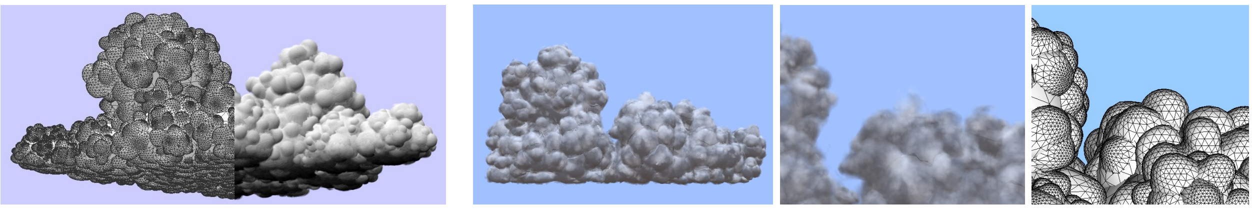
Modeling clouds shape

Antoine Bouthors[†] Fabrice Neyret[†]

EVASION-GRAVIR/ IMAG-INRIA
http://www-imagis.imag.fr/Publications/2004/BN04/



Abstract

We propose a model for representing the shape of cumulus clouds. We draw on several approaches that we combine and extend: We store a hierarchy of quasi-spherical particles (or blobs) living on top of each other. The shape of these particles is defined by an implicit field which deforms under the influence of neighbor particles. We define a set of shaders to simulate a volumetric appearance in the spirit of Gardner's textured ellipsoids*, to make the shape* appear *continuous, and to account for dedicated shading effects.*

In this paper we deal only with the definition of the shape. However, we believe that this model is well suited to be integrated with particle animation and advanced rendering.

Keywords: *natural phenomena, clouds, particles, implicit surfaces, shaders, procedural modeling.*

1. Introduction and Previous Work

Clouds are an important element of natural scene in quality rendering as well as real-time applications (e.g. flight simulators). Billowing clouds (and smoke) are complex matter in many ways: the shape has a fractal nature, so does the animation, and the rendering should account for local variations as well as global volumetric illumination. A wide variety of clouds exists, requiring different approaches. In this paper we only deal with *cumulus clouds*, which are well-formed and very contrasted clouds showing a quasi-surface.

Three ways have been followed in the representation of cloud shapes: Volumetric clouds (either explicit [NND96] or procedural [Ebe97, SSEH03]), billboards [HL01, DKY*00], and surfaces [Gar84, Gar85]. (Note that the dense billboarding of [DKY*00] may be considered as a volume discretization). Volume approaches are still not amenable to real-time rendering of clouds: volume rendering techniques are getting very fast, but the resolution required by a cloud sky – which is both large and very detailed – will long be too much for graphics hardwares. Billboards and impostors are the current solution used in games [HL01, Mic], but still a small number of slices must be used to keep an interactive pace (the problem lies in the huge number of pixels drawn rather than in the number of polygons). Surfaces permit very efficient rendering, but they look too crude to represent a volumetric shape. However, Gardner [Gar84, Gar85] was able to make ellipsoid surfaces look volumetric in the scope of ray-tracing thanks to view-dependent transparency shaders evaluated at each ray. Fortunately, recent hardware allows to define *pixels shaders*, making this approach amenable to real-time rendering. An early attempt in that direction was done in [ES00].

Concerning the specification of clouds shape, three approaches have been followed: using simulation data as input (or some extrapolation of acquired data such as satellite views), relying on procedural fractal noise [Per85] and a shaping function such as implicit surfaces like in [Ebe97, SSEH03], or relying on particles (manually or procedural placed). In particular, [NND96] combined hierarchical particles and implicit surfaces: a level is built upon the previous one by tessellating the implicit surface defined by the last layer of particles and placing next level blobs evenly on this surface.

† [Antoine.Bouthors | Fabrice.Neyret]@imag.fr

Considering that cloud shape is simply the result of the air movement, other possible approaches consist in simulating the fluid dynamics [FSJ01, HBSL03] or an animated system of particles [Ney03].

Our model is based on particles (or blobs) associated to an implicit field which allows to define a surface (to be rendered or to sustain other particles). This primitive is associated to a shader in order to make it appear volumetric. Our hierarchical model relies on subdivision and repulsion to populate each level evenly. The implicit field of particles is responsible for deforming them so to avoid particle intersections in the spirit of [Gas93].

Figure 1: A real cloud.

2. Our representation

A cloud is composed of a set of hierarchical levels recursively defined on top of each other. Each level l consists of a set of particles p_i and defines a surface S_l. Each particle of level l is defined by local and global parameters: its location $\mathbf{P}_i$ on the surface S_{l-1}, its radius r_i, its flattening e_i, and blending parameters defined globally for each level. Some of these parameters are automatically adjusted depending on the level or on the surrounding particles whereas the others can be directly or indirectly tweaked by the user.

