SURFACE-BASED EFFICIENT CLOUD VISUALISATION FOR ANIMATION APPLICATIONS

Andrzej Trembilski **Andreas Broßler**

Abt. Visualisierung und Virtuelle Realität
Fraunhofer Institut für Graphische Datenverarbeitung
Rundeturmstr. 6
64283 Darmstadt, Germany
Tel: (+49)(0)6151/155-287
Fax: (+49)(0)6151/155-196
trembils@igd.fhg.de
www.igd.fhg.de/~trembils/

ABSTRACT

For the local TV presentation of weather forecast data it is important to have high-quality and fast visualisation of clouds. In this paper we present surface-based methods for the high performance visualisation of clouds from data produced by a routine meteorological weather simulation. Isosurfaces, which are originally too coarse because of the data grid resolution are refined and deformed. The refined geometry is used for a light simulation and transparency computation.

Keywords: Meteorological visualisation, cloud modelling and visualisation, surface refinement.

1. INTRODUCTION

Numerical weather simulation models, like these of the German Weather Service (DWD), produce twice a day large quantities of simulation data, which must be visualised for the presentation on television.

Visualisation systems like TriVis are used to visualise the weather forecast in the overview for whole countries. Special weather phenomena, which occur only locally, would disturb and are not desired in such a general view. Therefore the realistic representation of clouds is not so important, because single clouds would appear too small in the global view anyway. Global cloud fields are visualised for the presentation of the general weather situation in a larger region.

The requirements change as soon as one tries to make a local weather presentation. They result from the combination of the techniques of Augmented Video and Scientific Visualisation. If weather visualisation should be integrated into a video film, the schematic, global representation of the clouds is not longer acceptable. On the one hand the spectator can compare the artificial clouds with the genuine world presented in the video. Any break in the representation diverts from the actual information and irritates the spectator. On the other hand only a realistic cloud representation yields understandable weather forecasts.

The actuality of the data in the presentation is an important restriction for the selection of the visualisation methods. For a film of only one minute 3000 images must be rendered and stored quickly enough to guarantee that the resulting TV presentation is still up to date. If the production of a frame takes only one second, 50 minutes are needed for the film, for practical applications the approximate computing time must not exceed 5 seconds.

Such rendering times can only be achieved with the help of 3D graphics hardware. The common way is to use OpenGL graphics accelerator hardware, which is available at low prices these days.

This paper presents methods to generate appropriate cloud visualisation from isosurfaces generated by the Marching Cubes algorithm.

2. PREVIOUS WORK

The concept of representing clouds as textured, semitransparent and stiff ellipsoids was proposed by Gardner [3]. Gardner did not use simulation data, but Knöpfle simplified his approach for the integration of weather forecast data in a virtual environment [6]. He achieved a real-time visualisation of cloud fields by using textured polygons for the top and bottom layer of the clouds. This method is also used in the TriVis TV weather presentation software [12]. Since the resolution of the layers is just as low as the resolution of the data, the resulting clouds appear very schematic.

In the volume rendering approach, the 3D world is subdivided into voxels and the local gas density is computed using spectral turbulence [13, 11, 2]. After these calculations are finished one has to render the scene with special ray tracers. The disadvantages are obvious: volume ray tracing is too slow and the amount of data for complex scenes is too large to handle. Similar volume grid based approaches were introduced in [5] and [7].

Neyret presented in [8] a method to produce clouds from overlaid spheres. We use a modified form of this method for the calculation of the surface of our clouds.

In [9] very complex, high-quality volume-based cloud lighting models are presented. [1] extends these results with even more complex volume rendering techniques based on the splatting algorithm using billboards. An own cloud simulation is developed to achieve a high quality animation. For our purpose these methods are still too slow (up to 30 seconds per 640x480 pixels frame) and the visualised data does not originate from a meteorological routine weather simulation. Due to the rendering algorithm the generated clouds appear extremely blurry, so the method cannot be used to generate all kinds of clouds, especially not the classical cumulus clouds.

Commercial applications like Softimage or Bryce seem to use some of the above methods. They can generate and render fine clouds, but these clouds still have nothing to do with weather simulation data.

Unbescheiden and Trembilski presented in [16] a particle-based method for the visualisation of smoke in Virtual Reality. It models the movement of a smoke cloud and guarantees that a moving observer gets a correct optical impression especially inside the smoke cloud, but it is also not suitable for the visualisation of any given simulation data.

Trembilski presented in [14] two different, prototypical methods for cloud generation from isosurfaces. Their main advantage was the change of the shape of the original isosurface geometry towards a realistic looking cloud. Still, the fractal algorithm's space complexity is not acceptable. The light model used here is too simple to be used in a high quality TV presentation.

