Visualization of 3D clouds using free forms

Sergei I. Vyatkin, Boris S. Dolgovesov
Institute of Automation and Electrometry SB RAS, Novosibirsk, 630090 Russia
Alexander V. Yesin
Institute of Informatics Systems, SB RAS, Novosibirsk, 630090, Russia

Abstract

At the present work the method of fast rendering the translucent objects formed by free form surfaces is offered. For the majority flight simulators or car simulators atmospheric effects are realized in the manner of 2D texture (clouds) or simply resulting colour in the pixel is blended with some weight, which depends on the coordinate Z with the white colour (fog). In given work realization of three-dimensional atmospheric effects is offered, in which the information on deep complexity of scene is presented. The purpose of the given work is a realization rendering such objects, filled by the texture, for which integral of transparency on the given length is fast calculated.

Keywords: *free forms, 3d clouds, voxels, multilevel ray casting.*

1. INTRODUCTION

In work [1] the method of generation a 3D clouds and fog proposed by company Evans & Sutherland is described. In this project 3D clouds are presented as layers of ellipsoids, estimation of necessary amount of ellipsoids for more or less realistic scene is done. The model to calculate colour of object observed through clouds is described. Since clouds divided into layers, colour *ci* and attenuation constant *pi* are assigned for each layer. Formula for colour calculation of object observed through clouds is presented.

$$Apparent\ color = (Actual\ Color) * exp[-0.5 (a_0 + a_1) d] + 0.5 (c_0 + c_1) * \{1 - exp[-0.5(a_0 + a_1) d]\},$$

where *Actual Color* is real colour of object.

d is a distance between observer and object.

As possible understand from the given formula colour and constant of attenuation is linear interpolated between two layers. If object close several layers, author resorts to such term as a virtual colour of object. For the apparent difficulty of this notion stands enough simple thing, first *Apparent color* is calculated for two layers, afterwards result is substituted as *Actual Color* for two other layers and so on. The author notes that attenuation constant must increase as increasing a layer height. Such approach reduces an amount of calculations when calculates colour of object on the land observed through N layers. If all layers sufficiently transparent it is necessary to carry out calculations of colour of object for each layer. If upper layers sufficiently thick the observer can see nothing except the colour of given layer of clouds. The benefit of such strategy of visualization of atmospheric effects is underlined in the article [2]. Visualization of complex scenes, for instance, terrain, includes a problem of displaying the objects under changing levels of details. Let us consider the terrain as example: if we have a grid of heights with sharply changing values, at animation possible to observe sudden changes on the silhouette of mountains. In the first place all said concerns to a case of polygonal approximations of terrain. So developers resort to using a fog. Then if fog is used, the sudden changes on the silhouette will not so noticeable. And if take more general case, it is possible do not display small details of objects far from observer, or not at all display distant objects. So it leads to reducing an amount of rendered objects, i.e. to reducing a number of processing primitives, and hereunder to increasing a frequency of generation frames.

In the article [2] the different ways of modeling of fog are described. The two ways of calculation of colour of pixel subjected to the fog are proposed:

1. Colour of pixel subjected to the fog is calculated in vertex of triangles, then approximates in the pixel as of values in vertex.

2. Colour of pixel subjected to the fog gets from lookup tables and does not depend on vertexes. Such way of calculation is called table or pixel method.

In the same way it is needed to distinguish two ways of calculation of attenuation. More precisely, the difference is concluded in the following: what to consider as a point of entry into the fog and on what distance to calculate attenuation.

1. Distance is a length between some plane and given point. Fog defined by the table possible to use with such way of calculation of attenuation only. On the one hand this rather fast method, but its using associates with arising number of artifacts, for instance, objects can appear and disappear in the fog when an observer moves.

2. Distance is a length of segment from the eye of observer to the object. This way more stable with respect to artifacts, but more expensive on computation time.

These are formulas to calculate function of attenuation:

Linear attenuation:

$$F = (fogend - d) / (fogend - fogstart),$$

where *fogstart, fogend* – points of entry and output in the fog.

d is a distance between the observer and the object.

Exponential attenuation:

$$F = 1 / (d * fogdensity)$$

fogdensity is a density of fog.

Quadratic exponential attenuation:

$$F = 1 / (d * fogdensity)^2$$

It is necessary once again to note that all above said is referred to polygonal techniques. In the article [2] is mostly concentrated on functions realized in Direct3D.

