Real-time Visualization of Clouds

Paul Heinzlreiter, Gerhard Kurka, Jens Volkert

GUP Linz, Johannes Kepler University Linz
Altenbergerstraße 69, A-4040 Linz, Austria/Europe
heinzlreiter@gup.uni-linz.ac.at
http://www.gup.uni-linz.ac.at

ABSTRACT

Rendering realistic cloud images is a challenging computational intensive task. The high computational demand makes physics-based on-the-fly rendering impossible for real-time applications on low-cost workstations.
We present an approach to overcome this problem by using alpha-blended billboard textures. During a pre-processing step a series of pre-calculated images is generated for each cloud object of the scene. The images are showing each cloud object from different view points.
For a given camera position on the ground the final cloud depiction is composed by an appropriate subset of alpha-blended images, assigned to billboard textures. In so doing, we are able to create a smooth transition between arbitrary camera positions on the ground.

Keywords: clouds, textures, billboarding, interactive visualization

1 INTRODUCTION

An important goal of computer graphics is the visualization of real world objects. In order to create realistic looking images, it is often necessary to simulate the physical nature that determines the visual appearance of our environment.
The simulation of atmospheric cloud effects is a challenging topic in the scope of the aforementioned research area. There are several reasons which make rendering of clouds a complicated and tedious task, especially for real-time applications:

- General hardware accelerated real-time graphics algorithms operate on polygonal surfaces and vertex-based geometries. In contrast, clouds have a volumetric amorphous structure.
- As a well known real-world object, the realism of synthetically generated cloud images is very susceptible to artefacts introduced by the rendering or simulation process.
- Simulating the effects of light propagation through cloud objects is a time consuming and computational intensive process.

In order to render realistic looking images there are two possibilities:

1. Rendering is done by using fractal-based methods [Ebert98] that repro-

duce realistic looking 2D-images, which are mapped on the sky by using one or more layers of alpha textures.

2. Employing computational intensive physics-based simulation, that tackles the physical nature of light propagating though the objects.

Both approaches come along with advantages and disadvantages. As long as we are dealing with thin cloud layers on the upper end of the atmosphere the fractal methods will provide an accurate run-time efficient solution. Due to the large distance between camera and cloud layer it will not be necessary to create different views from different viewing angles for real-time applications.
This situation changes in case of inherent spatial cumulus-clouds, where we have to deal with complex light effects that depend on the position of the sun and on the camera position. This cloud-type can not be satisfactory modelled by fractal methods for different view points.
Physics-based simulation provides a straight-forward solution to this type of cloud objects. The simulation creates a three dimensional density map, which describes the density of the water vapor the object consists of. The density-map is rendered by means of a volume renderer, which takes into account the specific scattering of light-rays, caused by tiny water dropplets inside the object.
Unfortunately, due to its high computational demands, using physics-based cloud simulation is not appropriate to real-time rendering. Even on todays fastest processors rendering times of about a few minutes per image are common.
In order to overcome this problem and to exploit texture mapping hardware we employ a simple but efficient billboard approach. For each cloud object a series of images is created by using numerical expensive physics-based simulation. Each image shows the entire object from a different view point on the ground. Given an arbitrary ground located view point, the final image is composed by blending a subset of the pre-rendered images. By carefully adjusting the alpha values of the images to be blended, smooth and popping free transitions between different cloud-images are obtained.

The paper is organized as follows: The next section describes research work related to the topic of this paper. Section 3 gives an overview of the system structure and the subsequent sections describe the various subsystems and their tasks in more detail. Section 7 and 8 conclude with a presentation of the achieved results and a brief description of possible improvements.

2 RELATED WORK

A number of researchers have been working on the topic of realistic cloud representation:

- Gardner [Gardn85] focuses on an approach using ontogenetic cloud rendering with impressive results. The modelling of the clouds is done using textured ellipsoids and transparency.
- Stam [Stam94] presents an approach for stochastic rendering of density fields by generating a realization of the intensity field to be visualized directly out of its statistics.
- Nishita et al. [Nishi96] describe physics-based rendering of static cloud images which yields good results. An adaption of their algorithm was used to render the cloud images in our approach.
- Dobashi et al. [Dobas00] present an approach for cloud animation which uses a fixed observer position and a realistic model of cloud development based on a finite state machine.

