グラフィクスとCAD 109－5
（２００２．１１．２２）

一次散乱光を考慮したOpenGLベース・３Ｄテクスチャレンダリング

太田　真† 田村　真智子† 藤田　邦彦†
藤本　忠博‡ 村岡　一信¶ 千葉　則茂‡

概要

ボリュームデータなどの３Ｄテクスチャで表された雲等のレンダリングには，一次散乱光の効果を考慮したレイトレーシング型のボリュームレンダリング法が適用されることが多いが，積分計算に膨大なサンプリング計算が必要とされるため，実用的なレンダリング速度を得ることは困難である．本研究では，OpenGLによるＺバッファ法に基づく３Ｄテクスチャの高速レンダリング法を提案し，雲，稲妻，爆発火炎の高速レンダリング法への本手法の拡張法についても示す．さらに，これらの手法の有効性を映像生成例により示す．

OpenGL-based Rendering of Three-dimensional Textures Taking into Account First Order Light Scattering

Shin Ota†, Machiko Tamura†, Kunihiko Fujita†,
Tadahiro Fujimoto‡, Kazunobu Muraoka¶ and Norishige Chiba‡

Abstract

Ray-tracing type volume rendering is often applied to three-dimensional textures representing gaseous phenomena such as clouds. However, since this rendering algorithm includes time-consuming integration steps for simulating first order light scattering, it is not practical as it is. In this paper, we propose an efficient OpenGL-based rendering method, and describe how to apply it to the rendering of clouds, lightning and explosive fire, and show the effectiveness of the methods by presenting the rendered CG images.

１．　はじめに

ＣＧによる自然物や自然現象の表現では，ボリュームデータなど，ボクセルで定義される３次元テクスチャ（３Ｄテクスチャ）によるレンダリング法が用いられることが多い．雲［1,2］，火炎・煙［3］，稲妻［4,5］などのボリューム表現が適当なガス状物体のみならず，森林景観や動物の毛などのようにアンタイエイリアシングのために密度表現を行うことが効果的である場合もあり［6,7］，効率的な３Ｄテクスチャのレンダリング法を開発することは重要である．また，等方的なボリュームデータのみならず，異方性の３Ｄテクスチャのモデリング法とレンダリング法も提案されている［6,8］．３Ｄテクスチャはボクセル表現されているため解像度を低下させることも容易に可能であり，ＬＯＤ（level of detail）によるエイリアシングフリーな効率的なレンダリングが可能である．

一方，これらの現象の多くは，少なくとも１次散乱光のシミュレーションを行うことが効果的であることが示されてきており，そのシミュレーションにはこれまで，原理の単純なレイトレーシング型のレンダリング法が適用されることが多く膨大な計算時間を費やさざるを得なかった．本論文では，１次散乱光を考慮したボリュームデータによる３Ｄテクスチャのレンダリング法として，OpenGLを用いた効率的なレンダリング法を提案する．

†（株）岩手ソフトウェアセンター　‡岩手大学工学部　¶東北工業大学
†Iwate Software Center Inc.　‡Faculty of Engineering, Iwate University　¶Tohoku Institute of Technology

筆者らの一部が開発した粒子ベースの炎のシミュレーション法に対して，OpenGLベースの効率的なレンダリング法が開発されている[9]．この手法では，炎の効率的なレンダリング法として，３Ｄテクスチャによるレンダリング法に対して，炎粒子にポリゴンによる炎テクスチャをマッピングし，OpenGLによるαブレンデイングを行っている．また，筆者らの一部により，森林景観をレンダリングする目的で，レイトレーシング型の基本的な手法［7］に対して，３Ｄテクスチャから得られる複数の投影テクスチャを用いた効率的なレンダリング法も提案されている［10］．また，最近，0penGLベースの３次元医用画像（ボリュームデータ）の断層面をテクスチャとしたレンダリング法［11］や，メタボールで表現された雲の，ビルボードを用いた0penGLベースの効率的なレンダリング法などが提案されてきており［12］，ボリューム表現された対象の効率的なレンダリング法に対する研究が活発になってきている．