3. THE PROBLEM AND OUR SOLUTION

Without any obstacles we can see a horizon with a radius of 35.7 km from a 100m tower. The visible piece of the sky is however still larger with a radius of 514 km. However, this is only the theoretically visible part of the atmosphere. The radius can be limited, assuming that even on best days the atmosphere is never quite clear and thus the diameter of the visible area will surely remain limited to 200 km (see [14]). Additionally, this is absolutely sufficient for the needs of the local weather forecast.

Using the model of the German weather service with the mesh size of 2.5 km for the local weather forecast, we get a grid of 80x80x35 data points, i.e. 224000 data points.

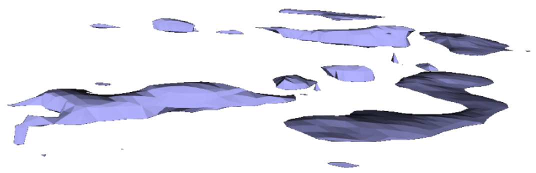

Fig. 1: Isosurfaces computed from the original data grid (200 km^2)

The Marching Cubes algorithm produces closed, coherent polygonal cloud geometries. An example of an isosurface calculated from such a data grid is presented in fig. 1. The geometry seems to be very coarse. It does not look like a cloud, parts of it are very sharp-edged, and other parts are very flat. The simple gouraud shaded rendering does not improve the result.

Therefore we developed some new postprocessing algorithms to use the isosurfaces for a realistic cloud visualisation.

To achieve this goal we pass through the following steps:

1. Refinement of the geometry
1.1 Production of the sub-triangles (fig. 2b)
1.2 Rounding of the sharp edges (fig. 2c)
1.3 Shift of the vertex positions (fig. 2d)
2. Colour calculation
2.1 Ray tracing (only for the triangle vertices)
2.2 Calculation of transparencies
3. Texturing.

These steps are described in the following sections.

GEOMETRIC MODEL

As our algorithms are mainly designed to construct cumulus clouds, the isosurfaces we used are constructed from data, which represents cumulus clouds. The data-pre-filtering step is done at first with help of the meteorological data base. To convert a given isosurface to a cloud, we have to refine it by partitioning the original triangles (fig. 2b) and a smooth approximation (fig. 2c) of the original geometry and then to deform the smoothed geometry (fig. 2d). We use a simple level-of-detail technique to reduce the amount of produced sub-triangles depending on the distance of the original triangle from the eye-point. To produce the typical cloud surface properties we then use Neyret's sphere approach.

The following sections describe these steps.

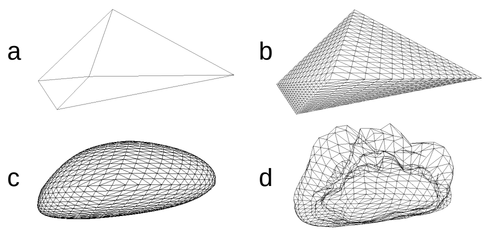

Fig. 2: Steps for the refinement and deformation of a rough cloud geometry: a: original geometry, b: partitioning of the triangles, c: smoothing, d: deformation.

Partitioning of the original triangles and the calculation of smooth cloud forms

Our approximation procedure calculates a smoothed geometry where the new vertices P_j depend of the base vertices S_i from the original geometry (see fig. 3).

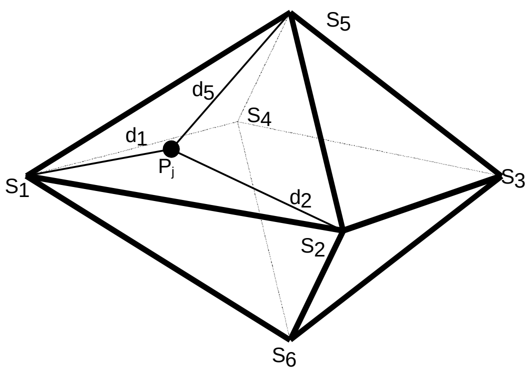

Fig 3: Computation of a new point P_j to smooth the rough cloud surface.

In order to control the number n of the base vertices S_i influencing the new vertices P_j, we construct a base function f(d):

$$f(d) = \begin{cases} \left(\frac{1}{D}d^2 - 2d + D\right)^2 & \text{for } d \in [0, \frac{D}{2}) \\ \left(-\frac{1}{D}d^2 + \frac{D}{2}\right)^2 & \text{for } d \in [\frac{D}{2}, D) \\ 0 & \text{for } d \in [D, \infty) \end{cases} \quad (1)$$

In (1) D defines the influence range of the old vertices. It is a user-defined, experimentally found constant, depending only on the coordinate range of the points S_i.