In the article [3] the new technique Elevation Maps is described. Its main idea is to use alpha-channel for storing of some values (conditionally heights), which are used to select appropriate

texture layer. In this work new way of rendering the three-dimensional atmospheric effects (3D clouds, smoke) is offered. To value of given method should refer using a small quantity of primitives, such as triangles and at the same time of the rich texture. However there is a restriction on the orientation of object when applying of a given method. That is to say

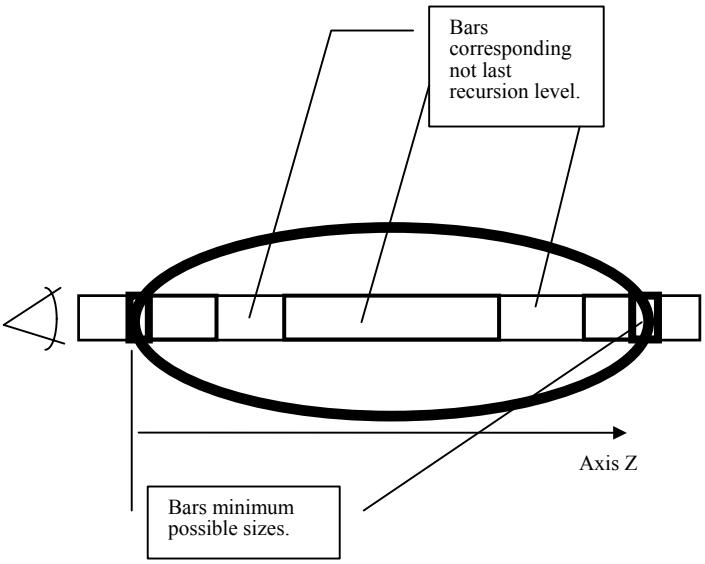

Figure 1. An optimized algorithm skips through uniform area.

impossible observe a silhouette of object under the certain angle.

In the flight simulator system MaxView [4] from Canadian company CAE Electronics clouds realized as a set of raster light points. To defects of given approach should refer a greater amount of primitives for modeling more or less realistic cloud. So in the system MaxView, in first, it is impossible simultaneously render greater groups of clouds, and in secondly, in such clouds are absent heterogeneity. Model of calculation of colour of object observed through such clouds does not correspond physical.

Also it is known one more technique of modeling of atmospheric effects: the form of clouds is built from base primitives; colour and transparency are taken from texture map. Probably the most difficult problem of visualization of such sort of objects is a modeling of atmospheric perturbations without additional calculations.

The simplest method of perturbations modeling of atmospheric effects offered in number of work is the applying a texture for the calculation of functions of density. In this case colour on surfaces of object gets from texture map, but function of transparency is calculated according to this colour (or in general gets from the other table), for instance, possible take density to the proportional colour, or inversely proportional.

Also it is known way of visualizations of atmospheric perturbations, which possible get by POV-Ray [5]. Summarizing all said above it is necessary to note that in most of mentioned articles the three- dimensional atmospheric effects are achieved basically by applying of rich textures with the small amount of primitives (excluding are [1] and [4]). The absence of forms of such objects causes the undesirable artifacts. So in the present work is offered to use alongside with the rich textures and complex form of objects.

2. RENDERING OF TRANSLUCANT OBJECTS FORMED BY FREE FORM SURFACES

The method was developed by authors who show how rendering of uniform translucent objects can easily be performed using a free form surfaces [6]. Figure 1 illustrates an optimized algorithm skips through uniform area. In given work the mechanism of rendering the objects with pseudo-heterogeneous density distribution was developed. It is used such types of functions of attenuation as to easily calculate integral of transparency for intervals along Z coordinate.

In given work the following method of modeling of atmospheric perturbations is applied: using a texture as function of colour and density. In this case colour for object surfaces gets from texture map, but function of transparency is calculated by this colour, for instance, pro rata colour, or inversely pro rata. By this method the objects modeled with the perturbations of density are processed such way either as uniform. Also processing speed differs unessential.