Each particle defines its own surface S_i, which consists of a 'pure' shape (a flattened sphere) altered to fit the surrounding particles (taking the particle's blending parameters into account). This is done using an implicit formulation inspired by [Gas93]. This implicit surface is defined by:

$$S_i = \{\mathbf{P} \in \mathbf{R}^3 / f_i(\mathbf{P}) = 1\}$$

where $f_i(\mathbf{P}) = g_i(\mathbf{P}) + h_i(\mathbf{P})$

$g_i(\mathbf{P})$ is a basic flattened spherical field function which yields the 'pure' shape:

$$g_i(\mathbf{P}) = \exp\left(1 - \frac{\mathrm{d}_i}{r_i(1 - e_i \mathrm{d}_{l-1})}\right)$$

with d_i and d_l the distance from $\mathbf{P}$ to $\mathbf{P}_i$ and S_l, respectively. The flattening (set randomly) accounts for the various stages of development of cloud bubbles and provides a less uniform aspect of the cloud surface. $h_i(\mathbf{P})$ is a field function which alters this spherical shape to meet our fitting constraints, as in [Gas93]. Their purpose was to define exact contact between two shapes. Contrary to them, we do not want blobs to stick to each other (real blobs on clouds do not, see Figure 1): we want *furrows* between blobs, which should nevertheless be continuous. We define $h_i(\mathbf{P})$ as a combination of several field functions controlled by the particle parameters: $h_i(\mathbf{P}) = m_i(\mathbf{P}) + n_i(\mathbf{P}) + o_i(\mathbf{P})$

• $m_i(\mathbf{P})$ is a field function that prevents the blobs from overlapping, and is thus in its spirit very close to that of [Gas93]. However, we added an offset parameter ε to repulse the surfaces more than for contact in order to create furrows between the blobs. Also, we used a different equation for the bulge generation to produce a smoother transition between the repulsion area and the pure shape far from the interaction area. This gives $m_i(\mathbf{P}) = \sum_j m_i^j(\mathbf{P})$

$$\text{with} \quad m_i^j(\mathbf{P}) = (1 - \varepsilon - g_j(\mathbf{P}))\min(1, g_j^2(\mathbf{P}))$$

• $n_i(\mathbf{P})$ is a field function that enlarges the blob surface as it is closer to the base surface (*i.e.,* the previous level surface S_{l-1}) so that the particle does not appear as "popping" out of the cloud. $n_i(\mathbf{P})$ is controlled by two parameters : b, that defines the amount of blending, and I, that adjusts the fraction of the bubble that will be enlarged to create the blending. We choose:

$$n_i(\mathbf{P}) = b\ \min\left(1, e^{-\frac{Id_{l-1}}{r_i}}\right) e^{1 - \frac{d_i}{r_i}}$$

• $o_i(\mathbf{P})$ is a field function responsible for the flat cloud bottom, by restricting the blob surfaces above a given absolute height h_0 with a stiffness (*i.e.,* repulsion force) α_h.

$$o_i(\mathbf{P}) = g_i(\mathbf{P})\min\left(0, \frac{\mathrm{height}(\mathbf{P}) - h_0}{\alpha_h}\right)$$

Finally, S_l is defined from the potential $f_l(\mathbf{P}) = \max\left(f_{l-1}(\mathbf{P}), \max_i f_i(\mathbf{P})\right)$

3. Making a cloud shape

We need the surface S_{l-1} in order to place the particles of the next level, and to render the last level of the hierarchy. We can obtain the discretized surfaces of all the blobs belonging to one hierarchy level $l-1$ by using the method described in [Gas93]. To place the particles p_i of level l, we preferred extracting an isosurface of $l-1$ using a particle system as described in [WH94] and [CA97], instead of discretizing explicitly the surface S_{l-1} as done in [NND96]. This particle system is then used directly as the particle set of level l, using the repulsion radius of the particles as the blob radius.

This approach simulates particles that repulse each other and grow to populate all the available surface. When a particle has grown too much or when an area is not crowded enough, the concerned particles split to create new particles. On the other hand, if a part of the surface is overcrowded some particles are removed. A target inter-particle distance is provided (*i.e.,* twice the target particle radius), that all particles will have after reaching the equilibrium state.

We want to reproduce a natural distribution, so regular distributions are not desired. We take different radius of influence for each particle, so that all the particles of a level do not have the same aspect and the layer look uneven. To take in account this new independent radius of influence, we modified the algorithm [WH94] as follows : if the energy of a particle is too low, the repulsion radius of the particle evolves so as to reach the desired energy, as in [WH94]. Otherwise, it evolves so as to reach the desired radius. In addition, the desired energy is also different for each particle, as it is modulated as much as the desired radius.