- Harris et al. [Harri01] use particle systems to represent the cloud model and dynamically generated imposters to achieve an interactive frame rate.
- Elinas et al. [Elina01] present an approach using ellipsoids as geometric primitives for real-time cloud rendering.

3 SYSTEM STRUCTURE

The implemented system consists of three independent subsystems:

1. cloud generator
2. cloud renderer
3. scene visualization

The following section describes the generation of the three-dimensional cloud model.

4 THE CLOUD MODEL

The cloud data used as input for the rendering process consists of a spatial density distribution of water vapor which is constructed by the cloud generator. It is described by a standard voxel model. Each voxel specifies the density at its location.
Before calculating the density distribution the structure of the cloud must be specified. It is modelled by a couple of ellipsoids [Gardn85, Taxen99]. Each one is specified by following parameters:

- position of the center given in object coordinates
- length of the half-axes
- density at the center
- density at the surface

The shape of intersecting ellipsoids is used to approximate the spatial density distribution of realistic looking cumulus-clouds. The density distribution of a single ellipsoid is calculated by linear interpolation between its density values at the center and the surface. To determine the density values of the voxel model for each voxel the functions of the ellipsoids containing the voxel position are evaluated and the resulting density values are added. Finally, we have to restrict the maximum density to 1.0 by reducing greater values.
The following section discusses the image rendering process, which is taking into account light absorption and scattering.

5 IMAGE RENDERING

The image generation process requires the following input parameters:

- cloud density distribution
- cloud position
- observer position
- view direction of the observer
- direction of the incident light
- position, size and resolution of the projection plane

Image rendering consists of the following steps:

1. calculation of light absorption
2. calculation of light scattering
3. ray-casting for image generation

The implemented rendering algorithm is an adaption of the method described in [Nishi96]. The first two steps operate on the voxel model while the third realizes the projection onto the two dimensional space.

5.1 Light Absorption

During this step of the rendering algorithm the light intensity remaining after absorption is calculated for each voxel. It can be defined as follows:

$$I_p(\lambda) = I_0(\lambda) e^{\tau(S,\lambda)} \quad (1)$$

I_p denotes the light intensity after absorption, I_0 is the incident light intensity, λ is the wave length of the incident light and $\tau(S,\lambda)$ the optical length of the calculated volume, which is determined by the density distribution within the passed volume:

$$\tau(S,\lambda) = \int_0^S \beta\rho(s)ds \quad (2)$$

S denotes the length of the light ray through the volume in question, $\rho(s)$ is the water vapor density in the area passed by the light ray and β is the absorption coefficient which depends on the particle types (water, ice, ...) contained in the cloud.

The calculation is implemented by a recursive algorithm providing flexibility with respect to the direction of the incident light. It comprises following steps:

1. determining the light intensity of the source voxel (this may include a recursive invocation of the algorithm if the source voxel isn't already calculated)
2. evaluating Eq. 1. to calculate the remaining light intensity for the current voxel.

5.2 Light Scattering

The calculation of light scattered by water droplets in clouds is very important, because it has a significant effect on the visual appearance.
For cloud droplets Mie scattering is the appropriate calculation model because due to the size of the water droplets the light distribution is anisotropic.
The angular dependency of the light distribution is given by the scattering phase function [Corne92]:

$$F(\theta,g) = \frac{3(1-g^2)}{2(2+g^2)} \frac{(1+cos^2\theta)}{(1+g^2-2g\,cos\,\theta)^{\frac{3}{2}}}$$

The value g quantifies the asymmetry amount of the light distribution.
For calculating light scattering it is necessary to take into account higher orders of scattering due to their importance for the visual impression of the cloud. Therefore we compute the light contribution until the third order of scattering.
Due to the long runtime of the calculation we use

- a reference pattern for light scattering [Nishi96] and
- a cosine lookup table.

The reference pattern contains the percentage of light intensity received by a voxel from its neigbours due to light scattering. This contribution ratios are stored in a three-dimensional array. The target voxel is located in the center of the data structure.
There are several reasons for using a reference pattern:

1. The amount of light transmitted between pairs of distant voxels by scattering is negligible.
2. The light scattering distribution is independent of the voxel position. Therefore the reference pattern can be calculated in advance.
3. The pattern contains all calculated orders of scattering.

