. This combination of the physical and stochastic simulation methods can animate large-scale phenomena efficiently without enormous computational time and memory. The physical and stochastic simulations are achieved by particle-based methods, and the mapping is applied on simulated particles. Experimental results show that the proposed method efficiently creates realistic animations of large-scale cumulus and cumulonimbus clouds.

1. Introduction

Animation techniques for representing large-scale natural phenomena, such as the clouds shown in Figure 1, are a very challenging task in the field of computer graphics. There are mainly two ways to produce realistic animations of natural phenomena. One is to employ a physical-based simulation method, which accurately simulates the physical process of a natural phenomenon, such as computational fluid dynamics. The other is to use a mathematical-based procedural method, which approximates a natural phenomenon as a mathematical model. When it comes to gaseous or fluid phenomena, various physical-based simulation methods have been proposed to produce realistic animations [2, 3, 5, 10, 11, 12, 14, 23, 24, 26]. However, in general, such simulation methods

Figure 1. Large-scale cumulus clouds produced by the proposed method

demand large amounts of computational time and memory. In contrast, mathematical-based procedural methods, such as implicit functions [4], fractals [4, 21, 29], Fourier synthesis [4, 6], and noise functions [4, 20], tend to be computationally inexpensive and easier to implement. However, the main challenge with such procedural methods is to obtain realistic-looking animations [3].

Physical-based fluid simulation methods are categorized into two types. One is grid-based methods, which divide a simulation space into grid cells and calculate physical values on the grid cells. The other is particle-based methods, which use particles to represent fluid and calculate the motion of the particles. In particular, particle-based methods have the great advantage that the fluid can change the topology of its shape freely, which means the method can be applied to complex shapes and the boundaries of gaseous phenomena, such as flames [2], clouds [3], and smoke [5, 23, 26]. However, in order to simulate a large-scale phenomenon, enormous computational time and memory are requisite to compute the interactions between huge amounts of particles [10, 11, 14].

In this study, we propose a method for creating large-scale animations of vertical developing cumulus-type cloud growth efficiently. This method employs particle-based simulations and needs less computational time and memory than common particle-based simulation methods [10, 11, 14, 27]. The proposed method utilizes a hybrid approach that combines a *physical* simulation method and a *stochastic* simulation method for efficiently producing realistic motions of numerous *cloud particles*. During the formation and growth process of clouds in the atmosphere, water goes from an evaporation phase to a condensation phase [30]. Evaporation is the process in which the water changes into *water vapor* due to heating the ground by the sun. The generated water vapor is discharged into the atmosphere and continuously rises due to the effect of buoyancy caused by the difference of temperature. This is an invisible process. Condensation is the process in which the water vapor changes into *droplets* due to cooling and then clouds begin to form and grow. This is a visible process. These processes are very complex and difficult to represent with a continuous growth effect. The proposed method represents the water vapor and clouds by using *particles* and effectively represents the above complex phenomenon. The proposed hybrid method is summarized as follows.

1. Physical fluid simulation method: A small-scale 3D primitive pattern of the cloud growth, which is called a *3D cloud primitive set* (Figure2), is prepared. This 3D cloud primitive set consists of a set of *fluid particles* that are simulated by a particle-based physical fluid dynamics simulation method, which solves the incompressible Navier-Stokes equations.

2. Stochastic simulation method: First, a 2D density map is set on a ground surface. The map is defined using a 2D $1/f^{\beta}$ noise function, and it greatly affects the result of the simulation stochastically. Then, *water vapor particles* are initially generated on the ground according to the intensity of the 2D $1/f^{\beta}$ density map, which imitates the evaporation process from water to water vapor on the ground. Each individual water vapor particle is given several attributes, such as position, velocity, mass, temperature and dew point temperature threshold. Then, the rising and condensation processes of the water vapor particles are simulated under the thermal effect. After reaching the dew point altitude, the water vapor particles become *condensation particles* and continue to rise.

3. Hybrid method: After condensation, copies of the 3D cloud primitive set begin to be mapped on some selected condensation particles, and fluid particles in the copies become cloud particles to represent cloud growth. Afterward, the cloud particles are mapped recursively to produce a continuous cloud growth. This recursive mapping is the essence of the hybrid approach that combines fluid particles accurately simulated by the physical simulation method and condensation particles (water vapor particles) stochastically simulated by the stochastic simulation method.

In the next section, we discuss previous methods used for the generation and animation of clouds in computer graphics. In Section 3, we describe the cloud formation process in nature. In Section 4, we explain the physical fluid simulation method that we employ for producing the 3D cloud primitive set. In Section 5, we present our hybrid method in detail. In Section 6, we show some experimental results produced by the proposed method. Finally, we present our conclusions in Section 7.

2. Previous work

There is a considerable amount of research on simulation and animation of clouds for computer graphics. Nagel et al. [18] presented a numerical

model of cloud growth based on a simple Boolean function iterated within a volumetric data space. This method, called a cellular automaton, can give fairly realistic-looking growth patterns for the formation of clouds. The simulation can be done using low quantities of computational time. However, this method tends to produce symmetrical cloud structures. Later, Dobashi et al. [3] extended this method to 3D space to demonstrate more realistic cloud growth. Kajiya et al. [7] proposed a method for animating clouds using a numerical simulation of the fluid dynamics of the atmosphere. This method is very complex and time-consuming. Kaneko [8] used the CML (Coupled Map Lattice) method to simulate gaseous phenomena, specifically clouds. More recently, this method has been applied by Miyazaki et al. [12] to generate clouds based on atmospheric fluid dynamics and has produced high-speed formations of cirrus clouds. Neyret et al. [19] proposed a method for forming clouds using qualitative simulation. This method is quite unique in its focus on the formation of clouds, and hence its suitability for animation is unknown.

The physical simulation of gaseous phenomena such as clouds is better represented if it is based on computational fluid dynamics. Stam et al. [24] developed a fast and reliable method in which Navier-Stokes equations are discretized using the finite element method, which is executed with large time steps. However, creating a large-scale animation of cloud growth using only physical simulation methods is still time-consuming. Harries et al. [15, 16, 17] implemented Stam's methods on the GPU. This method would be excellent for generating images of static clouds because the algorithm is mainly considered for the rendering method of clouds on the GPU based on the Dobashi's [3] method. However, the suitability of the Harris method for the growth effect of dynamic clouds is also unknown. Kikuchi et al. [9, 10] proposed a method utilizing the particle method to make an animation of cumulus clouds. Their dynamic simulation of particles forms cumulus clouds and takes into account several forces, such as the gravitational force and attraction/repulsion forces. Takeshita et al. [27] proposed an extended method by adding entrainment, adiabatic cooling, and latent heat effects to the Kikuchi method. However, this method implies an expensive simulation process to create large-scale animations.

More recently, the focus of attention has moved to the problems associated with the large-scale presentation of natural phenomena in computer graphics. Rasmussen et al. [23] proposed a method for representing large-scale phenomena such as smoke. In this method, a small volume of 2D simulation data was obtained from the physical process, solved using a semi-Langrangian method [24], and was then extended to the 3D field by using non-physics processes such as spectrum theory, thus significantly reducing the computation time and memory requirements. Wang et al. [22] also presented almost the same idea for creating animations of ocean water.

3. Physical process of cloud formation

Clouds are formed when air containing water vapor is cooled below a critical temperature, called the *dew point temperature threshold*, and the water vapor condenses into droplets or tiny particles in the atmosphere. Another requirement for clouds to form is the presence of ascending air [30, 31]. This is most often caused in the atmosphere by thermal currents (processes), caused by temperature variations within the atmosphere. When the sun irradiates the ground, the ground temperature increases and the air molecules near the ground are heated. This generates buoyancy due to the temperature difference and the air molecules begin to rise. As this happens, the air expands and the pressure decreases, in turn causing the air to cool. Cool air can hold less water vapor than warm air, resulting in increased relative humidity as a moist air parcel rises. Eventually, the rising air reaches the dew point temperature threshold, at which time condensation occurs and a cloud begins to form. This is not the end of the process, because condensation is an exothermic process [31] that causes the release of the latent heat of condensation. The transfer of energy increases the temperature of the nearby air, creating more buoyancy and increasing the vertical velocity. This is the reason that clouds appear to grow upward, especially when the surrounding air is calm [25]. In a later section, we show how our method performs the animation of cloud growth based on this physical process.