For the calculation of P_j the weight of a base vertex S_i depends on its distance d_j from S_j. We define this dependency using the base function f:

$$P_j = \frac{\sum_{i=0}^{n} f(d_i) S_i}{\sum_{i=0}^{n} f(d_i)} \quad (2)$$

See fig. 2c for an example of a smoothed cloud geometry.

The refinement can be repeated. We experimented with up to 6 iterations (starting with an octahedron).

The smooth geometry has now to be deformed, in order to give it the typical cloudy appearance.

Deformation of the refined surface

Similarly as suggested by Neyret in [8], we describe the surface of a cloud as a set of hemispheres, which are pushed into a plane (see fig 4).

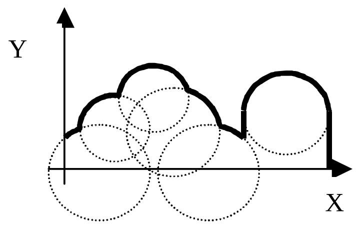

Fig. 4: Spheres shape the cloud surface.

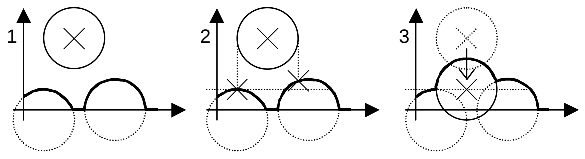

Fig 5: How to avoid flat parts while building a cloud shape

Fig. 5 shows our procedure schematically for the 1D case:

1. A sphere is set into the cloud structure at a (pseudo-) random position.
2. Examine all points of the outline of the projection of the sphere along the y-axis.
3. The midpoint of the sphere is shifted up to the lowest point found in 2 and the sphere is stamped into the structure.

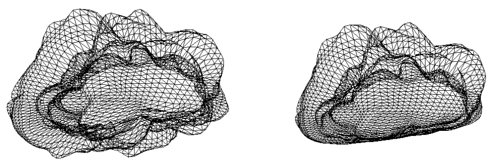

Fig. 6: Left: a cloud surfaces computed with a constant K = 0.1. Right: a cloud with a smoother lower surface after the reduction of K for low N_y values.

In order to form the cloud surface in a realistic way, each vertex P_i of the new smooth geometry is shifted by the value of $g(P_i)$ along its (normalised) normal $N = (N_x, N_y, N_z)$:

$$P_j' = P + N \cdot g(P_j) \cdot \left(\frac{(1-K)(N_y+1)}{2} + K \right) \quad (3)$$

where g(P) is a Fourier-based pseudorandom wave function, as introduced for clouds by Gardner in [3].

It is important to note that in the natural environment the characteristic cloud forms are especially distinct on the topside of the clouds. We achieve this effect simply by the last term of (3). K = const should be situated between 0 (for N_y = -1) and 1 (for N_y = 1). Thus with K = 1 the surface of the cloud is shifted equally in all directions, for K = 0 the lower surface stays smooth. We received realistic cloud forms for $K \approx 0.1$.

The result makes a quite natural impression (fig. 6). The calculation process is quite complex, it is however possible to store the values of function g in a table and interpolate the intermediate values to accelerate the computation.

LIGHTING MODEL

Our lighting model is not strictly a physical one. We tried to obtain best visual results with a minimum of computational cost – our aim was not the exact light simulation. Our model determines the colour of vertices on the cloud surface. For efficiency reasons this calculation is executed only for the vertices of the cloud geometry. The pixel colours within the triangles are interpolated by the graphics hardware.

For the lighting calculations the distances the light beams pass within clouds must be determined. In order not to test each ray against each triangle we organize the intersections of all light beams with their appropriate level in a k-d-point-quadtree. To find the rays cutting a triangle, the triangle must be projected (in lighting coordinates) on the plane. Then, all light beams cutting the bounding box of the triangle can be efficiently found in the k-d-tree. Only these rays are accurately tested whether they cut the triangle.

For the calculation of the colour of a cloud geometry vertex we determine the following light components:

1. Ambient light
2. Sunlight reflected depending on the direction
3. Sunlight reflected not depending on the direction
4. Mie-scattered sunlight
5. Sky light

Ambient light

If I_a is the ambient brightness and k_a the ambient reflection factor of clouds, then the brightness I of a cloud vertex is as always:

$$I = I_a k_a \tag{4}$$

k_a indicates the percentage of the ambient light, reflected by a cloud. This constant is approximately equal to the albedo of a cloud. Nishita recommends in [9] an albedo of approximately 0.7 for cumulus and 0.9 for stratus.

Direct sunlight

In contrast to the ambient intensity I_a, which is a constant for the entire scene, light rays crossing clouds should have a reduced intensity. We calculate the intensity I_P of the sunlight at the point P_n on the surface of a cloud generally as:

$$I_P(P_n) = \frac{I_{\max}}{1 + O_P \sum_i d_i} \tag{5}$$

with

$I_{\max}$	= maximum intensity of the resulting cloud surface, should depend on the current weather situation and sun position
$\sum_i d_i$	= sum of the distances, which the actual light ray travels within the clouds
O_P	= opacity factor of the clouds, a user-defined constant.