このような中で，本論文では，ボクセル空間として定義される３次元テクチャが３次元空間に複数分布している（マッピングされている）シーンを想定した一般的な手法を提案する．本手法は，一次散乱を考慮した３Ｄテクスチャのレンダリングを OpenGLによるＺバッファを用いることによって効率よく実行するものである. 本手法は，レイトレーシング型の３Ｄテクスチャレンダリング法に比べて，微細な表現において劣るが，少ない計算量で良好なＣＧ画像を生成することができるという利点を持つ. 以下では，まず１次散乱のシミュレーションを実現する本手法の基本的な原理について述べ，次に具体的な表現対象として，雲，稲妻，および爆発火炎をとりあげ，それぞれについて現象特有の拡張点について述べ，レンダリング例を示すことによりその有効性を示す.

２．　Ｚバッファによる高速３Ｄテクスチャレンダリング法

式(1)に１次散乱光のモデルを示す（図１参照）．このモデルは，レイトレーシング型のレンダリング法により原理的には容易に実行できる．しかしながら，ボクセル内にその解像度に応じた適当な数のサンプリングポイントをもうけなければならず，またその点から光源方向への２次レイを放射し積分計算を実行しなければなければならず，計算量の点では膨大な計算時間が要求される．

$$B=\sum_{t=t_{near}}^{t_{far}}\left\{\exp\left(-\gamma\sum_{s=t_{near}}^{t}\rho(x_s,y_s,z_s)\right)I_s(x_t,y_t,z_t)\phi(x_t,y_t,z_t)\rho(x_t,y_t,z_t)\right\} \tag{1}$$

$$I_s=I\cdot\exp\left(-\gamma\sum_{r=s}^{Lin}\rho(x_s,y_s,z_s)\right) \tag{2}$$

B：視点における輝度

γ：密度を光の減衰係数に変換する係数

Is：光源からの照度

ϕ：シェーディング関数

ρ：物体の密度

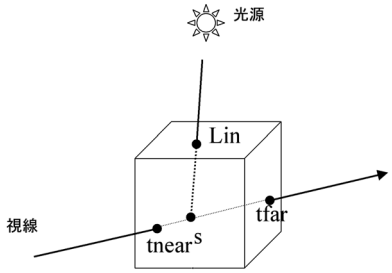

図１　１次散乱光モデル

ここでは，OpenGLの機能を用いて効率的に１次散乱光を計算する，高速な３Ｄテクスチャレンダリング法を示す.

２．１　光源減衰を考慮した３Ｄテクスチャレンダリング

レイトレーシング型の３Ｄテクスチャレンダリングにおける光源方向へのレイの放射の代わりに，ローカルシャドウバッファと呼ばれる密度を蓄積する２次元のシャドウバッファを用い計算時間を短縮する．ローカルシャドウバッファを用いたレンダリングは，バウンディングボックス（３Ｄテクスチャをワールド座標系へ配置するための直方体）ごとに行う．ローカルシャドウバッファは，図２に示すように，レンダリング対象の３Ｄテクスチャ，すなわちバウンディングボックスに重なり，かつ，光源に対して垂直方向になるように配置する．また，３Ｄテクスチャの各ボクセルは，シェーディング結果のRGB値を保存できるものとする.

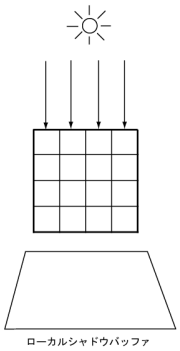

図２　ローカルシャドウバッファの配置

次に，ボクセルを光源方向でＺソートし，光源に近いボクセルから順にシャドウバッファを参照し，光源光の到達率を求めてシェーディング計算をし，その結果（RGB値）を各ボクセルに保存する．