4. Physical fluid simulation method

As described in Section 3, clouds are composed of water droplets resulting from the rising evaporation and condensation process of water

vapor in air currents by the thermodynamics. In our proposed method, we try to model these cloud formation processes by a stochastic simulation using particles, as described in Section 5. Simulating a cloud motion exactly by a physical fluid dynamics simulation is one of the most effective and practical choices for creating a realistic animation of cloud growth because the clouds are formed by the air currents governed by fluid dynamics. However, in general, a physical fluid simulation needs lots of computation time and memory even for a small-scale fluid; so, naturally, it is difficult to animate a fluid on a large scale. Therefore, our proposed method takes a strategy in which a pre-computed small-scale simulation result is reused many times in a large-scale space to produce a large-scale result. This efficient strategy greatly reduces the necessary computation time and memory. Our essential idea comes from the recursive structure of clouds. We first produce a small-scale 3D primitive pattern of cloud motion, which is called a 3D cloud primitive set (Figure2), using a physical fluid simulation. Then, lots of copies of the 3D cloud primitive set are recursively mapped into the air currents simulated by the stochastic simulation of the thermodynamics mentioned above. This mapping produces a large-scale continuous cloud growth. This approach means that we propose a hybrid method that combines a physical fluid simulation for the accurately computed cloud motion with a stochastic simulation for the cloud formation processes. We employ the particle-based fluid simulation method and then combine it with the particle-based stochastic simulation. In addition, in contrast to a grid-based method, the particle-based method has an important advantage in that it easily allows arbitrary topological changes of the clouds.

We employ Takeshita's particle-based fluid simulation method [27] to produce a small-scale 3D cloud primitive set. This method discretizes the incompressible Navier-Stokes equations below using the particles-based method.

$$\nabla \cdot \boldsymbol{u} = 0$$
$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla \boldsymbol{u}\right) = -\nabla p + \mu\nabla^2 \boldsymbol{u} + \boldsymbol{f} \quad (1)$$

In Eq.(1), $\boldsymbol{u}$ is the fluid velocity, p is the pressure, ρ is the density, μ is the kinematic viscosity, and $\boldsymbol{f}$ is the external force. In addition, Takeshita's method uses the following thermodynamics equation.

$$\frac{\partial T}{\partial t} + (\boldsymbol{u}\cdot\nabla)T = \alpha\nabla^2 T - \Gamma u_z + QC_c \quad (2)$$

In Eq.(2), T is the temperature, α is the coefficient of thermal diffusion, Γ is the dry adiabatic lapse rate, u_z is the velocity in the vertical direction, Q is the coefficient of latent heat, and C_c is the condensation rate of water vapor. The buoyancy is considered as an external force. In the particle-based method, the viscosity is approximated through discrete particles, and the pressure gradient is modeled as the interaction force between particles. The entrainment is calculated using the heat conduction between particles. Figure2 shows a 3D cloud primitive set produced by this physical fluid simulation method;

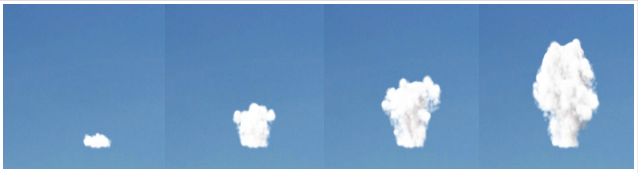


Figure2. 3D cloud primitive set produced by the physical fluid simulation method

the images in Figure2 are frame images extracted from the animation produced. This 3D cloud primitive set is used in the recursive mapping for the large-scale animation described in Section 5.

5. Hybrid method for large-scale animation

In this section, we explain in detail our algorithm for large-scale animation. In Section 5.1, we describe the stochastic simulation method based on 2D $1/f^{\beta}$ noise functions for simulating the rising (thermal) process of particles, and in Section 5.2, we describe how our method generates the growth of clouds.

5.1. Stochastic simulation method for rising and condensation of particles under the thermal effect

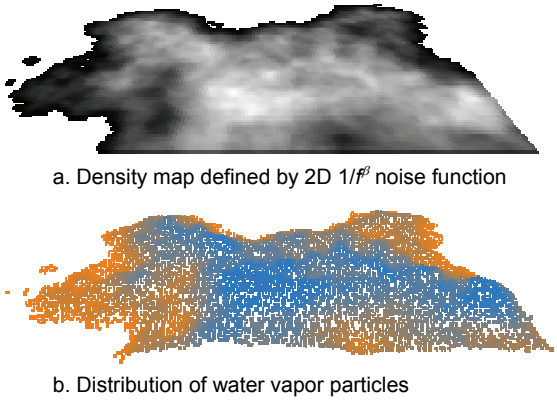
Clouds begin to form when evaporated water vapor begins to condense at the dew point altitude. To represent the evaporation effect of water, first, our method generates an initial arrangement of water vapor particles above the ground surface. The 2D ground surface is defined in the 3D simulation space; the ground surface is defined on the *x*-*y* plane, and the height is defined in the *z* direction, as shown in Figure 4, which will be described later in more detail. The ground surface is divided into grid cells of an arbitrary resolution. Each grid cell is given a density value from a density map, which is defined by a 2D $1/f^{\beta}$ noise function. An example of the density map is shown in Figure3.a; the regions with higher intensity are shown in the white color and the regions with lower intensity are shown in the black color. Then, water vapor particles are generated proportionally to the density values in the grid cells, i.e., more particles are generated on regions with high density, and fewer particles are generated on regions with low density. The number W_p of generated water vapor particles at a grid cell (ix, iy) is determined by the following equation:

$$W_p(ix, iy) = W_{p\_min} + (W_{p\_max} - W_{p\_min}) fNoise(ix, iy) \quad (3)$$

In Eq.(3), W_{p_min} and W_{p_max} are the minimum and maximum number of generated particles. The 2D $1/f^{\beta}$ density map $fNoise(ix, iy)$ gives a density value within [0.0,1.0] at each grid cell (ix, iy). Then, each water vapor particle is given several individual attributes, such as initial position, velocity, and temperature, while all particles are given the same values for mass and dew point temperature threshold throughout the simulation.

The initial 3D position $\boldsymbol{P}_v(t_0) = (P_{vx}(t_0), P_{vy}(t_0), P_{vz}(t_0))$ of a water vapor particle at the initial stochastic simulation time t_0 is determined by using the 2D $1/f^{\beta}$ density map $fNoise$. The initial vertical position $P_{vz}(t_0)$ of a particle at a grid cell (ix, iy) is determined by Eq.(4).

$$P_{vz}(t_0) = P_{vz\_min} + (P_{vz\_max} - P_{vz\_min}) fNoise(ix, iy) \quad (4)$$



a. Density map defined by 2D $1/f^{\beta}$ noise function

b. Distribution of water vapor particles

Figure3. Generation of water vapor particles from a density map

The heights P_{vz_min} and P_{vz_max} are the minimum and maximum heights that are allowed for the initial height of a water vapor particle. The initial horizontal position ($P_{vx}(t_0)$, $P_{vy}(t_0)$) is randomly set within the region of the grid cell (ix, iy).

The initial temperature $T_v(t_0)$ of a water vapor particle is set based on the initial height of Eq.(4) using Eq.(5).

$$T_v(t_0) = T_{v\_max} - (T_{v\_max} - T_{v\_min}) \frac{P_{vz}(t_0) - P_{vz\_min}}{P_{vz\_max} - P_{vz\_min}} \quad (5)$$

In Eq.(5), T_v is the temperature of a water vapor particle, which becomes a condensation particle during the simulation as described later. The temperature T_{v_max} and T_{v_min} are respectively the maximum and minimum temperatures. Eq.(5) means that a lower temperature is set for a particle with a higher height, and vice versa. An example of the initial distribution of water vapor particles using the 2D $1/f^{\beta}$ density map is shown in Figure 3.b. The orange particles have high temperature and lower height, and the blue particles have low temperature and higher height.