We split I_P into three parts, which differ in how the light rays are reflected or scattered at clouds:

1. I_d : sunlight reflected like ambient light (independent of the angle, direct intensity)
2. I_r : diffusely reflected sunlight, (depending on the angle between the light beam and the cloud surface, direction-controlled intensity)
3. I_m : strongly forward scattered sunlight (scattering on tiny water droplets, with the laws of the Mie scattering, Mie intensity).

For the reflection and scattering effects, which strongly depend on the direction of the arriving light, we introduce a different opacity of the clouds for the intensity parts I_d, I_r and I_m with $O_d \approx$ const., $O_r \approx 4\ O_d$ and $O_m \approx 8\ O_d$. Thus rays, which crossed larger cloud sections, can hardly cause such effects, as it is the case in the nature. Therefore we write:

$$I_d(P_n) = \frac{I_{\max}}{1 + O_d \sum d_i} \tag{6}$$

$$I_r(P_n) = \frac{I_{\max}}{1 + O_r \sum d_i} \tag{7}$$

$$I_m(P_n) = \frac{I_{\max}}{1 + O_m \sum d_i} \tag{8}$$

These intensities are successively weighted and inserted to our lighting model in the following sections.

Direct intensity

Direct intensity I_d is only weighted with a reflection factor k_d:

$$I = I_a k_a + I_d k_d \tag{9}$$

We assume $k_d \approx k_a$. Fig. 7 shows a cloud using backlight with a small (left) and with large O_d (right). For vertices which are lit up directly by the light source applies $I_d = I_{\max}$, i.e. all these vertices have the same brightness (fig. 7 left).

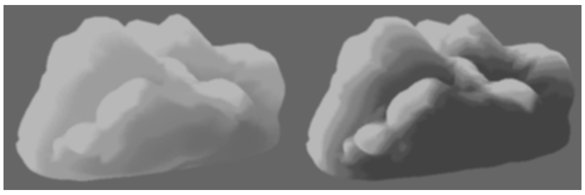

Fig 7: A strongly (left) and a weakly translucent cloud (on the right) in with direct lighting.

Direction-controlled intensity

The part of diffuse reflection decreases with the increasing angle ϕ between a light ray and the surface-normal. Thus (fig. 8) also directly illuminated cloud parts are optically resolved. Using this extension the new formula for the calculation of the intensity is

$$I = I_a k_a + I_d k_d + I_r k_r \cos\phi \tag{10}$$

A suitable value for k_r is $k_r \approx k_d/2$, as we found in experiments.

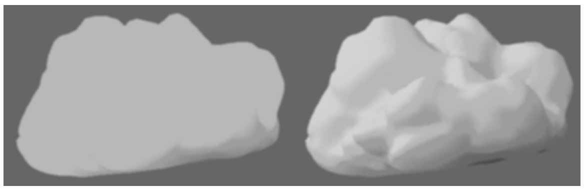

Fig 8: Clouds lit without (left) and with diffuse reflection.

Mie intensity

Mie scattering becomes clearly visible, if the sun is behind a cloud. In this case the thin edges of the cloud light up brightly in the backlight. This brightness depends strongly on the angle φ between the view ray and light ray and decreases quickly from a maximum (for $\varphi = 180°$) almost to 0. This behaviour can be achieved with a factor like $((-\cos\varphi + 1)/2)^n$. The exponent n is a measure for the angle-dependence. We found experimentally $n \approx 11$ as a suitable value.

Thus the new formula for the calculation of the brightness of a vertex is:

$$I = I_a k_a + I_d k_d + I_r k_r \cos\phi + I_m k_m \left(\frac{-\cos\varphi + 1}{2}\right)^n \tag{11}$$

As shown in fig. 9, the part of Mie scattering can be very bright in the thin parts of the cloud, where $\varphi \approx 180°$. For this reason we introduced a scattering coefficient k_m, which should be chosen as large as $k_m \approx 4\ k_d$.

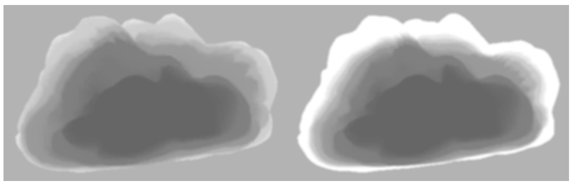

Fig. 9: Cloud visualisation without (left) and with the Mie scattering effect (right).