Once the particle set of level l is created on the implicit surface S_{l-1}, the blobs surfaces are defined as described in section 2. Thus we can generate the implicit surface S_l from these blobs that we will use to place the particles of level $l+1$, and so on. The last one defines the cloud shape.

Particles of level 0 (*i.e.,* root) have no implicit surface to rely on and have to be created another way. Depending on the application purpose, this can be done procedurally (*e.g.,* to create a cloud sky) or this can be controlled by the user. In the first case we create a small number of particles randomly positioned on a horizontal plane or in a small subspace. In the second case, an interface lets the user shape and place them by hand, allowing him to create realistic features such as cloud turrets.

4. Rendering a cloud shape

The scope of this paper is the shape, not the rendering. However, local details are not always best represented by a geometric surface, especially if they are complex or fuzzy. Kajiya suggests in [Kaj85] to switch from geometry to texture to shaders with distance (or size). This is precisely what the Gardner's *textured ellipsoid* model [Gar85] does. We define the local aspect of the cloud using textures and shaders in the same spirit.

First, we simulate blobs at lower scales relying on a texture mapped on the blobs surface (we used a classical Perlin solid texture).

Then we define a Gardner-like shader simulating a fuzzy layer of material by increasing the transparency t near the silhouette (it is thus a view-dependent shader). The silhouette proximity is detected using $\vec{N}.\vec{C}$ which values 0 on the silhouette, where $\vec{N}$ is the local normal (*i.e.,* the field gradient) and $\vec{C}$ is the camera direction. Then we can set the transparency t as $(1-\alpha\vec{N}.\vec{C})^{\alpha_s}$, where α and α_s control the thickness and the sharpness of the transparent margin, respectively.

Similarly, continuity between blobs can be eased using a "normal blending" shader: assuming the final mesh corresponds to level l, we store at each mesh vertex the potential p and the normalized gradient $\vec{g}$ of S_{l-1}. This provides us with a normalized distance $d = 1-p$ to the base surface of the blobs and a normal of this base surface. Thus we can interpolate the normals close to the base surface so that the furrows looks continuous even if the blob surfaces do not connect perfectly smooth: $\vec{N}' = (1-p^{\alpha_N})\vec{N} + p^{\alpha_N}\vec{g}$ where α_N controls the size of the smoothing area.

5. Results

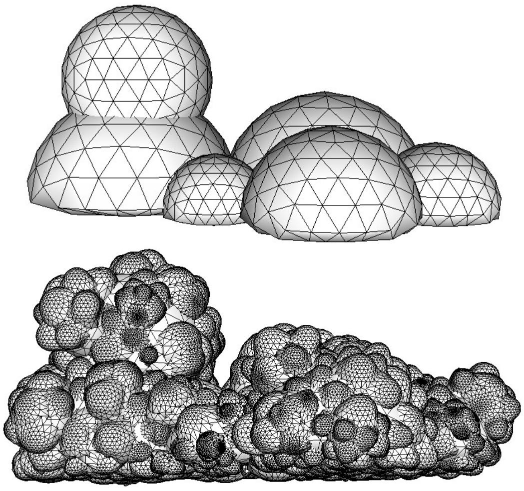


Figure 2: *Top:* 6 user-defined blobs. *Bottom:* with 2 levels generated.

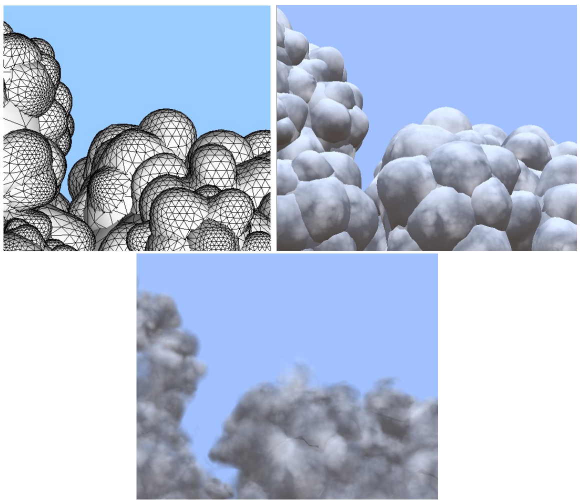


Figure 3: *Top left:* a zoom on the mesh of the generated blobs. *Top right:* with the Perlin texture. *Bottom:* with the Gardner-like shader.

The pictures presented in Figure 2,3 and the teaser show a cloud with a root level made of six hand-placed particles. It is a very small work for the user, but this controls the overall shape of the cloud. Figure 2.top shows this root level. We generated on top of it two extra levels resulting in the picture shown in Figure 2.bottom. The effect of the shaders is shown on Figure 3 and on the teaser.