3. MODEL DESCRIPTION

Difference between ray casting and ray tracing algorithms is that in first from them in purposes of increasing speed processing are not traced secondary rays. When light passes through translucent regions it is also neglected a refraction and attenuation of secondary rays (Fig. 2). In given models considers only reflection and attenuation of light on the length

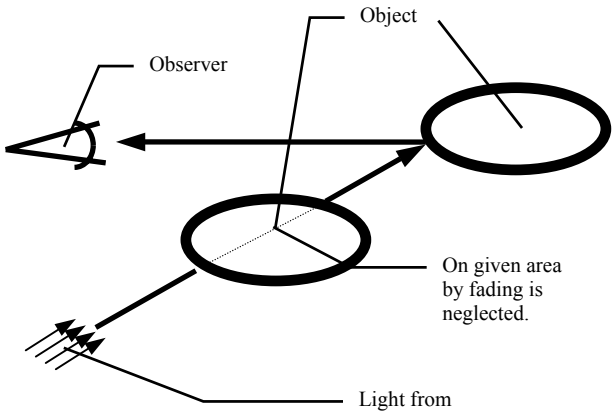

Figure 2. Model description.

from the object to the eye of observer.

3.1. Algorithm of colour accumulation

Formula to calculate colour of pixel possible to express as follows [7]:

$$P_\lambda = \sum_{n=0}^{N} I_{\lambda n} \Omega_n \prod_{m=0}^{n-1} (1 - \Omega_m) \quad (1)$$

where P_λ is the final colour of pixel, $\tilde{\lambda}$ may be r, g or b (i.e. red, green or blue, accordingly). $I_{\lambda n}$ intensity of n-voxel, evaluated by Phong illumination model, Ω_n opacity of n-voxel. $I_{\lambda 0}$ reflected light from first point on ray, $I_{\lambda N}$ background colour and $\Omega_N = 1$. Overrunning of density threshold possible to check as follows. If on k-step overall transparency $(1-\Omega_0)(1-\Omega_1)...(1-\Omega_{k-1})$ becomes less certain ε, it means that contribution all following for k-voxel will be small and therefore scan possible to stop.

Functioning of algorithm possible to express as follows:

if (*Transparency* > ε) **and** (*last level of recursion*) **and** (*intersection*= 1)**, then**

{

$P_{\lambda i}$ += $I_{\lambda i}$ * Ω_i * *Transparency*

Transparency *= 1 - Ω_i

}

For the first step $P_\lambda = 0$, for λ = r,g,b; *Transparency* = 1.

If it is used perspective transformation, should contribute a correction in the algorithm of accumulation of. This is because size of voxel, as a result geometric primitive transformations, becomes dependent from the coordinate Z. That is to say with increase Z coordinate the size of voxel increases also. So at the each step opacity of voxel should also be recalculated with the correction for changing its length.

If replace sum with integral, intensity [8] is:

$$I = \int_0^D color(x(s))\exp\{-\int_0^s opacity(x(t))dt\}ds \quad (2)$$

D is a distance on Z coordinate, where a calculation of intensity is performed.

x(s) is a point on the length along the ray of observation.

Color and *opacity* are colour and value of opacity in a given point.

This formula more suits for our case in contrast with the formula (1), since it takes into account a length on axis z. However, it would be to calculate an integral on each length or approach with some expression. So we use an assumption, for instance that on length of segment a value of transparency constantly. In this case (1) for N elements possible to change:

$$P = 1-(1-\Omega)^N \quad (3)$$

Ω is opacity in point.

P is total opacity for N elements.

If consider N not as all amount of elementary lengths on ray, but as its length, finally:

$$P = 1-(1-\Omega)^{\Delta Z} \quad (3.1)$$

where ΔZ is a length of segment.

As already noted due to distortion of the geometric primitives a length of voxel becomes dependent on Z coordinates (Refer to 3.2. Projective transformation). This dependency possible to express as follows:

$$\Delta Z = M_{zz}\left(\frac{Z_{\max}}{M_{tz}Z_{\max}+M_{tt}} - \frac{Z_{\min}}{M_{tz}Z_{\min}+M_{tt}}\right) \quad (4)$$

it is necessary to perform transformation for coordinates of start point of segment and the end one.

M_{zz}, M_{tz}, M_{tt} are the elements of perspective transformation matrix,

$Z_{\max}, Z_{\min}$ are start and end of segment in the coordinate system of object.

Finally this part of algorithm possible to express as follows:

if (*Transparency* > ε) **and** (*intersection* = 1)**, then**

{

AccomulatedColor += *Color* * *P* * *Transparency*

Transparency *= *1 - P*

}

AccomulatedColor is the final colour of pixel,

Color is colour calculated the in point, for instance, on Phong illumination model.