Sky as light source

During the day the sun is by far the most important source of light. However Nishita points out (in [9]) that in the morning and in the evening the light emitted by the sky hemisphere plays an increasing role. If the sun sank already so far behind the horizon that it does not illuminate the clouds directly any longer, the skylight becomes the strongest source of light. In our model the skylight comes directly from above and is limited to a direct lighting part I_h. Thus the formula for the calculation of the cloud vertex brightness is now:

$$I = I_a k_a + (I_d + I_h) k_d + I_r k_r \cos\phi + I_m k_m \left(\frac{-\cos\varphi + 1}{2} \right)^n \qquad (12)$$

Fig. 10 shows two clouds lit by the sun and skylight with the sun in the back of the spectator (left) and with some back light (right).

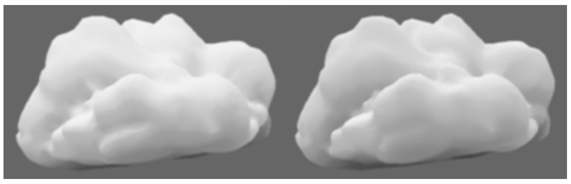

Fig. 10:Two examples of clouds lit by sky and sun light.

The Σd_i term in the brightness calculation (5-8) naturally introduces shadows in our visualisation. If a second cloud is situated within this shadow (as in fig. 11), it appears darker.

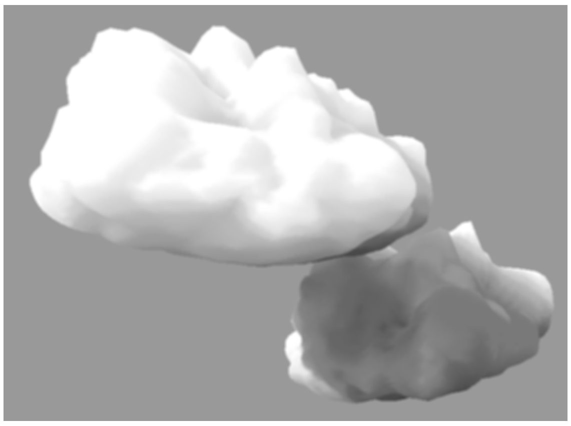

Fig. 11: Shadows on clouds.

Transparency and opacity

For the calculation of the opacity of the vertices we use view rays. Starting from the viewer's eye they cross the vertices P_i of each triangle. We call the distance, which a viewing ray passes after the vertex P_i within the cloud, *thickness* s. Of course s depends on the eye position. The opacity O_i (see fig. 12) of a vertex is proportional to the respective thickness s. We also choose a constant thickness T starting from which a cloud appears completely opaque. However, with this simple method problems can occur while calculating animations (fig. 12,13).

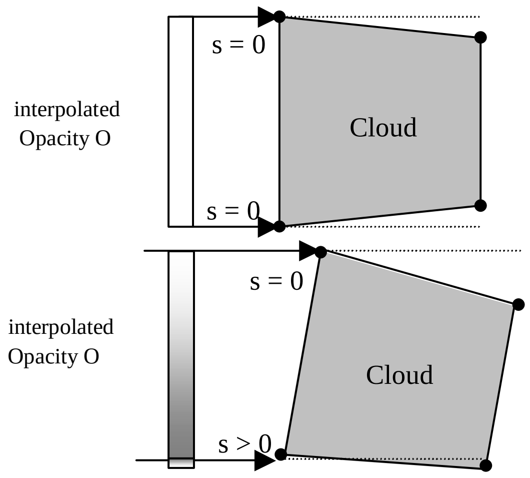

Fig. 12: After a small rotation the thickness s for individual vertices can change dramatically .

The thickness s and thus at the same time the opacity O_i can change suddenly for individual vertices after only a small turn of the cloud (see fig. 13). Thus the observer can get an impression of a ring of flickering triangles around the cloud during its slow rotation.

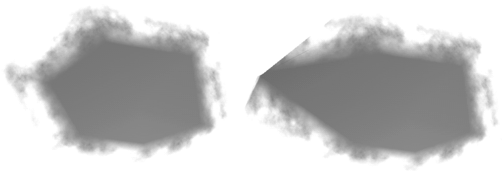

Fig. 13: Sudden vertex opacity changes after a rotation: the right cloud is turned around 1° around the y axis, compared to the right cloud.

A method to suppress flickering triangles is to consider the angles between view rays and cloud triangles (fig. 14). The opacity is still determined from s but is afterwards weighted with a function of the angle γ:

$O_P = f(\gamma)s.$

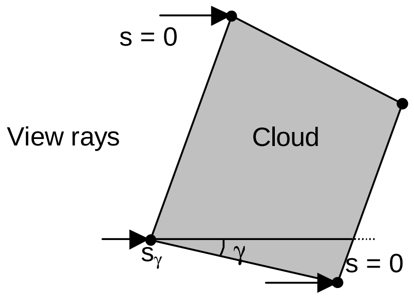

Fig 14: Computation of s depending on γ.