After arranging the water vapor particles on the ground surface at the initial time, we simulate the rising process of water vapor particles based on the thermal process described in Section 3. This process provides the preconditions for cloud formation caused by temperature differences in the simulation space. To model the water vapor particle rising, we take into consideration the buoyancy that occurs due to temperature differences between the water vapor particles and the environment. The buoyancy here is different from the one used in the Navier-Stokes

equations described in Section 4. The gravity effect on the water vapor particles is also taken into consideration. The buoyancy that acts on the water vapor particle is defined as follows:

$$f_{vbz}(t) = C_b(T_v(t) - T_e(t)) - mg \tag{6}$$

In Eq.(6), f_{vbz} represents the vertical component of the buoyancy $\boldsymbol{f}_{vb} = (f_{vbx}, f_{vby}, f_{vbz})$, C_b is the buoyancy coefficient, T_e is the environmental temperature, t is the stochastic simulation time, m is the mass of the water vapor particle, and g is the gravity acceleration in the vertical direction. We describe later how to determine the temperature T_v and T_e at a given time t.

The simulation advances by updating the velocity and position of each water vapor particle. The acceleration is defined by the buoyancy force $\boldsymbol{f}_{vb}$ and is integrated by using the Euler method with a time step increment Δt, which has the same value as that for the physical fluid simulation. The velocity $\boldsymbol{U}_v=(U_{vx}, U_{vy}, U_{vz})$ and position $\boldsymbol{P}_v = (P_{vx}, P_{vy}, P_{vz})$ of the water vapor particle are updated by the following equations.

$$\begin{aligned} \boldsymbol{U}_v(t+\Delta t) &= \boldsymbol{U}_v(t) + \Delta t \cdot \boldsymbol{f}_{vb}(t)/m \\ \boldsymbol{P}_v(t+\Delta t) &= \boldsymbol{P}_v(t) + \Delta t \cdot \boldsymbol{U}_v(t) \end{aligned} \tag{7}$$

In this process, the interaction force between particles is ignored. The wind force is also ignored; thus, the x and y components of the buoyancy force are zero.

The particle temperature T_v and environmental temperature T_e at a given time t, used in Eq.(6), are determined as follows:

Particle temperature: When a water vapor particle is rising until it reaches the dew point temperature threshold, its temperature decreases $10.0°C$ per kilometer. At the dew point, the water vapor particle condenses into a condensation particle. Afterward, the particle temperature decreases $6.0°C$ per kilometer. In our model, the particle temperature T_v is updated according to the altitude P_{vz} of the particle, as follows:

$$\begin{aligned} T_v(t+\Delta t) &= T_v(t) - C(P_{vz}(t+\Delta t) - P_{vz}(t)) \\ C &= \begin{cases} 10°C/km & (T_v(t) \geq T_{dp}) \\ 6°C/km & (T_v(t) < T_{dp}) \end{cases} \end{aligned} \tag{8}$$

The temperature T_{dp} is the dew point temperature threshold, and C is the coefficient for the particle's temperature decrease rate.

Environmental temperature: The environmental temperature T_e around a water vapor particle or a condensation particle is determined in consideration of the altitude P_{vz} of the particle as follows:

$$T_e(t) = T_{sl\_max} - C_e P_{vz}(t) \tag{9}$$

The temperature T_{sl_max} is the maximum temperature of the environment at the ground surface level, and C_e is the coefficient of the environmental temperature decrease rate. The coefficient C_e is set as $6.4°C$ per kilometer.

5.2. Recursive mapping for generating cloud particles

In the cloud growth phenomenon in nature, clouds are produced by the condensation effect that occurs at the altitude of the dew point temperature threshold. Our method represents this phenomenon by producing cloud particles from condensation particles, which are described in Section 5.1. Moreover, to create a large-scale animation of the cloud growth with continuous effects, cloud particles are produced recursively. In order to obtain realistic motions of the growth of clouds, our hybrid method combines the physical fluid simulation described in Section 4 with the stochastic simulation described in Section 5.1: the moving fluid particles provided by the physical fluid simulation are recursively mapped to condensation particles or cloud particles, according to the stochastic simulation.

After condensation particles are produced at the dew point altitude, some condensation particles are selected as source particles to generate cloud particles. Our method uses a set of moving fluid particles provided by the physical fluid simulation as a 3D cloud primitive set as described in Section 4, and the 3D cloud primitive set is mapped onto the position of each source particle. After this mapping, each fluid particle in the 3D cloud primitive set is treated as a cloud particle. Afterward, to produce cloud particles recursively, some cloud particles are selected as source particles, onto which the 3D cloud primitive sets are mapped in the next recursive level. This recursive mapping of "child" cloud particles onto "parent" cloud particles is repeated. That is,

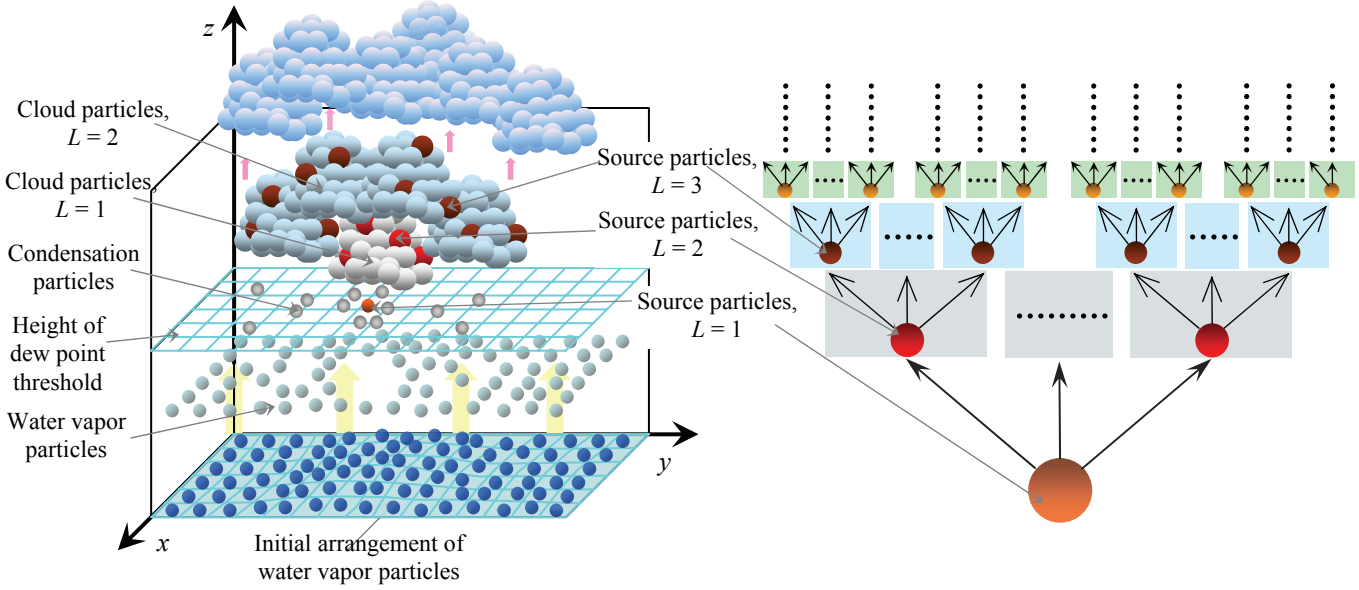


Figure4. Description of cloud generation process

some cloud particles of recursive level L, $L \geq 2$, become source particles to generate cloud particles of recursive level L+1. The source particles selected from among the condensation particles generate the cloud particles of recursive level $L = 1$.