Parameters have been chosen as follow. The radius of the hand-placed root particles (*i.e.,* level 0) is comprised between 200m and 450m. The radius of the particles for level l is one third of the radius of particles of level $l-1$ (thus mimicking the lacunarity estimated on Figure 1), modulated by a random factor of $\pm$ 30%. The particles were flattened with a factor e_i randomly chosen between 0.1 and 1. The parameter controlling the furrow size ε was set to 0.5, meaning that each blobs is separated at least half of its radius from the others.

We set the amount of blending b as null for levels 0 and 1, and 1.0 for level 2. The parameter I was set to 10.

The reference height h_0 corresponding to the floor of the cloud was 100m, which was approximately the height of the lowest particles of the root level. The stiffness of the repulsion α_h was 100m as well.

6. Conclusion and Future Work

In this paper, we proposed a model to define and build the *shape* of cumulus clouds, resulting in a surface mesh and shaders. Since we do not rely on a volumetric representation contrary to [NND96], the rendering (and therefore the animation) of our model is really fast. It is also more realistic, as the blobs are deformed to mimic the shape of real clouds. Moreover, we do not suffer from voxel discretization.

The realistic *rendering* of this shape taking into account the various global illumination effects is yet to be done. We believe that it is feasible to reproduce most of these effects using shaders and precomputations, which is our main goal as future work. Moreover our model is based on particles and thus should be animatable. We would be interested in studying how to adapt an animation model like [Ney03] to our case. Our long term goal is to obtain a plausible evolving cloud sky with plausible rendering at interactive rate.

References

[CA97] Crossno P., Angel E.: Isosurface extraction using particle systems. In *IEEE Visualization '97* (Nov. 1997), pp. 495–498. 2

[DKY*00] Dobashi Y., Kaneda K., Yamashita H., Okita T., Nishita T.: A simple, efficient method for realistic animation of clouds. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 19–28. 1

[Ebe97] Ebert D. S.: Volumetric procedural implicit functions: A cloud is born. In *SIGGRAPH 97 Technical Sketches Program* (Aug. 1997), Whitted T., (Ed.), ACM SIGGRAPH, Addison Wesley. ISBN 0-89791-896-7. 1

[ES00] Elinas P., Stürzlinger W.: Real-time rendering of 3D clouds. *Journal of Graphics Tools 5*, 4 (2000), 33–45. 1

[FSJ01] Fedkiw R., Stam J., Jensen H. W.: Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 15–22. 2

[Gar84] Gardner G. Y.: Simulation of natural scenes using textured quadric surfaces. In *Computer Graphics (SIGGRAPH '84 Proceedings)* (July 1984), Christiansen H., (Ed.), vol. 18, pp. 11–20. 1

[Gar85] Gardner G. Y.: Visual simulation of clouds. In *Computer Graphics (SIGGRAPH '85 Proceedings)* (July 1985), Barsky B. A., (Ed.), vol. 19, pp. 297–303. 1, 3

[Gas93] Gascuel M.-P.: An implicit formulation for precise contact modeling between flexible solids. In *Proceedings of SIGGRAPH 93* (Aug. 1993), pp. 313–320. 2

[HBSL03] Harris M. J., Baxter W. V., Scheuermann T., Lastra A.: Simulation of cloud dynamics on graphics hardware. In *Graphics Hardware 2003* (July 2003), pp. 92–101. 2

[HL01] Harris M. J., Lastra A.: Real-time cloud rendering. *Computer Graphics Forum 20*, 3 (2001), 76–84. 1

[Kaj85] Kajiya J. T.: Anisotropic reflection models. In *Computer Graphics (SIGGRAPH '85 Proceedings)* (July 1985), Barsky B. A., (Ed.), vol. 19(3), pp. 15–21. 3

[Mic] Microsoft: Microsoft flight simulator 2004. (also presented as a Siggraph Sketch 2003). `http://www.ofb.net/~eggplant/clouds/`. 1

[Ney03] Neyret F.: Advected textures. *Symposium on Computer Animation'03* (july 2003). 2, 4

[NND96] Nishita T., Nakamae E., Dobashi Y.: Display of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), Rushmeier H., (Ed.), ACM SIGGRAPH, Addison Wesley, pp. 379–386. 1, 2, 4

[Per85] Perlin K.: An image synthesizer. In *Computer Graphics (SIGGRAPH '85 Proceedings)* (July 1985), Barsky B. A., (Ed.), vol. 19(3), pp. 287–296. 1

[SSEH03] Schpok J., Simons J., Ebert D. S., Hansen C.: A real-time cloud modeling, rendering, and animation system. *Symposium on Computer Animation'03* (july 2003), 160–166. 1

[WH94] Witkin A. P., Heckbert P. S.: Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH 94* (July 1994), Computer Graphics Proceedings, Annual Conference Series, pp. 269–278. 2, 3