For $\gamma = 0°$, $s(\gamma)$ should be 0. In this way we achieve that during a slow turn the originally transparent cloud does not suddenly become visible, but it slowly fades in. As a well suitable function for this fading we found by experiments

$$f(\gamma) = 1 - \cos^n \gamma \ , \qquad (13)$$

where n describes the rate of the transition. Useful values of n are found within the range of $n \in [6 \ldots 12]$.

This technique works very well with convex clouds. In the general concave case however it produces new unpleasant effects (fig. 15).

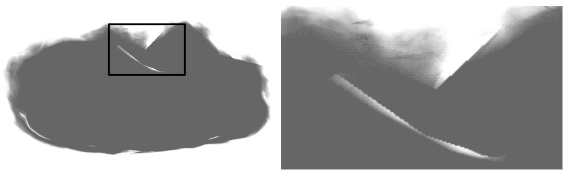

Fig. 15: Problems with angle-dependent opacity.

They occur within thick areas inside of the clouds, as soon as some triangles are situated almost parallel to the view rays. In this case is $f(\gamma)$ is almost zero, and actually completely opaque areas become transparent.

In order to moderate these problems that occur at the internal outlines of concave clouds, the calculation of the opacity is executed in such a way that the vertices of triangles situated directly in front of further cloud sections are rendered more opaque.

To calculate the opacity of the point P_i we write:

$$O_i = f(\gamma) d_{i,0} + \left(1 - \frac{x}{E}\right) \cdot d_{i,1} \qquad (14)$$

with

$d_{i,0}$: the length of the first part of the ray running *inside* of the cloud,

x: the length of the part of the ray running *outside* of the cloud (see the right part of fig. 17).

$d_{i,1}$: the length of the second part of the ray running *inside* of the cloud,

We choose quite a large constant E = 4D. Errors resulting from a too inaccurate overlay of the outlines are now substantially reduced. Fig. 17 shows the changed thickness calculation schematically.

The opacity of the cloud now rises within the middle area, which was otherwise completely underestimated, and approximates the real opacity (see fig. 16).

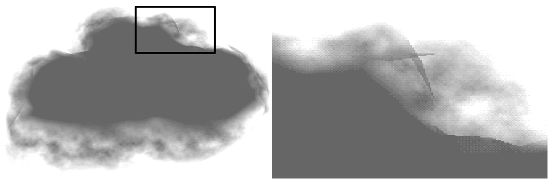

Fig. 16: Reduction of the opacity problems by our modified computation method.

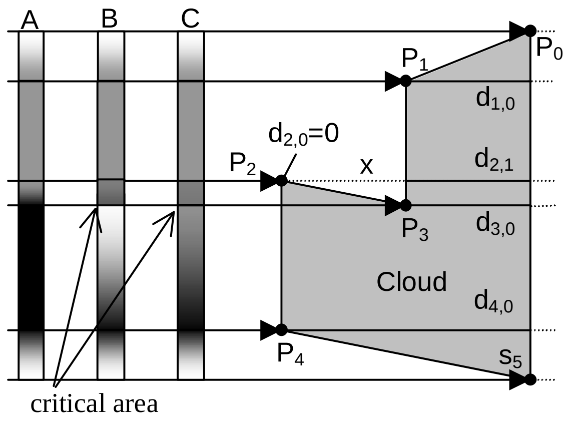

Fig. 17: The real opacity A compared with the opacity B calculated with the first method and the opacity C computed with the use of $d_{2,0}$, $d_{2,1}$ and x. The extreme transparency difference is avoided.

Semi-transparent texturing

In order to be able to show details, the transparency on the cloud surface is distorted by a two-dimensional density function. Our technique is similar to Gardners [3] and works with semi-transparent textures (see fig 18). Please see [17] for more details.

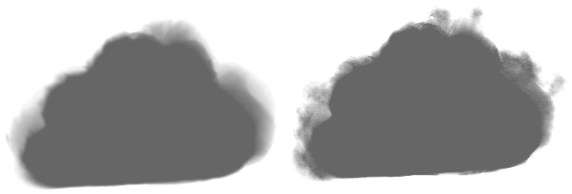

Fig. 18: A cloud without (left) and with semi-transparent textures (right)

4. RESULTS AND FURTHER WORK

Fig. 19 demonstrates that our methods are suitable to produce different types of cumulus clouds, e.g. cumulus fractus, cumulus humilis, cumulus mediocris and cumulus congestus. Fig. 20 shows the suitability of our algorithms for the production of animated clouds from varying original isosurface data: if the original polyhedrons change slowly the modification of resulting cloud geometry is just as slow. Consequently there is a strong coherency between two subsequent frames of the animation.