The mapping of a cloud primitive set to a source particle in recursive level L, $L \geq 1$, is achieved as follows. However, it should be first noted that we use two kinds of time. One is the stochastic simulation time t, which is used in Section 5.1 and is the frame time of an animation in this section. The other is the physical simulation time s, which is used in the physical fluid simulation for a 3D cloud primitive set. We let $P_s[L, j](t)$, $j = 1, \ldots, G[L](t)$, denote the 3D position of a source particle j at time t, where $G[L](t)$ is the number of source particles of recursive level L existing at time t. We let $P_f[i](s)$, $i = 1, \ldots, N_f(s)$, denote the 3D position of a fluid particle i at time s in a cloud primitive set provided by the physical fluid simulation, where $N_f(s)$ is the number of fluid particles existing at time s. Then, the 3D position of a cloud particle (j, i) at time t in an animation is determined as the following $P_c[L, j, i](t)$:

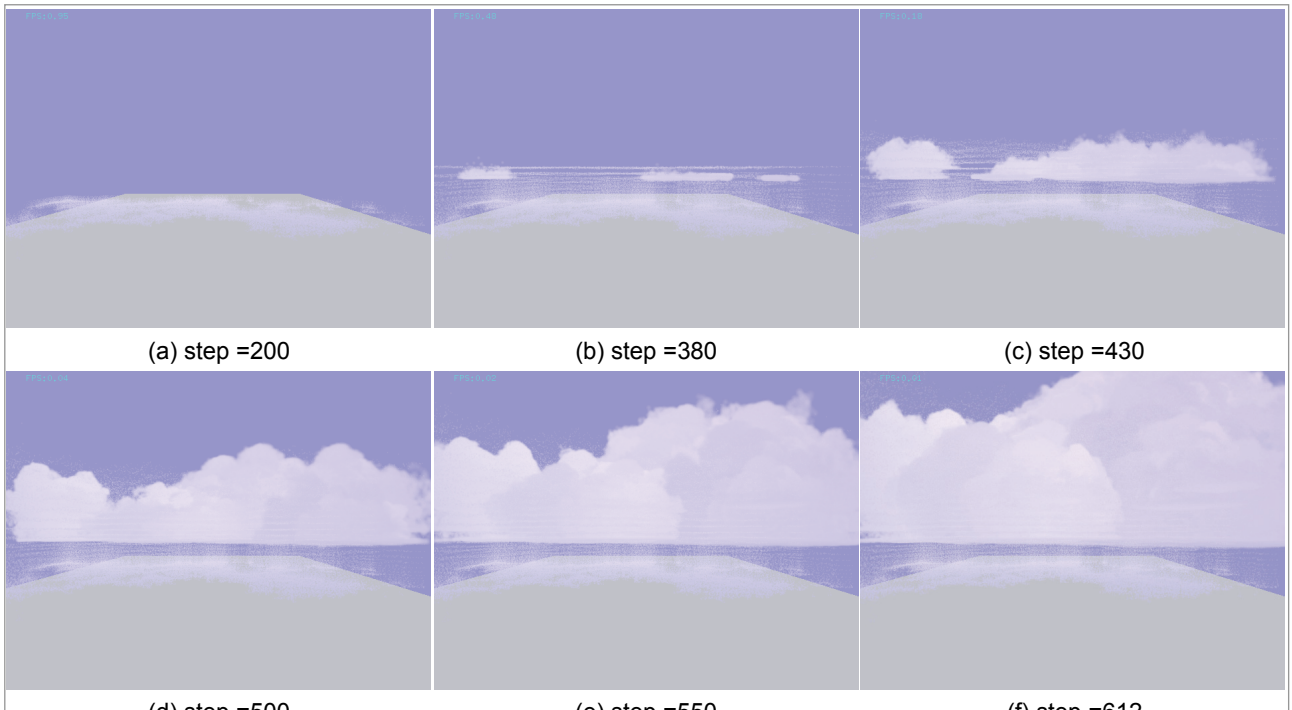
$$P_c[L, j, i](t) = P_s[L, j](t) + R[L, j]P_f[i](s) \quad (10)$$

In Eq.(10), the stochastic simulation time t and the physical simulation time s are related to each other as follows:

$$s = A[L, j](t\text{-}t_0[L, j]) \quad (11)$$

At the stochastic simulation time $t = t_0[L, j]$, the mapping of the 3D cloud primitive set to the source particle j starts at the physical simulation time $s = s_0 = 0$. The coefficient $A[L, j]$ is a speed control parameter. When $A[L, j]$ is larger, the time s proceeds more quickly and the mapped fluid particle i moves more quickly in the animation. This enables us to control the cloud growth speed at the position of each source particle in each recursive level using a common 3D cloud primitive set, which contributes to generating realistic cloud growth motion. In Eq.(10), the magnification factor $R[L, j]$ controls the range of motion of the fluid particle i, which represents the contraction and expansion effect of the cloud growth. We describe how to determine the values of A and R later.

As described above, to produce recursive cloud structures, source particles of recursive level L are selected from among the condensation particles or cloud particles of recursive level L-1. The 3D position of a source particle of recursive level L at time t, $P_s[L, j](t)$, $j = 1, \ldots, G[L](t)$, in Eq.(10) is determined as follows:



(a) step =200 (b) step =380 (c) step =430

(d) step =500 (e) step =550 (f) step =612

Figure5. Simulation progress for generating cloud particles by the hybrid method

$$P_s[L, j](t) = \begin{cases} P_v[r(j)](t), & L = 1 \\ P_c[L-1, r_j(j), r_i(j)](t), & L \geq 2 \end{cases} \quad (12)$$

In Eq.(12), $P_v[k](t)$, k = 1, …, $N_v(t)$, denotes the 3D position of a condensation particle at time t, where $N_v(t)$ is the number of condensation particles existing at time t. The 3D position P_c of a cloud particle is defined by Eq.(10). The number of cloud particles existing at time t is represented using the number of fluid particles in the 3D cloud primitive set, N_f, and Eq.(11) as follows:

$$N_c[L](t) = \sum_{j=1}^{G[L](t)} N_f\left(A[L, j](t\text{-}t_0[L, j])\right) \quad (13)$$

In the recursive level L, $G[L](t)$ source particles exist at time t, which have been randomly selected until the time t from among $N_v(t)$ condensation particles if L = 1, or $N_c[L\text{-}1](t)$ cloud particles if $L \geq 2$. The number $G[L](t)$ is determined by a selection ratio $M[L]$, $0 \leq M[L] \leq 1$, as follows:

$$G[L](t) = \begin{cases} M[L]\, N_v(t), & L = 1 \\ M[L]\, N_c[L-1](t), & L \geq 2 \end{cases} \quad (14)$$

The random selection of source particles in Eq.(12) is executed by random integer numbers $r(j)$, $r_j(j)$, and $r_i(j)$, j = 1, …, $G[L](t)$, such that

$$\begin{aligned} &1 \leq r(j) \leq N_v(t), \\ &1 \leq r_j(j) \leq G[L\text{-}1](t), \\ &1 \leq r_i(j) \leq N_f\left(A[L\text{-}1, r_j(j)](t - t_0[L-1, r_j(j)])\right) \end{aligned} \quad (15)$$

respectively. We can give an arbitrary value to the selection ratio $M[L]$ for each recursive level L to control the amount of cloud particles. On the other hand, the numbers N_f, N_v, N_c, and G are determined during the simulation process and are not given by the user: N_f results from the physical fluid simulation, N_v results from the stochastic simulation, N_c and G result from the hybrid simulation.

As shown in Eq.(11), the selection of a source particle j in a recursive level L is done at a stochastic simulation time $t_0[L, j]$, and the mapping of a 3D cloud primitive set to the source particle begins. The timing for setting $t_0[L, j]$ is very important to enhance the reality of the cloud growth. To control

this timing, we set a time range $[t_s[L], t_e[L]]$ for each recursive level L to select source particles j, such that $t_s[L] \le t_0[L, j] \le t_e[L]$. The starting time $t_s[L]$ of the time range is determined according to the time when the condensation particles or cloud particles of recursive level L-1 begin to appear. The ending time $t_e[L]$ is determined to control the interval of the time range and it is a user-controlled parameter.