最終的に，シェーディングしたボクセルはカメラ視点から遠い順にＺソートし，シェーディング結果のRGB値を順次，αブレンディングを用いフレームバッファへ書き込む．

本手法の処理手順を以下に示す．

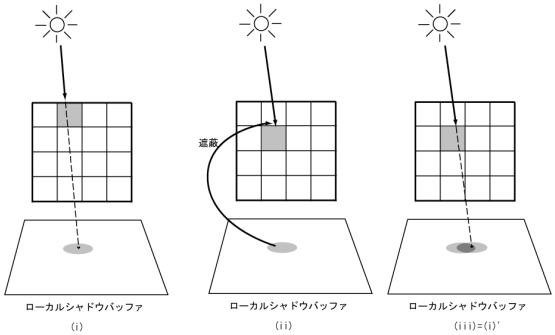

図３　ローカルシャドウバッファによる光源減衰

ステップ0　シャドウの初期化：　ローカルシャドウバッファを影の無い状態にクリアする（ローカルシャドウバッファの値は0～1の連続値で，影の無い状態は1である）．

ステップ1　ソートされた順に，各ボクセルについてステップ1-1～1-4を行う．

ステップ1-1　シャドウの取得：　ボクセルをローカルシャドウバッファに投影し，ローカルシャドウバッファ上の該当する位置の値を取り出し，それを*Tb*とする．*Tb*はボクセルへの光源光の到達率を表す．

ステップ1-2　光源光の減衰計算：　ボクセルのシェーディング計算に用いる光源からの光の強さ*Lb*を次式で求める．

$$Lb = LsTb \tag{3}$$

ここで，*Ls*は光源光の光の強さである．

ステップ1-3　シェーディング：　*Lb*を用いてボクセルを次式でシェーディングし，そのRGB値（*Cb*）をボクセルに保存する．

$$Cb = Ie + IaKd + Lb(\sigma Kd + \tau Ks) \tag{4}$$

ここで，*Kd*はボクセルの拡散反射係数，*Ks*は鏡面反射係数，*Ie*は放射光の強さ，*Ia*は環境光の強さ，係数σはボクセルが法線要素を持たない場合は0，法線要素*n*を持つ場合は

$$\max(Lv \cdot n, 0) \tag{5}$$

である．係数τは，ボクセルが法線要素を持たない場合は0，法線要素*n*を持つ場合は

$$\max\left(\frac{Vc - Lv}{|Vc - Lv|} \cdot n, 0\right)^{Shi} \tag{6}$$

であり，*Shi*はPhongのモデルの係数，*Vc*はボクセルから視点方向へのベクトル，*Lv*は光源ベクトルである．

ステップ1-4　シャドウの更新：　ボクセルの透明度を求め，ローカルシャドウバッファに乗算する．

２．２　複数光源への拡張

２．１節では光源が１つの場合について述べたが，ここでは，複数光源への拡張について述べる．

複数光源に拡張するため，式(3)～(6)を以下のように変更する．

$$Lb_i = Ls_i Tb \tag{7}$$

$$Cb = Ie + IaKd + \sum_{i=1}^{Ln}\left(Lb_i(\sigma Kd + \tau Ks)\right) \tag{8}$$

$$\max(Lv_i \cdot n, 0) \tag{9}$$

$$\max\left(\frac{Vc - Lv_i}{|Vc - Lv_i|} \cdot n, 0\right)^{Shi} \tag{10}$$

ここで，*Ln*は光源数，添字*i*は*i*番目の光源を意味する．

複数光源では，図４に示すように光源の数だけロ

ーカルシャドウバッファを用意し，シェーディングを行う．

以下に処理手順を示す．

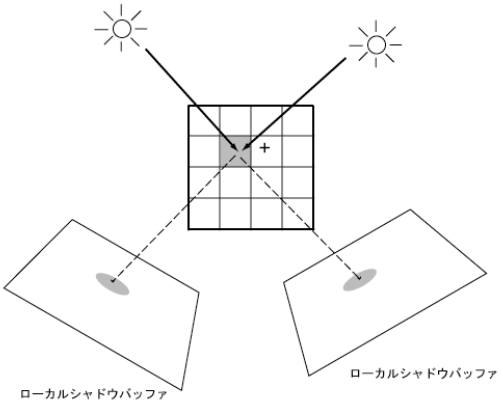

図４　複数ローカルシャドウバッファの配置

ステップ0　初期化

ステップ0-1　ボクセルの初期化：　各ボクセルに保存するRGB値を $Ie+IaKd$ で初期化する．

ステップ0-2　シャドウの初期化：　全てのローカルシャドウバッファを影の無い状態にクリアする．

ステップ１　光源の数だけステップ1-1を繰り返す．

ステップ1-1　光源方向でＺソートし，ソートされた順に，各ボクセルについてステップ1-1-1～1-1-4を行う．ローカルシャドウバッファは各光源に対応したものを用いる．