Fig. 21-24 show several cloud visualizations as they are needed by our target application, TV weather presentation with augmented video.

We tested our routines on a SGI Visual Workstation with a Pentium-III-450 processor. The times consider the processing of our entire procedure, including the refinement, rounding and deformation of the given isosurface geometry, the calculation of the lighting and transparency, as well as the rendering of a typical PAL-resolution frame (768x576 pixels). Background cirrus clouds in these pictures are not generated with our method.

Triangles	18510	42054	70792	89084	103270
Time (s)	2,11	2,73	5,35	6,32	6,88

Table 1: Performance of our rendering algorithm

In these examples geometry generation uses roughly 15 % of the time, lighting 60 % and rendering 25 %.

Our further work will include the calculation of the colours of the sky and the clouds themselves from the sun position and weather simulation data. Of course, more visualisation methods for different cloud types are needed. The smooth integration of the sky and cloud visualization in video sequences will be our further focus.

ACKNOWLEDGEMENTS

We thank Mr. Koppert from Deutscher Wetterdienst DWD for the meteorological data.

REFERENCES

1. Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., Nishita, T.: A Simple, Efficient Method for Realistic Animation of Clouds. *Proceedings of ACM SIGGRAPH, 2000.*
2. Ebert, D., Musgrave, F.K, Peachey, D., Perlin, K., Worley, S.: Texturing & modelling: a procedural approach. *AP Professional, Academic Press*, Chestnut Hill, MA, USA, 1998.
3. Gardner, G.Y., Visual Simulation of Clouds, *Proceedings of ACM SIGGRAPH 1985*.
4. Gamito, M.N., Lopes, P.F., Gomes, M.R.: Two-Dimensional Simulation of Gaseous Phenomena Using Vortex Particles, *Eurographics Proceedings* 1995.
5. Kajiya, J.T., B.P. von Herzen, RayTracing Volume Densities, *Proceedings of ACM SIGGRAPH* 1984.
6. Knöpfle, Ch., 3D Weather Forecast in a Virtual Studio. *FhG IGD, internal report*, August 1997.
7. Max, N., Efficient Light Propagation for Multiple Anisotropic Volume Scattering, Proceedings of th 5th Eurographics Workshop on Rendering, Darmstadtt, Germany, 1994.
8. F. Neyret: Qualitative Simulation of Convective Cloud Formation and Evolution, In *Eurographics Proceedings* 1997.
9. T. Nishita, Y. Dobashi, E. Nakamae: Display of Clouds and Snow Taking into Account Multiple Anisotropic Scattering and Sky Light, *Proceedings of ACM SIGGRAPH*, 1996.
10. Ruprecht, D., Müller, H.: A Meta Scheme for Iterative Refinement of Meshes, *Mathematical Visualisation*, Springer Verlag, 1998
11. Sakas, G., *Fraktale Wolken, virtuelle Flammen. Computer-Emulation und Visualisierung turbulenter Gasbewegung*. Springer Verlag, Berlin, Germany, 1993.
12. Schröder, F., *Visualisierung meteorologischer Daten*, Springer, Berlin, 1997.
13. Stam, J., Fiume, E., Turbulent Wind Fields for Gaseous Phenomena. *Proceedings of ACM SIGGRAPH 1993*.
14. Trembilski, A., Two Methods for Cloud Visualisation from Weather Simulation Data, *Proceedings of WSCG 2000, Plzen, Czech Republic.*
15. Trembilski, A., Brossler, A., "Transparency for Polygon Based Cloud Rendering", to appear at the ACM Symposium on Applied Computing (SAC), Multimedia and Visualization Track, Madrid, Spain, March 10-14, 2002.
16. Unbescheiden, M., Trembilski, A., Cloud Simulation in Virtual Environments, *Proceeding of VRAIS 1998*, Atlanta, USA.
17. World Meteorological Organization, *International Cloud Atlas*, Vol I+II, 1987.

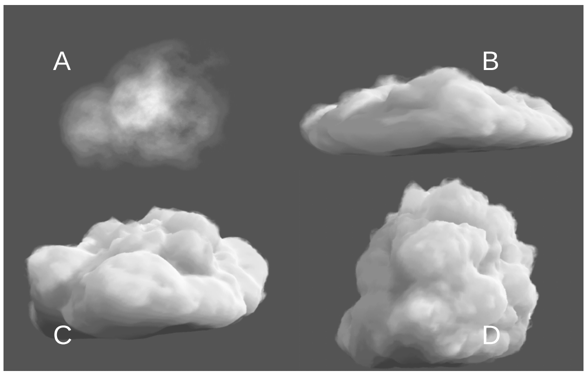

Fig 19: Cumulus fractus (A), Cumulus humilis (B), Cumulus mediocris (C) und Cumulus congestus (D)