Transparency is a value that is equivalent to composition from the formula (1), meaning total transparency from the beginning of ray before the given point.

P is opacity in the point calculated on the formula (3.1) where length of segment gets from the formula (4).

In the same way should point to the fact: in check is not met condition *Last recursion level*, this corresponds segments on the ray of scan can be a different length.

3.2. Projective transformation

Projective transformation extrapolates the rendering algorithm to pyramidal volumes and thereby allows generate images with perspective. In 3D space, the point with the Cartesian coordinates (x, y, z) is associated with an infinite set of homogeneous coordinates (x', y', z', a) such that x=x'/a, y=y'/a, z=z'/a i.e. the homogeneous coordinates are determined within a common nonzero factor. The transformation matrix affects the homogeneous coordinates in the following manner:

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{pmatrix} * \begin{pmatrix} x_m \\ y_m \\ z_m \\ 1 \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \\ z_p \\ a_p \end{pmatrix} \text{ or (C)(M) = (P),}$$

where (C) is the transformation matrix; (M) are the homogeneous coordinates of the point of space M; (P) are the coordinates at P corresponding by the transformation. In projective geometry proves a theorem that the projective transformation of the space M to the space P is unambiguously defined by specifying five pairs of points corresponding by the transformation, on conditions that from five points specified in

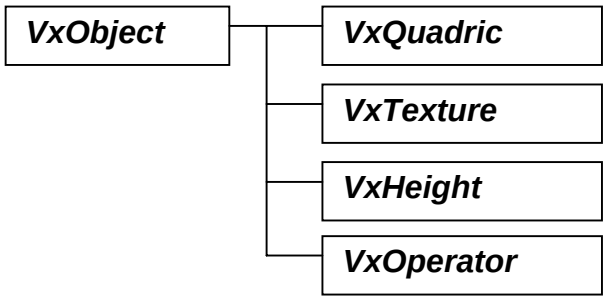

Figure 3. Hierarchy of classes.

the space M none four ones are in the same plane. Let us choose five pairs of such reference points (M^i) and (P^i) (the upper index corresponds to the number of pair) and compose the set of

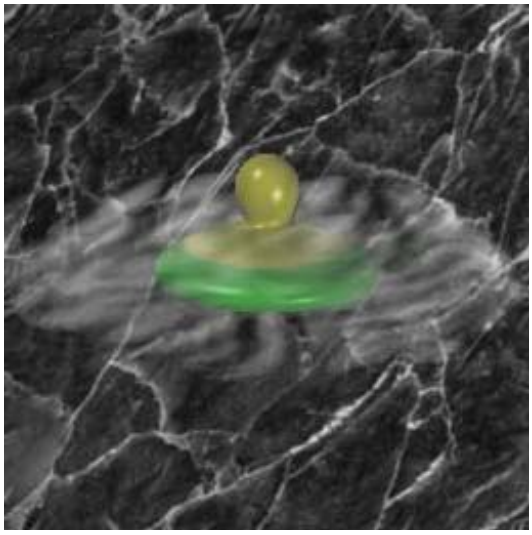

Figure 4. A scene consists of the composition of two main objects *Lid* and *Cloud*.

equations:

$$(C)(M^i) = \rho^i (P^i),$$

where i = [1,...,5], ρ^1, ρ^2, ρ^3 and ρ^4 are the unknown factors; ρ^5=1. Solving these equations, find the coefficients of the projective transformation matrix (C) used further to transform the geometric primitives.

3.3. Texture

As a new technique, authors demonstrate that it is possible to perform texture mapping on free form surfaces with perturbation functions [6]. In hierarchies of classes a class of texture VxTexture is inherited from VxObject base class (Fig. 3 Hierarchy of classes) so exists a possibility to perform manipulates with the texture to similar manipulates with other objects, for instance, rotation, shift, are realized closely as this performs with free form surfaces. Consequently there is a possibility to use such objects as the textures itself, i.e. object that used as texture is not displayed, but limits texture mapped on other object. It is necessary to note an advantage that texture is separate class, rather then is a field of already existing classes.

