Fast algorithm for cloud rendering using flat „3D textures”

Dariusz Domański
Institute of Computer Science,
Warsaw University of Technology,
ul. Nowowiejska 15/19,
00-665 Warsaw, Poland
ddomansk@ii.pw.edu.pl

Abstract

This paper presents an efficient method to render clouds in their natural dynamic (*i.e.*forming, dissipating, flowing). The algorithm uses flat „3D textures” to represent volumetric data. This is used to compute the amount of light reaching each voxel. The light values are computed in real time and then visualised using a slicing technique to enable on-line interaction with the cloud model.

Keywords: natural phenomena visualization, 3D rendering, real-time animation

1 INTRODUCTION

Clouds are often included in online applications such as flight simulators and video games. The problem is that rendering clouds in a fast and realistic way is very complex. Various simplifications are used to allow the real-time rendering of cloudy scenes, however very often they decrease the quality of the animation. A very valuable objective is to be able to produce scenarios where the observer is flying through clouds.

In this paper, simple cellular automata was used to obtain cloud data in a few processing steps. These data are then processed with the proposed cloud volume lighting algorithm and visualised with a technique using slices to approximate to the volume.

2 PREVIOUS WORK

The most realistic cloud rendering method, so far produced, was developed by Harris and Lastra [HL01]. The algorithm takes into account the phenomena of light scattering: multiple forward scattering along the light direction and single scattering towards the viewer.

This work was extended by Harris *et al.*[HBSL03] to model realistic motion of clouds on GPU via stable fluid simulation introduced by Jos Stam. The improved algorithm deals with the rendering aspect of the problem as they mentioned a method for rendering in 3D condensed vapour density provided by simulation. This extension is discussed in more detail by Harris in his PhD dissertation [Har03].

Light scattering was introduced even earlier by Dobashi *et al.*[DKY$^+$00], however, theirs method modelled single scattering only. A common feature of all these rendering algorithms is that they occur in two phases. The first pass calculates the scattering along the light vector and creates the lighting information. The lighting data is then applied to the cloud model in the second phase of the algorithm.

A simpler approach was used by Vane [Van04] based on work by Behrens and Ratering (widely cited in Vane's paper). The visual effect of his work demonstrates that Behren's algorithm is suitable for shadowing but not for lighting.

3 VOLUMETRIC DATA

For the proposed rendering method clouds are represented as density distribution model. From the array of cloud data generation methods the Dobayashi's algorithm was selected (described in detail in [DKY$^+$00]).

The computationally expensive process of the continuous distribution calculation (called *smoothing*) was spread over a number of animation frames. The assumption given for real-time applications is that each frame must be displayed in 40 milliseconds there is about half this time available for simulation or smoothing with visualization technique proposed in section 4.

4 VISUALIZATION MODEL

The most common approach to visualize volumetric data is to use a slicing algorithm. The basic idea is to take a slice of volume data, render it onto texture and paste this texture on quads in 3D space. Figure 1 shows mapping of volumetric data onto a flat „3D texture” which is used to cover the slice's surface.

As denoted by Vane in [Van04] there are two classes of slicing algorithms: object aligned and view aligned. For the implementation of the visualisation model object aligned slices was chosen. The key advantages of

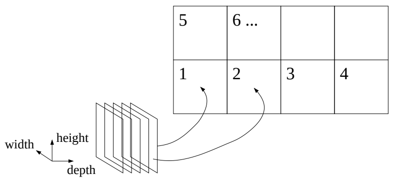

Figure 1: Mapping volumetric data of discrete space onto flat „3D texture”

this method are its efficiency and its simplicity, considered both methods introduce similar artifacts: when the viewer is very close (or inside) the volume the space between the planes becomes apparent and the whole object can be perceived as a stack of slices.

5 RENDERING

The rendering algorithm proposed in this section is similar to the original concept of Harris and Lastra [HL01]. The key difference is that the method described below adopts images of volume rendered in slices and then uses hardware blending function to process the whole slice simultaneously. This approach speeds up the whole process as the additional buffer used to read back the illumination for each pixel can be read as one operation for whole slice.