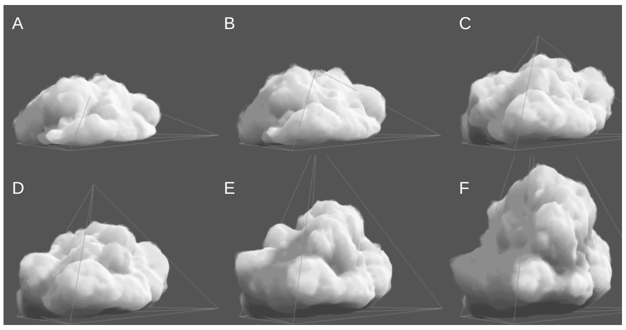

Fig 20: Cloud animation by the use of a varying base geometry

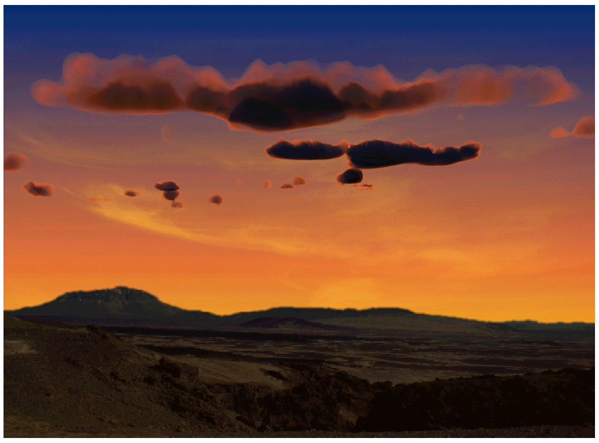

Fig. 21: Isocumulus clouds lit by the sunset. Mie scattering effects are clearly visible. Background cirrus clouds result from the original picture.

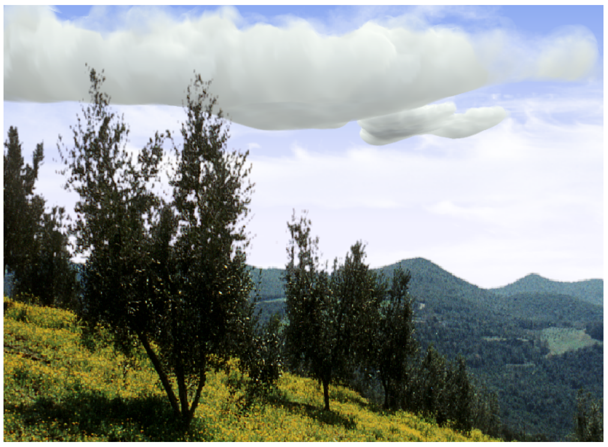

Fig.22: Isocumulus clouds. Background cirrus clouds result from the original picture.

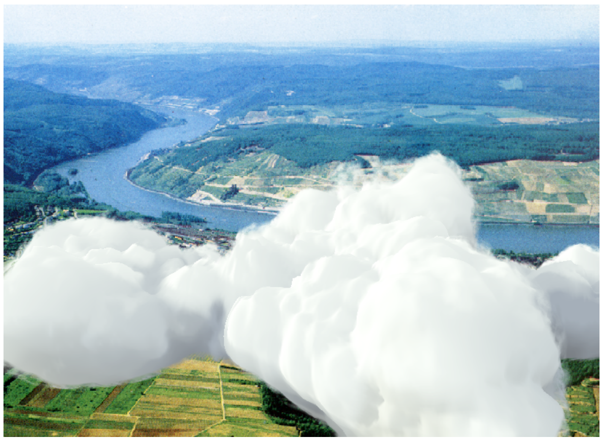

Fig.23: Isocumulus clouds with some back light.

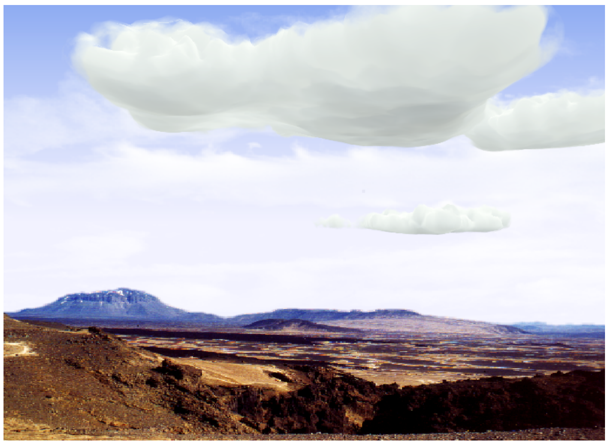

Fig. 24: Isocumulus clouds. Background cirrus clouds result from the original picture.