ステップ1-1-1　シャドウの取得：　ボクセルをローカルシャドウバッファに投影し，ローカルシャドウバッファ上の該当する位置の値を取り出し，ボクセルへの光源光の到達率Tbとする．

ステップ1-1-2　光源光の減衰計算：　(7)式を用いてボクセルのシェーディング計算に用いる光源からの光の強さLb_iを求める．

ステップ1-1-3　シェーディング：　(8) 式の第３項によりボクセルをシェーディングし，そのRGB値をボクセルに加算する．

ステップ1-1-4　シャドウの更新：　ボクセルの透明度を求め，ローカルシャドウバッファに乗算する．

２．３　複数オブジェクトへの拡張

ここまで述べてきた手法では，３Ｄテクスチャ同士は互いに影を落とすことができないため，不自然な画像となる場合がある．この問題は，レイトレーシング法のように物体から光源方向へレイを放射して遮蔽オブジェクトを探すことで解決可能であるが，処理時間がかかってしまう．ここでは，３Ｄテクスチャ相互の影計算を高速に行う手法として，グローバルシャドウバッファを用いる方法を示す．

グローバルシャドウバッファは，ローカルシャドウバッファと同様，光源に対し垂直方向となるように配置した２次元の濃淡画像バッファである．ローカルシャドウバッファが１つのバウンディングボックスの影を保持するのに対し，グローバルシャドウバッファはシーン上にあるすべてのバウンディングボックスの影を保持する．

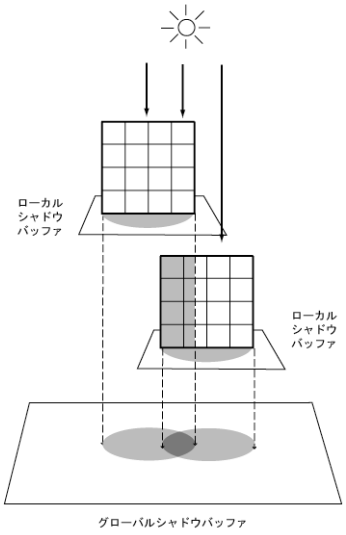

図５　グローバルシャドウバッファの配置

以下に処理手順を示す．

ステップ0　初期化

ステップ0-1　ボクセルの初期化：　各ボクセルに保存するRGB値を $Ie+IaKd$ で初期化する．

ステップ0-2　グローバルシャドウの初期化：　グローバルシャドウバッファを影の無い状態にクリアする．

ステップ0-3　バウンディングボックスのソート：　バウンディングボックスを光源方向でＺソートし，光源に近い順に番号を振り，この順で以降の処理を行う．

ステップ1　光源の数だけステップ1-1～1-3を繰り返す．

ステップ1-1　ローカルシャドウの初期化：　バウンディングボックスをグローバルシャドウバッファに投影し，その領域をローカルシャドウバッファにコピーする．

ステップ1-2　シェーディング：　バウンディングボックスを２．２節と同様の方法でシェーディングする．

ステップ1-3　グローバルシャドウの更新：　シェーディング処理で同時に更新されたローカルシャドウバッファの内容をグローバルシャドウバッファに書き出す．

３．　雲・稲妻・炎の表現への拡張

ここここでは，雲，稲妻，爆発火炎などの特定の現象を表現するための，本手法の拡張法について示す．

３．１　雲の表現法

雲のシェーディングでは，レイリー散乱とミー散乱を一次結合した位相関数による一次散乱で色を求める．また，シャドウバッファに書き込む密度は，ボクセルの密度そのものではなく，減衰曲線を用いて密度を調整したものとする［1，2］．