The cosine lookup table is a straight-forward approach to reduce the calculation time of

trigonometric functions. For each evaluation of the scattering phase function during the reference pattern calculation the cosine of the scattering angel has to be evaluated twice. By storing the cosine values of the angles between 0° and 180° in a lookup table it is not necessary calling the trigonometric function.
The aforementioned scattering calculation would be too computational intensive for standard workstations without the expoitation of the these algorithmic impovements.

5.3 Projection

The final image is generated by using ray casting on the voxel model. For each pixel of the projection plane a ray is sent through the cloud volume and the intensities along this ray are accumulated.
The following section focuses on the utilization of the rendered cloud images for real-time visualization.

6 INTERACTIVE SCENARIO

A pseudo outdoor scene derived from a random height field was used to test the described approach.
In the scene each cloud is represented by a four-sided polygon, onto which the cloud image is mapped.

6.1 Observer Positions for Image Rendering

Generating the images requires determining the observers position for each image relative to the cloud. (see section 5).
The positions used for rendering are equally distributed on the contour of a circle around the projection of the cloud center on the ground (Fig. 6.1.)

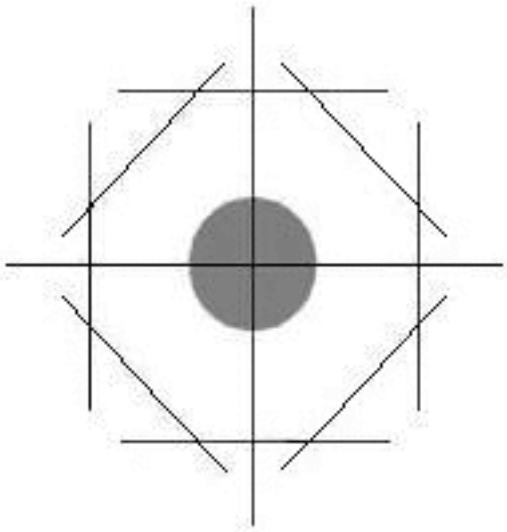

Figure 1: Projection plane positions for eight images from a bird's-eye view (cloud in the center)

6.2 Billboard Adjustment

To create a realistic impression of a cloud it is necessary that the normal vector of the polygon representing the billboard points directly to the observer position because otherwise the geometrical structure of the cloud representation would be revealed.
Therefore billboarding is used by rotating the polygon around two axes to reflect the changes of the viewers position. Most important is the rotation around the y-axis (up-vector) because only the front face of the polygon is textured. The rotation around the z-axis is necessary because otherwise a viewers position somewhere under the cloud would reveal the polygonal nature of the celestial object.
The angle for y-axis rotation is determined by calculating the angle between the direction vector of the x-axis and the vector pointing from the cloud center to the observers position.
The angle for z-axis rotation is the angle between the view direction of the observer and the normal vector of the polygon ranging from 0 to 90 degrees.

6.3 Texture Exchange and Blending

In order to adapt the visual appearance of the cloud according to the changes of the observers position it is necessary to map another texture on the billboard if the observer moves out of the validity range of the currently shown texture. In Fig. 6.3. the dashed lines (texture exchange borders, TEBs) seperate the texture validity ranges (TVRs) for eight textures.

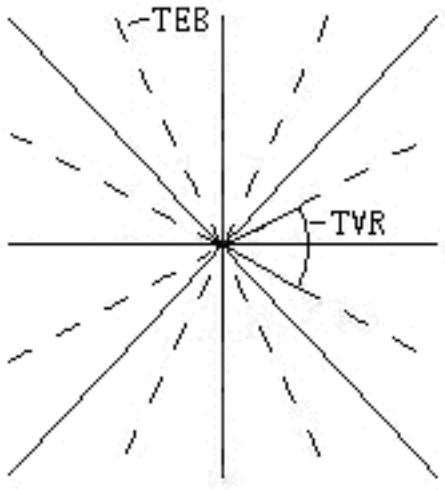

Figure 2: valid ranges (TVRs) and texture exchange borders (TEBs) for eight textures

Visualizing a realistic volumetric cloud requires the usage of enough textures rendered from different viewpoints. Our tests have shown that a convincing cloud visualization requires a minimum of 32 textures.