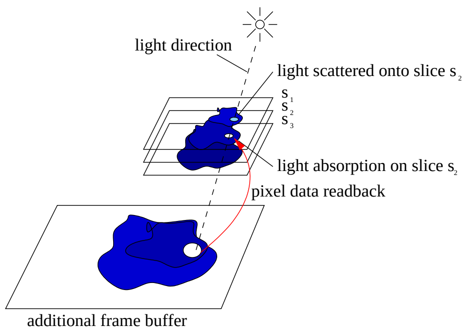

Figure 2: General idea for cloud illumination algorithm using slices: illuminating slice s_2

Flat „3D textures” illuminated by presented algorithm 1 (picture 2) can now be displayed with visualization technique described in section 4.

6 CONCLUSIONS AND FUTURE WORK

The sample results included in this paper (picture 3) were actually generated by a real time application running at 30*fps* and better.

The proposed lighting algorithm works well for reasonable sizes of volume data and numbers of slices (experiments were done with volume $64 \times 64 \times 64$ with 64 slices). Further extensions, which will enable the natural simulation of day lit skies and rendering of clouds in a large scale space will add significant realism to animated scenes.

Algorithm 1: Cloud lighting algorithm

```
input: model viewed align its axes $\vec{X}$, $\vec{Y}$ and $\vec{Z}$
       (three sets of slices covered with textures),
       normalized vector of light direction $\vec{l}$,
       global light color $c$,
       scattering coeficient $a$,
       absorption factor $e$

/* determine the set of slices to be used for lighting
   and the order they are processed */
1  GetMaxABSVecIndex($\vec{l}\cdot\vec{X}, \vec{l}\cdot\vec{Y}, \vec{l}\cdot\vec{Z}$)
2  CreateAndFillBuffer($b, res_w \times res_h, c$)
3  foreach slice $s$ do
4      foreach pixel $p_{(u,v)}$ do
           /* compute pixel coordinates and radius
              corresponding to buffer content */
5          $(u_b, v_b, r) \leftarrow$ ProjectPixel($b, p_{(u,v)}$)
6          $\tau \leftarrow \sum_{u'_b=-\frac{r}{2}}^{\frac{r}{2}} \sum_{v'_b=-\frac{r}{2}}^{\frac{r}{2}}$
7              GetPixelAlpha($b, p_{(u_b+u'_b, v_b+v'_b)}$)
8          $\tau_{(u,v)} \leftarrow \tau \cdot \frac{a \cdot e}{4 \cdot \pi^2 \cdot r^2}$
9          SetPixelColor($s, p_{(u,v)}, \tau_{(u,v)} \cdot c_{(u,v)}$)
10     DrawSlice($b, s$)
```

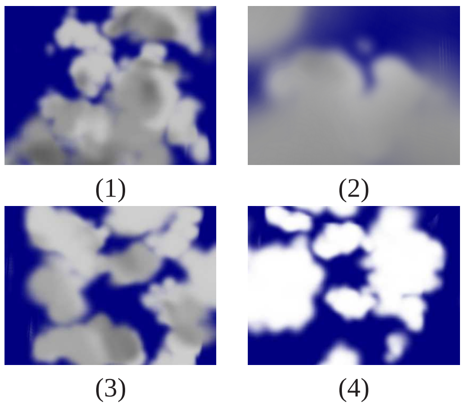

Figure 3: Sample frames from cloud simulation: (1) and (2) show same cloud model viewed from different perspectives; (3) experiments with increasing scattering factor; (4) unlit cloud model (very bright day)

REFERENCES

[DKY$^+$00] Y. Dobashi, K. Kaneda, H. Yamashita, T .Okita, and T. Nishita. A simple, efficient method for realistic animation of clouds. In *ACM SIGGRAPH*, pages 19–28, 2000.

[Har03] M. J. Harris. *Real-Time Cloud Simulation and Rendering*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 2003.

[HBSL03] M. J. Harris, W. V. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In *Graphics Hardware*, 2003.

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

[Van04] E. Vane. Cloud rendering using 3d textures, 2004.