This does a process of texture mapping independent from objects to which it is covered. For instance, if we have only such classes of objects rendering as quadrics and quadrics with analytical perturbations, and it is required to add scalar perturbations to the system. In the first case we have to implement function for class of scalar perturbations, then to implement function of texture mapping in this class itself, but if the texture does not depend on the object to which it covered, second it is not already needed to do. For the texture also realized several methods of parameterization. By default it is considered that texture not parameterized: it is located inside cube by layers to perpendicular axis Z, in this case for the calculation of texture coordinates, X,Y are simply scaled on the size of texture. If texture rotates, it multiplies by corresponding rotation matrix. In the case of spherical parameterization, for the calculation of texture coordinates the transformation corresponding to projection on the sphere is performed.

Also a method of filtration a texture is developed. There is a possibility to apply MIP-map and perform bilinear interpolation of texture. In general, voxel gets between four texels from the given level of detail and colour in given voxel gets from these four values, which are taken with certain weights. Also the method of trilinear interpolation of texture for this approach was designed. This method of interpolation is close to others well

Figure 5. An example with cloud.

known methods [9-10]. Having been developed very well a trilinear interpolation of texture is widely used.

4. CONCLUSION

On the figure 4 shown a scene consists of the composition of two main objects *Lid* and *Cloud*. *Cloud* is defined by free form surfaces; also one texture map is applied. Value of opacity is proportional to the value of colour, *opac = (R + G + B) / 3*, where *opac* is a value of opacity; *R,G,B* components of colour.

Rendering speed of given scene unessential differs from the speed of rendering the similar scene without using a texture for *Cloud* object. On figure 5 is shown one more example with other cloud.

References

1. B. Schachter, “Computer image generation”, A Wiley-Inderscience Publication John Wiley & Sons, New York, 1980.

2. D. Rogers, “Implementing fog in Direct3D”, NVIDIA Corporation, http://www.nvidia.com/Marketing/Developer/DevRel.nsf/WhitepapersFrame?OpenPage

3. S. Dietrich, “Elevation Maps”, NVIDIA Corporation, http://www.nvidia.com/Marketing/Developer/DevRel.nsf/WhitepapersFrame?OpenPage

4. Visual Systems. & Components, Maxvue, http://www.cae.com/aerospace/visualsystems/maxvue.shtml

5. A. Castro, “POVRAY III: Design of Recursive Structures”, LinuxFocus, http://www.linuxfocus.org/English/July1998/article50.html

6. S. Vyatkin, B. Dolgovesov, A. Yesin, etc. “Voxel Volumes Volume-Oriented Visualization System” International Conference on Shape Modeling and Applications (March 1-4,1999, Aizu-Wakamatsu, Japan) IEEE Computer Society, Los Alamitos, California, 1999, P. 234.

7. G.Knittel, “Voxel Engine for Real-time Visualization and Examination”, Eurographics’93,Volume 12, number 3, p. 37-48.

8. N. Max, R. Crawfits, B. Becker, “Application of Texture Mapping to Volume and Flow Visualization”, p. 108, Proc. GRAPHICON’95, Moscow.

9. S. Paul, “Heckbert. Survey of Texture Mapping”, IEEE Comput. Graph. and Applicat.- 1986.-6, N11, P. 56.

10. L. Williams. "Pyramidal Parametrics”, Computer Graphics, 1983-12, N 4, P. 270.

Визуализация трехмерных облаков с применением поверхностей свободных форм

Предлагается реализация метода быстрого отображения полупрозрачных объектов, образованных поверхностями свободных форм. В большинстве симуляторов полета или авто-симуляторах атмосферные эффекты реализуются в виде двухмерной текстуры (облака) или просто результирующий цвет в пикселе замешивается с некоторым весом, зависящим от координаты Z с белым цветом (туман). В данной работе предлагается реализация объемных атмосферных эффектов, где есть информация о глубинной сложности сцены. В системе Voxel-Volumes реализован метод отображения полупрозрачных объектов, образованных поверхностями свободных форм. Целью данной работы является реализация рендеринга таких объектов, заполненных текстурой, для которой быстро вычисляется интегральное значение прозрачности на заданном отрезке.

Сведения об авторах:

Лаборатория Синтезирующих Систем Визуализации, ИАиЭ СО РАН. Россия 630090, Новосибирск, пр. Коптюга, 1.
телефон: (+7-3832) 33-36-30

н.с. Вяткин С.И., sivser@mail.ru,

зав. лаб. Долговесов Б.С., bsd@iae.nsk.su,

Лаборатория теоретического программирования ИСИ, СОРАН, Россия 630090, Новосибирск, пр. Лаврентьева, 6.

аспирант Есин А.В., esin@woland.it.nsc.ru