In the method proposed above, the number of cloud particles generated above a certain region tends to be proportional to the density of the region on the 2D $1/f^{\beta}$ density map initially given to the ground surface. Therefore, we want the cloud growth in the higher-density regions to develop into the bigger clouds with quicker motion as compared to the lower-density regions. Therefore, the values of R and A in Eq.(10) and Eq.(11) are determined using the 2D $1/f^{\beta}$ density map *fNoise*, which is the same as the map used in Eq.(3) and Eq.(4). Experimentally, we use the following equations for a source particle j above a grid cell (ix, iy).

$$\begin{aligned} R[L, j] &= R_0 \; fNoise(ix, iy), & R_0 &= 1.5 \\ A[L, j] &= \left(1.0 + A_0 \; fNoise(ix, iy)\right), & A_0 &= 2.0 \end{aligned} \tag{16}$$

The above-mentioned mechanism is illustrated in Figure 4. In Figure 5, we show the simulation progress to create clouds by the proposed hybrid method. This simulation is carried out for 612 steps, and the density map has the resolution of 300x125. In Figure 5, not only cloud particles but also water vapor particles (and condensation particles) are rendered for illustration, although water vapor particles are not rendered in the examples in Section 6; in Figure 5, the water vapor particles are rendered in lighter white than the cloud particles. After initially arranging the water vapor particles on the ground at Step = 0, Figure 5(a) shows that the water vapor particles are rising. In Figure 5(b), some of the rising water vapor particles condense into condensation particles at the height of the dew point temperature threshold. Then, some source particles are selected from among the condensation particles, and a copy of the 3D cloud primitive set is mapped on each source particle for generating cloud particles of the recursive level L = 1; in Figure 5(b), three clumps of the deep white cloud particles appear above the light white water vapor particles and condensation particles. Afterward, the selection of source particles and the mapping of the copies of the 3D cloud primitive set are recursively repeated for generating cloud particles in consecutive recursive levels to animate the cloud growth, as shown in Figure 5(c), (d), (e), and (f).

Our proposed algorithm is summarized in the following pseudocode:

```
1:  // Initialization
2:  Prepare a 3D cloud primitive set by the particle-based
    fluid simulation method described in Section 4
3:  Generate a 2D-1/f^β density map by the FFT method
4:  Set stochastic simulation time t = t_0
5:  Generate water vapor particles above the ground
    surface using the density map, and initialize the
    attributes of the water vapor particles at time t_0 by
    Eq.(3), Eq.(4) and Eq.(5)
6:  // Hybrid simulation steps
7:  Set maximum recursive level L_max= 1
8:  for each time step t
9:      // Determination of condensation particles
10:     for each water vapor particle
11:         if (the particle temperature T_v(t) < T_dp)
12:             The water vapor particle becomes a
                condensation particle
13:         end if
14:     end for
15:     // Mapping of 3D cloud primitive set
16:     for each recursive level L, 1 ≤ L ≤ L_max
17:         if (time t is within the time range [t_s[L], t_e[L]])
18:             if (L = 1)
19:                 Select source particles j of level L from
                    condensation particles by Eq.(12), and
                    set t_0[L, j] to t
20:             else if (L ≥ 2)
21:                 Select source particles j of level L from
                    cloud particles of level L-1 by Eq.(12),
                    and set t_0[L, j] to t
22:             end if
23:             Start to map a copy of the 3D cloud
                primitive set to each selected source
                particle by Eq.(10). Then, the fluid particles
                in the 3D cloud primitive set become cloud
                particles of level L
24:         end if
25:     end for
26:     if (Cloud particles of level L_max appear above)
27:       Update maximum recursive level L_max = L_max + 1
28:     end if
29:     // Rising process of water vapor and condensation
        particles by stochastic simulation
30:     for each water vapor or condensation particle
31:         Compute the environmental temperature T_e(t)
            by Eq.(9)
32:         Compute the buoyancy force f_vbz(t) by Eq.(6)
33:         Update the velocity U_v and position P_v from t to
            t+Δt by Eq.(7)
```

Figure 6. An example of towering cumulus clouds

34: Update the particle temperature T_v from t to $t+\Delta t$ by Eq.(8)
35: ***end for***
36: Update stochastic simulation time $t = t + \Delta t$
37: ***end for***

6. Examples of large-scale animation

In this section, we demonstrate several vertical developing cumulus-type clouds generated using the proposed method and show the performance of the method.

6.1. Creating cumulus and cumulonimbus clouds

Cumulus type clouds are those that primarily exhibit vertical development. Two of the more common types are cumulus and cumulonimbus. Cumulus clouds are produced by buoyant vertical air motion. They are generally tall, with flat bases and rounded tops, and sometimes are described as resembling a cauliflower. Under certain conditions, cumulus clouds continuously grow vertically and can develop into a giant cumulonimbus, which is a thunderstorm cloud. These cumulonimbus clouds are much larger and more vertically developed than cumulus clouds. They can exist as individual towers or can form a line of towers called a squall line [31].

When observing cumulus clouds, we are actually observing the condensation process of rising thermal process of water vapor at a certain level in the atmosphere known as the condensation level or dew point temperature level. The thermal process of water vapor, which are invisible, rising above this level condenses spontaneously to form visible cloud particles [31]. Since the height of this level is almost constant at a particular time, then the bases of cumulus clouds are roughly flat. In this study, we present examples of these types of cumulus clouds. In these examples, based on the above discussion, we generated the clouds by setting the condensation level or dew point temperature as constant. Thus, the base of the clouds of the examples shown as follows seems roughly flat.

In Figure 6, we demonstrate a simple towering cumulus produced by the proposed method. In this simulation, the water vapor particles were generated from a density map of 64×64 with high density at the center. The maximum recursive level L_{max} is restricted to 5. The selection ratio $M[L]$ for source particles in Eq.(14) is set to 0.2 for all recursive level L. In the recursive level $L = 1$, the starting time $t_s[L]$ of the time range for selecting the source particles from among the condensation particles is set to the time at which condensation particles begin to appear. The ending time $t_e[L]$ is set to $100\Delta t$ after $t_s[L]$, where Δt is a time increment of the stochastic simulation. In the recursive level $L \geq 2$, the starting time $t_s[L]$ of the time range for selecting the source particles from among the cloud particles in recursive level L - 1 is set to the time $t_s[L] = t_s[L\text{-}1] + \chi\Delta t$, where χ is a constant that is set to 50. The ending time $t_e[L]$ is set to $60\Delta t$ after $t_s[L]$.

Figure 7 shows simulation results of a more complicated large-scale cumulus cloud. In this simulation, two different 2D $1/f^{\beta}$ density maps are used for comparing the influence of the density maps on the growing cloud shapes. The magenta and blue maps in Figure 7.A (a) and B (a) are the density maps that are generated when the value of β is 3.3 and 2.7, respectively. The magenta regions indicate high-density regions, and the blue regions indicate low-density regions. The resolution of the density map of Figure 7.A is 150×100, while that of Figure

7.B, is 200×100. The images (a) to (e) in Figure 7.A and (a) to (e) in Figure 7.B are animation frames that are arranged over time. In both Figures 7.A and B, the clouds tend to grow larger on the higher density regions on each density map: the clouds grow larger on both end regions of the map in Figure 7.A, and on the central region in Figure 7.B. Because more water vapor particles are generated on higher-density regions, more source particles tend to be selected in those regions after condensation. This results in mapping more cloud primitive sets and generating more cloud particles. Afterward, the recursive generation mechanism of the cloud particles in our method continues to generate more cloud particles in those regions. Consequently, the generated clouds form a shape based on the density map. From the result of the simulation, we can observe that clouds are developed realistically with continuous growth effects. The maximum recursive level L_{max} and the selection ratio $M[L]$ are the same in Figure 7.a and b with values 4 and 0.3 respectively. The time ranges are the same as in Figure 6.

Figure 8 shows the simulation result of a cumulonimbus cloud. In this simulation, the 2D $1/f^{\beta}$ density map has the resolution of 300×100, and the value of β is 3.0. The maximum recursive level L_{max} is restricted to 4. The selection ratio $M[L]$ is set to 0.2 in recursive level $L = 1$, and 0.5 in $L \geq 2$. In recursive level $L = 1$, the starting time $t_s[L]$ is set to the time at which condensation particles begin to appear the same as in Figure 6. The ending time $t_e[L]$ is set to $60\Delta t$ after $t_s[L]$. In recursive level $L \geq 2$, the parameter χ for the starting time $t_s[L] = t_s[L\text{-}1] + \chi\Delta t$ is set to 60, and the ending time $t_e[L]$ is set differently for each recursive level within $(40\sim60)\Delta t$ after $t_s[L]$.