But there is still one remaining problem: Just swapping the textures would lead to unpleasant visual artifacts and therefore a smooth transition between two neighbouring textures is needed. To achieve this three billboards are used, which are arranged behind each other:

- the currently valid texture (according to the viewers position within its TVR) is mapped on the frontmost billboard.
- the two subsequent billboards are mapped with the two neighbouring textures sorted by the angular distance towards their TVRs.

The influence of the textures is controlled by alpha values, which are changed according to the movements of the observer. Futhermore it is necessary to change the sorting of the textures so that the most important texture remains on the frontmost billboard.
For calculating the correct texture sorting and alpha values the position of the observer given in cartesian coordinates is mapped onto polar coordinates (ϕ, r) with the clouds position at the center of the coordinate system. Depending on the value of ϕ the correct textures are selected. The appropriate alpha values are set according to the angular distance between the observer position and the next texture exchange border (TEB, see Fig. 6.3.). The opacity O_0 for the front texture is calculated using the following formula:

$$O_0 = 1 - \frac{abs(\frac{180}{n} - a)}{\frac{180}{n}} \qquad (3)$$

In this equation n denotes the number of textures used and a is the angular difference between the observer position in polar coordinates and the direction of the projection plane's normal vector used for rendering the image.
The alpha values for the two remaining textures are calculated by weighting the remaining opacity using the angular distance between the viewers position and the border towards the validity area of the texture whose alpha value is to be calculated:

$$O_1 = \begin{cases} (1 - O_0) * \frac{a}{\frac{360}{n}} & \text{if } a > \frac{360}{n} - a \\ (1 - O_0) * \frac{\frac{360}{n} - a}{\frac{360}{n}} & \text{else} \end{cases}$$

$$O_2 = 1 - O_0 - O_1$$

To produce a natural visual represention for an eyepoint located directly under the cloud we have to extend the decribed approach by using one more billboard textured with

an image of the cloud rendered using an observers position directly under the cloud. The alpha value of this texture O_B is changed according to the current value of the angle e enclosed between the view direction vector of the observer and its projection on the xz-plane:

$$O_B = \frac{e}{90} \qquad (4)$$

6.4 Shadows

Shadow projection on the landscape caused by the clouds is another important issue to consider to achieve a realistic visualization. Therefore shadow projection was implemented using a shadow texture calculated out of the threedimensional cloud model using an observer position directly under the cloud. Before modulating the texture on the ground it is necessary to invert the intensity values of the retrieved luminance texture.

7 RESULTS

The following pictures show screenshots of the interactive application:

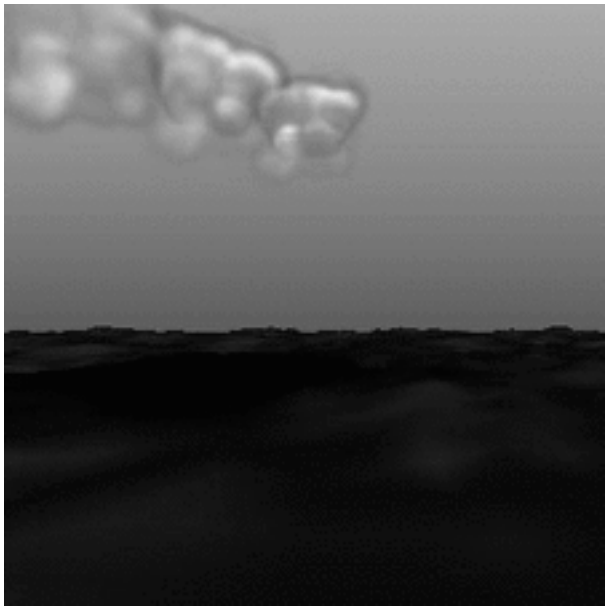

Figure 3: Cloud formation and landscape

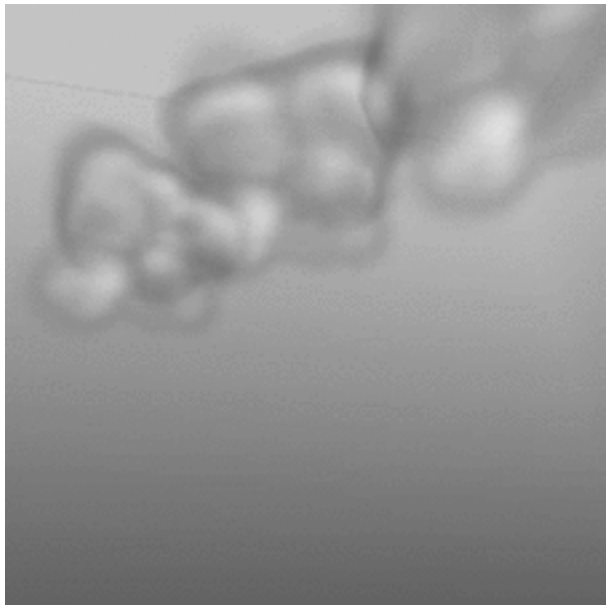

Figure 4: Closer look at the cloud formation

Figure 5: Shadow projection on the ground

7.1 Performance Measurements

The performance data presented in this section has been measured using a PC with an AMD K6-II 400 MHz processor and 128 MB RAM running Linux 2.2.16.
The cloud model used for calculation contains 100x100x100 voxels and the projection

plane's resolution is 256x256 pixels.

The runtime of the cloud model generator is 18 seconds per ellipsoid. The calculation times required for rendering one cloud image are presented in table 1.

Reference pattern	32 seconds
Absorption	44 seconds
Scattering	328 seconds
Ray-casting	334 seconds
Overall	738 seconds

Table 1: cloud renderer runtime

The interactive walkthrough of the scenario can be performed in realtime due to the hardware support for texture mapping and blending.

8 CONCLUSIONS AND FURTHER WORK

In this paper an approach for the interactive visualization of clouds using an physics-based model for rendering is proposed. Using textured billboards enables real-time visualization of clouds based on a physical model. This is achieved by calculating the cloud images for various observer positions in a pre-processing step. However, there are some open issues which still have to be adressed:

- extension of the observers mobility by using more textures
- improvement of cloud modelling

REFERENCES

[Bohre95] Bohren C. F.:*Atmospheric Optics*, Encyclopedia of Applied Physics, VCH Publishers, Volume 12, pp. 405–434, 1995

[Corne92] Cornette W. M., Shanks J. G.: *Physical reasonable analytic expression for the single-scattering phase function*, Applied Optics, Vol. 31, No. 16, pp. 3152–3160, 1992

[Dobas00] Dobashi Y., Kaneda K., Yamashita H., Okita T., Nishita T.: *A Simple, Efficient Method for Realistic Animation of Clouds*, ACM SIGGRAPH, pp. 18–28, 2000

[Ebert98] Ebert D. S., Musgrave F. K., Peachey D., Perlin K., Worley D.: *Texturing & Modeling. A Procedural Approach*, Second Edition, AP Professional, 1998

[Elina01] Elinas P., Stürzlinger W.: *Real-time Rendering of 3D Clouds*, Journal of Graphics Tools, Volume 5, Number 4, pp. 33–45, 2001

[Gardn85] Gardner G. Y.: *Visual Simulation of Clouds*, Computer Graphics, Volume 19, Number 3, pp. 297–303, 1985

[Harri01] Harris M. J., Lastra A.: *Real-Time Cloud Rendering*, Eurographics, pp. 76–84, 2001

[Nishi96] Nishita T., Dobashi Y., Nakamae E.: *Display of Clouds taking into Account Multiple Anisotropic Scattering and Sky Light*, ACM SIGGRAPH, pp. 379–386, 1996

[Stam94] Stam J.: *Stochastic Rendering of Density Fields*, Proceedings of Graphics Interface '94, pp. 51–58

[Taxen99] Taxen G.: *Cloud Modeling for Computer Graphics* Master's thesis, Royal Institute of Technology, Stockholm, Sweden, 1999

[Woo99] Woo M., Neider J., Davis T., Shreiner D.: *OpenGL Programming Guide*, Third Edition, Addison Wesley, 1999