Figure 9 shows the simulation result of sparse cumulus clouds. The clouds were rendered and viewed from a position below the clouds, while those of the other figures were observed from the top or side positions. The appearance of cumulus clouds greatly depends on the density distribution and thickness of the clouds, as well as on the viewing direction in relation to the sun's position. If the generated clouds are thick enough, the sunlight does not completely traverse the cloud, resulting in heavy-looking cloud with a flat base. If the generated clouds are distributed with low density by sparse cloud particles, some of the sunlight can traverse the cloud, resulting in an irregular rounded base with different brightness. In the simulation of Figure 9, we intended to generate clouds with low density by sparse cloud particles, while the other figures are generated with dense cloud particles. In Figure 9, we generated the clouds from a density map of resolution of 300×300. To generate less particles than the other figures, the selection ratio $M[L]$ was set to 0.1. From the simulation result, we can observe that the base of the clouds are less flat, thus looking more realistic.

6.2. Performance analysis

The simulation statistics of the examples in Section 6.1 are shown in Table 1. These experiments were carried out on a Pentium® 4 3.8-GHz CPU computer with 2 GB of memory, and a QuadroFX1400 video card with 128 MB of memory.

In Table 1, the statistics data of Figure 2 shows the performance of the particle-based physical fluid simulation method to produce a 3D cloud primitive set, while those of the other figures show the performance of the proposed hybrid method. The hybrid simulation of the figures, except Figure 2, used the 3D cloud primitive set of Figure 2, and their total simulation time in Table 1 does not include the computation time for preparing the 3D cloud primitive set.

To create an animation of realistic cloud formation with continuous growth effects, a simulation method that calculates the motions of all cloud particles in the whole simulation space together by a physical fluid simulation such as SPH method [13] or Fedkiw method [5, 26] is desirable. However, such a method is still computationally expensive. To reduce the computational cost, an accelerated algorithm for searching the neighbor particles has been presented such as Takeshita [28], and this algorithm is applied in the particle-based fluid simulation method used to obtain 3D cloud primitive set for the proposed hybrid method in this paper. In this algorithm, the simulation space is divided into voxels and each particle is registered into the voxel in which the particle exists. Then, for each target particle, neighbor particles of the target particle are searched only within the voxels around the target particle. Therefore, this algorithm make the simulation efficient and the computational cost can be reduced to $O(N)$ time complexity, where N is the number of particles.

However, even if neighboring particles are searched efficiently, the interaction of the target particle with all of the neighboring particles has to

be calculated. This calculation needs lots of time for approximately solving the fluid equations. When we extend the simulation to a large-scale, the number of interaction between particles increases according to the increase in the number of particles, thus, the computational time increases consequentially.

In contrast, our hybrid method achieves the efficient creation of a large-scale cloud animation by combining two simulation methods: a stochastic simulation method in a large scale and a physical fluid simulation method in a small scale. The stochastic method simulates the air current caused by thermal process in the whole simulation space, and this is executed in a small computation time since the interactions between particles are not taken into consideration. On the other hand, compared with the stochastic simulation, the physical fluid simulation that is used to obtain 3D cloud primitive set needs more computation time, especially for computing the interactions between particles as mentioned above. However, we pre-compute the particle-based fluid simulation with a small number of particles to obtain the 3D cloud primitive set. This pre-computation for the 3D cloud primitive set does not take much time because of a small number of particles. Therefore, in the proposed hybrid method, the calculation time is mainly consumed for mapping the particles from the 3D cloud primitive set. The 3D cloud primitive set has the position data of the particles of all the frames in the physical fluid simulation, and the size of this data is large. Therefore, we store this data on the hard disk to save the memory storage and we read it when we want. The computational time to read particle data of the 3D cloud primitive set from the disk and map it to generate cloud particles once is assumed to be T, and the number of the recursive mapping execution is assumed to be F. Thus the total computation time to consume for the recursive mapping of the 3D cloud primitive set can estimate as $O(T \times F)$. This computation time is so fast compared with the physical fluid simulation. Table 1 shows that, approximately, our method spends roughly the same computation time as the physical fluid simulation of Figure2, although our method uses roughly one hundred times as many particles as Figure 2.

To run a simulation using the particle-based physical fluid simulation method with a large number of particles, a large memory storage size is required to allocate the voxels and to store the associated attributes of the particles. Each voxel stores the index of a particle, and the index needs 4 bytes. When the voxel resolution is $M = M_x \times M_y \times M_z$, the M voxels need $4 \times M$ bytes. The attributes of a particle include its density, temperature, position (P_x, P_y, P_z), velocity (V_x, V_y, V_z) and acceleration (f_x, f_y, f_z), and each attribute value needs 8 bytes as double precision floating point number. Thus, we need 11 doubles, that is, 88 bytes to store the attributes of one particle. This means that we need $88 \times N$ bytes to store N particles. Thus, to generate N particles using the physical fluid simulation, $88 \times N + 4 \times M$ *bytes* are required.

In our method, the memory storage is mainly used to store the positions of the cloud particles, which are mapped from the 3D cloud primitive set. The position data of a cloud particle needs 3 doubles, that is, 24 bytes. Thus, to store the positions of N_c cloud particles, $24 \times N_c$ bytes are required. In addition, the stochastic simulation method needs some memory storage. Each water vapor particle, or condensation particle, has the same attributes as the particle in the physical fluid simulation above and needs 11 doubles, that is, 88 bytes. Thus, $88 \times N_v$ bytes are required for N_v water vapor particles or condensation particles. However, N_v is much smaller than N_c. So, our method approximately requires $24 \times N_c$ bytes. On the other hand, the physical fluid simulation to obtain the 3D cloud primitive set requires $88 \times N_f + 4 \times M_f$ bytes, where N_f and M_f are the number of fluid particles and that of voxels in the physical fluid simulation respectively. This physical fluid simulation is pre-computed and is not executed simultaneously with the stochastic simulation and mapping process. Usually, the number of cloud particles, N_c, is much larger than N_f and M_f. Thus, consequently, our proposed method requires $24 \times N_c$ bytes. Compared with $88 \times N + 4 \times M$ bytes required for simulating all N particles by the physical fluid simulation, our method can save $64 \times N + 4 \times M$ bytes.

As shown in Figure 1, 5, 6, 7, and 8, the large-scale animations produced by our hybrid method resulted in realistic cloud growth motions. We can say that our hybrid method can efficiently produce realistic cloud growth animations in large scale and reasonable computation time and memory consumption.

Table1. Simulation statistics

Simulation	No. of Cloud Particles	Total Simulation Time(sec)	Total No. of Steps
Figure 1	1592646	359.0	488

Figure 2 (3D Cloud primitive set)	16401	458.2	154
Figure 5	4534381	559.0	612
Figure 6	492646	159.6	491
Figure 7.A	1405644	354.0	500
Figure 7.B	1707802	380.8	503
Figure 8	1973087	423.9	500

6.3. Rendering method

From the generated cloud particles, we create volume data with alpha values. Then, the volume data is loaded into texture memory as a 3D texture, and the final rendered images are created by a volume rendering method using OpenGL [1]. Moreover, the rendering method simulates primary scattering through the medium in a single direction. This direction is usually the direction leading to the point of view. The rendering method is executed in two steps. First, the voxels of the volume are sorted according to their distances from the viewpoint, and the voxels are cast to the background by alpha blending ordered by the farthest voxels from the viewpoint. In the second step, the rendering of the clouds viewed from the viewpoint is created.

7. Conclusion

In this paper, we have proposed a method for efficiently animating a large-scale cloud growth. Our proposed method employs a hybrid approach that combines a stochastic simulation method and a physical fluid simulation method. The recursive mapping of the small-scale physical fluid simulation result into the stochastic simulation efficiently produces large-scale vertically developing clouds with continuous growing. In this method, whether we can obtain the 3D cloud primitive set with detailed behavior or not is a difficult task and it directly influences the overall cloud growth behavior. We want to find a way to obtain a more efficient 3D cloud primitive set to control the cloud behavior. In the real world, the formation and growth of clouds are greatly affected by atmospheric conditions, hence producing different types of clouds. In the future, we plan to extend this research to generate other types of clouds, such as stratus-cumulus clouds, by including the effect of atmosphere conditions, such as wind. Moreover, at the present, the base of the generated clouds seems roughly flat. We plan to improve the proposed method in order to generate more realistic clouds with irregular bases. We also plan to apply our large-scale animation method to other natural phenomena such as smoke, fire, explosion and volcano ashes, since they also have behaviors with continuous growth effects. Because the proposed method generates particles in a recursive structure, it is considered that the LOD (Level of Detail) method for rendering can be easily applied. For fast rendering, a GPU particle-based rendering method is preferable. We plan to attempt these approaches in future studies.

Acknowledgement: This work was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B) 19300022.

References

[1] N. Chiba, K. Muraoka, A. Doi, and J. Hosokawa, “Rendering of Forest Scenery Using 3D Texture,” *The Journal of Visualization and Computer Animation*, Vol. 8, pp. 191-199, 1997.

[2] N. Chiba, K. Muraoka, H. Takahashi and M. Miura, “Two-dimensional Visual Simulation of Flames, Smoke and the Spread of Fire,” *The Journal of Visualization and Computer Animation*, Vol. 5, pp. 37-53, 1994.

[3] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita, “A simple, efficient method for realistic animation of clouds,” *Proc. of SIGGRAPH* 2000, pp. 19–28.

[4] D. Ebert, F. Musgrave, D. Peachey, K. Perlin, and S. Worley, *Texturing & Modeling: A Procedural Approach*, Morgan Kaufmann, 3rd Edition, 2002.

[5] R. Fedkiw, J. Stam and H. W. Jensen, “Visual simulation of smoke,” *Proc. of SIGGRAPH* 2001, pp. 15-22.

[6] G. Gardner, “Visual Simulation of Clouds,” *Proc. of SIGGRAPH* 1985, pp. 297-304.

[7] J. Kajiya and B. V. Herzen, “Ray tracing volume densities,” *Proc. of SIGGRAPH* 1984, pp. 165–174.

[8] K. Kaneko, “Simulating physics with coupled map lattices-pattern dynamics, information flow, and thermodynamics of spatiotemporal chaos”, *World Scientific*, Singapore, Vol. 1, 1990.

[9] T. Kikuchi, K. Muraoka, and N. Chiba, “Visual Simulation of Cumulus Type Clouds,” *The Journal of The Institute of Image Electronics Engineers of Japan*, Vol. 28, No. 2, pp. 140-151, 1999, in Japanese.

[10] T. Kikuchi, K. Muraoka, and N. Chiba, "Visual Simulation of Cumulonimbus Clouds," *The Journal of The Institute of Image Electronics Engineers of Japan*, Vol. 27, No. 4, pp. 317-326, 1998, in Japanese.

[11] S. Koshizuka and Y. Oka, "Moving-particle Semi-implicit Method for Fragmentation of Incompressible Fluid," *Nuclear Science Engineering*, Vol. 123, pp. 421-434, 1996.

[12] R. Miyazaki, S. Yoshida, Y. Dobashi, and T. Nishita, "A method for modeling clouds based on atmospheric fluid dynamics," *Proc. of Pacific Graphics* 2001, pp. 363-372.

[13] M. Muller, D. Charypar and M. Gross, "Particle-based fluid simulation for interactive applications ," *Proc. of SIGGRAPH/Eurographics symposium on Computer animation* 2003, pp. 154-159.

[14] J. J. Monaghan, "Smoothed Particle Hydro-dynamics," *Annual Review of Astronomy and Astrophysics*, Vol. 30, pp. 543-574, 1992.

[15] M. J. Harris and A. Lastra, "Real-time cloud rendering," *Proc. of Eurographics* 2001, pp. 76-84.

[16] M. J. Harris, W. V. Baxter, T. Scheuermann and A. Lastra, "Simulation of cloud dynamics on graphics hardware," *Proc. of Eurographics on Graphics Hardware* 2003, pp. 92-101.

[17] M. J. Harris, "Real-time Cloud Simulation and Rendering," PhD thesis, University of North Carolina, 2003.

[18] K. Nagel and E. Raschke, "Self-organizing Criticality in Cloud Formation," *Physica A*, Vol. 182, pp. 519–531, 1992.

[19] F. Neyret, "Qualitative simulation of convective cloud formation and evolution," *Proc. of Eurographics Computer Animation and Simulation Workshop* 1997, pp. 113-124.

[20] K. Perlin, "An image synthesizer," *Proc. of SIGGRAPH* 1985, pp. 287-296.

[21] H. O. Peitgen and D. Saupe, (Eds), *The Science of Fractal Image*, Springer-Verlag, 1988.

[22] W. Qiang, B. J. Jun, C. Chun, T. Fujimoto, and N. Chiba, "Surface reconstruction for animation of ocean waves," *Proc. of CAD/Graphics* 2005, pp. 477-482.

[23] N. Rasmussen, D. Q. Nguyen, W. Geiger, and R. Fedkiw, "Smoke simulation for large-scale phenomena," *Proc. of SIGGRAPH* 2003, pp. 703-707.

[24] J. Stam, "Stable fluids," *Proc. of SIGGRAPH* 1999, pp. 121-128.

[25] B. Stevens, "Cloud Transitions and Decoupling in Shear-free Stratocumulus-topped Boundary Layers," *Geophysical Research Letters*, Vol. 27, No. 16, pp. 2557-2560, 2000.

[26] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," *Proc. of SIGGRAPH* 2005, pp. 910-914.

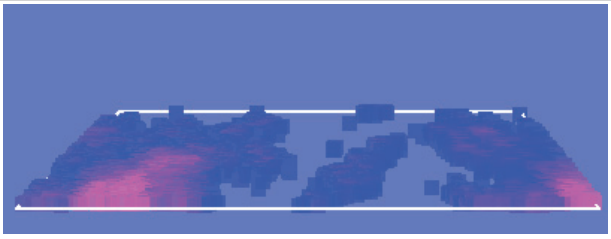
[27] D. Takeshita, T. Fujimoto, and N. Chiba, "Recursive particle generator for animating plume fluid," *International Workshop on Advanced Image Technology* 2005, pp. 487-492.

[28] D. Takeshita, S. Ota, M. Tamura, T. Fujimoto, K. Muraoka, and N. Chiba, "Particle-based visual simulation of explosive flames," *Proc. of Pacific Graphics* 2003, pp. 482-486.

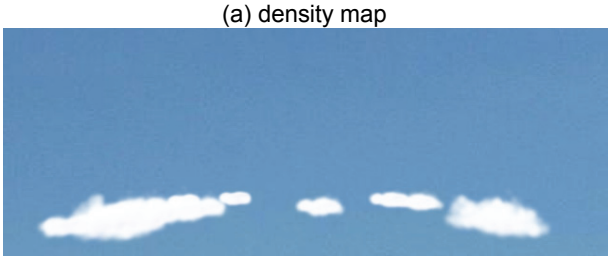
[29] R. Voss, "Random Fractals: Self-affinity in Noise, Music, Mountains, and Clouds," *Physica D*, Vol. 38, pp. 362-371, 1989.

[30] NASA Education, "The importance of understanding clouds," http://icp.giss.nasa.gov/education/cloudintro/

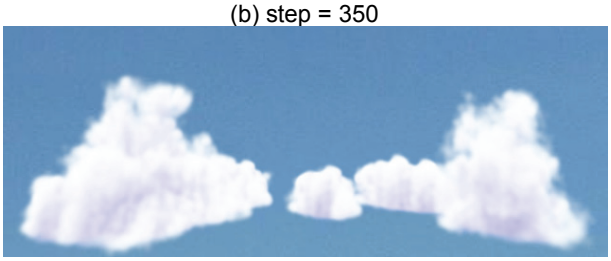
[31] WW2010, "Clouds and Precipitation," http://ww2010.atmos.uiuc.edu/(Gh)/guides/mtr/cld/home.rxml



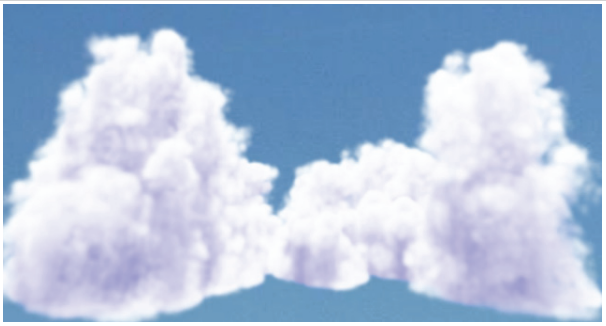
(a) density map



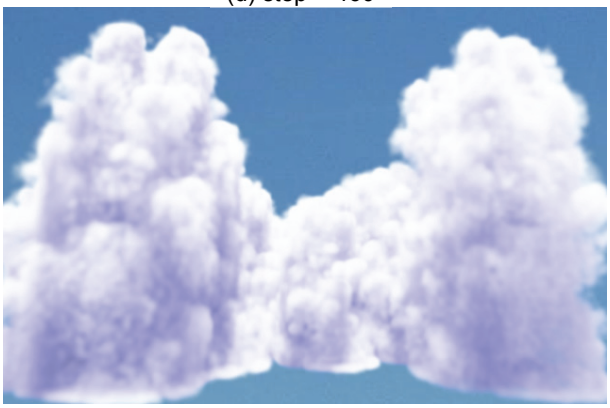
(b) step = 350



(c) step = 410



(d) step = 460



(f) step = 500

(A) Example 1: The resolution of the density map is 150×100.

(a) density map

(b) step = 350

(c) step = 410

(d) step = 450

(f) step = 503

(B) Example 2: The resolution of the density map is 200×100.

Figure 7. Examples of large-scale cumulus clouds

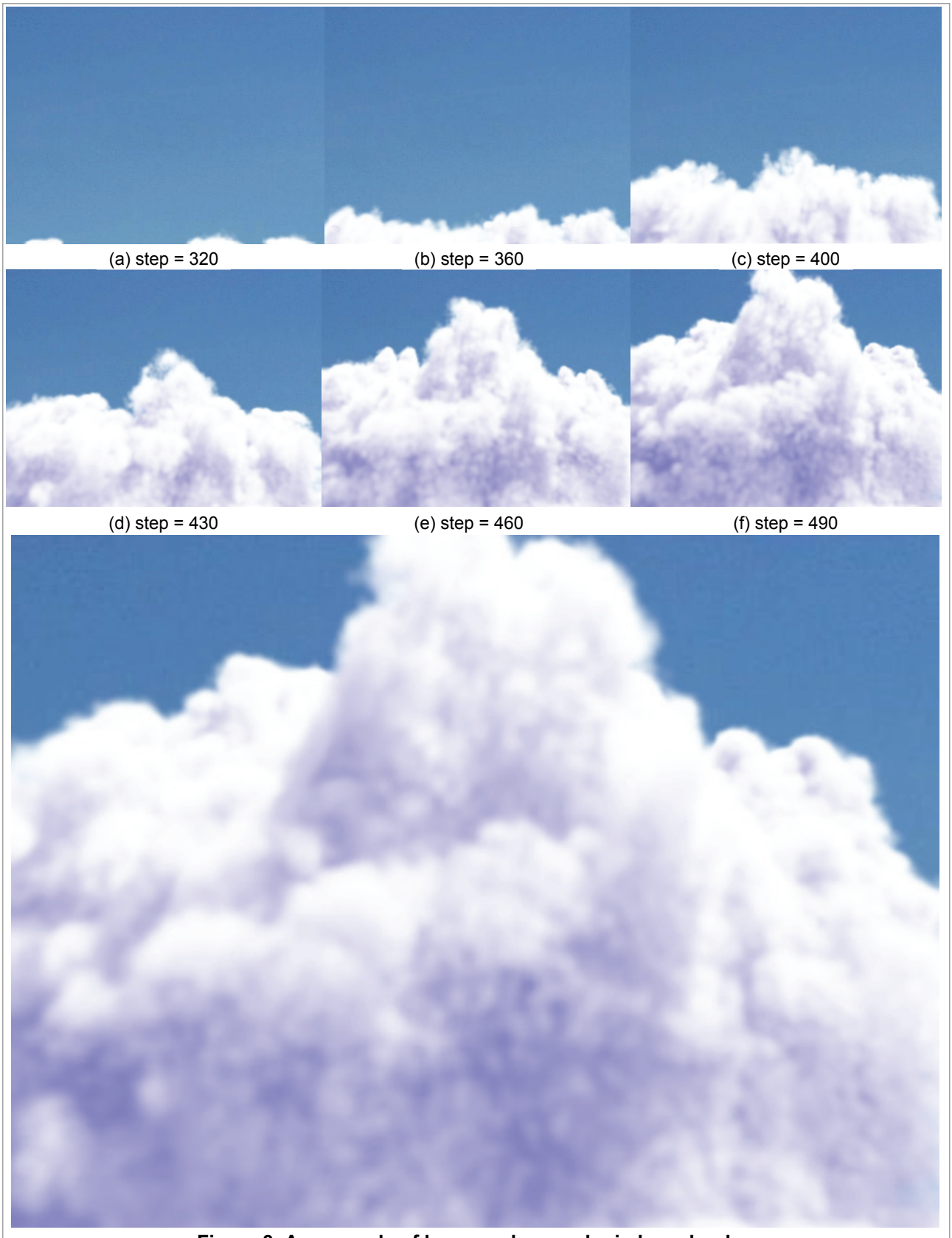


Figure 8. An example of large-scale cumulonimbus clouds

Figure9. An example of sparse cumulus clouds

Authors' biographies

MAMAT ABDUKADIR is currently a Ph.D. candidate in computer science at Iwate University. His research interests include computer graphics and physics-based animation. He received a BE in electrical engineering from Xinjiang University, China in 1999 and an ME in computer science from Iwate University in 2004.

TADAHIRO FUJIMOTO is currently an associate professor in the Department of Computer and Information Sciences at Iwate University. His research interests include computer graphics, geometric model, fractal theory, and mathematics for shape description in general. He received a BE in electrical engineering, and an ME and Ph.D. in computer science from Keio University in 1990, 1992, and 2000, respectively. He worked at Mitsubishi Research Institute from 1992 to 1995. He was a research associate in the Department of Computer and Information Sciences at Iwate University from 1999 to 2002, and a lecturer from 2002 to 2005. He is a member of SAS Japan, IEICE Japan, IPS of Japan, IEEE and ACM.

MAMTIMIN GENI is currently a professor in the Department of Mechanical Engineering, Xinjiang University. His research interests include computational mechanics, computational science and environmental simulation. He received the B.E in mechanical engineering from Xinjiang University, China and Ph.D. in engineering from Science University of Tokyo, Japan. He was a JSPS Post Doctoral Fellowship at Science University of Tokyo from 1997 to 1999. He is Visiting Associate Professor in the Department of Aircraft Engineering Northwestern Polytechnicl University, China.

NORISHIGE CHIBA is currently a professor in the Department of Computer and Information Sciences at Iwate University. His research interests include computer graphics, algorithm theory and science on form. He received a BE in electrical engineering from Iwate University and an ME and DE in information engineering from Tohoku University in 1975, 1981 and 1984, respectively. He worked at Nippon Business Consultant Co., Ltd. from 1975 to 1978. He was a research associate in the Department of Communication Engineering at Tohoku University from 1984 to 1986, an associate professor of computer science at Sendai National College of Technology from 1986 to 1987 and an associate professor of the Department of Computer and Information Sciences at Iwate University from 1987 to 1991. He is a member of SAS Japan, IEICE Japan, IPS of Japan, IEEE and ACM.