図６　雲のレンダリング結果

図６は，雲の形状を粒子で定義し，各々の粒子に，ノイズを加えた球状のボリュームを割り当て，雲用に拡張したシェーディング法でレンダリングした例である．粒子数，すなわち３Ｄテクスチャの数は約７万５千である．グローバルシャドウ処理により，雲に自身の影が落ちているようすが表現されている．

３．２　稲妻の表現法

稲妻のような発光物体の場合は，放射光*Ie*の値を高くし，拡散光*Kd*や鏡面光*Ks*の値を0にして，さらに，視線方向にボクセルの色（光）を加算しなければならない．このため，αブレンディングは，通常用いられる(11)式の代わりに(12)式を用いた．

$$F = As(Rs, Gs, Bs, As) + (1 - As)(Rd, Gd, Bd, Ad) \quad (11)$$

ここで，*Rs,Gs,Bs,As*はフレームバッファに書き込もうとしているRGB値とα値，*Rd,Gd,Bd,Ad*はフレームバッファのRGB値とα値，*F*は新しいフレームバッファの値である．

$$F = As(Rs, Gs, Bs, As) + (Rd, Gd, Bd, Ad) \quad (12)$$

ここで，*F*の各要素は0～1の値をとり，1を越えた場合は1とする．

図７は稲妻と雲を同時にレンダリングした例である．稲妻は，稲妻パターンを表す線分ごとに円柱状の３Ｄテクスチャを割り当てた．このように，シェーディング法が異なる３Ｄテクスチャが混在して使用されていても，レンダリングが可能である．

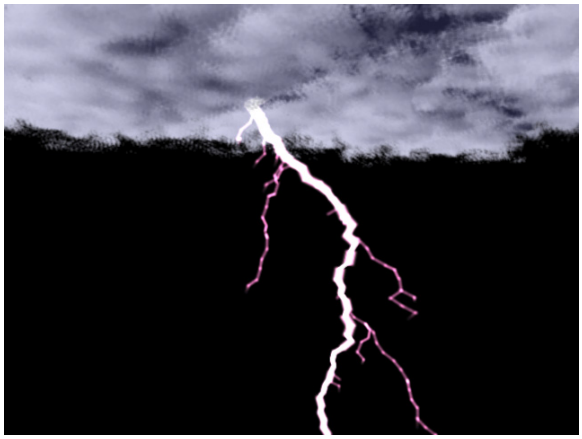

図７　稲妻のレンダリング結果

３．３　爆発火炎の表現法

炎は稲妻と同じような発光体であるが，プラズマ光でほぼ単色の稲妻に対し，炎は燃料の種類や燃焼が進行するに従って色が変化していく．これを表現するためには，稲妻のような色の単純な加算ブレンドだけでは難しい．そのため，色を各ボクセルに直接定義するのではなく，火炎のシミュレーションで得られた温度のデータから，燃料特性に合わせた色と密度を定め，炎の色の変化を表現した．

図８に温度と色の関係を示す．

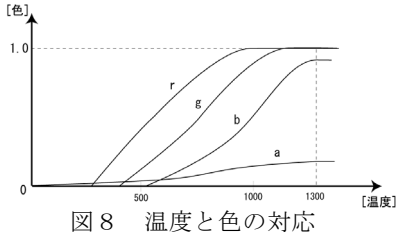

図８　温度と色の対応

図９　爆発火炎のレンダリング結果

４．　レンダリング時間

本手法とレイトレーシング法とのレンダリング時間の比較を表１に示す．レイトレーシングレンダラにはSOFTIMAGE｜3DのMental-Rayを使用した．爆発火炎については，Mental-Rayに該当する表現法が無かったため，文献[13]によるプログラムを使用した．

表１　レンダリング時間の比較

	粒子数	本手法	レイトレーシング
雲	75290	11.4秒	約2時間
稲妻	287	0.1秒	16秒
爆発火炎	13896	2.1秒	約1時間

画像サイズ：　500×424ピクセル

	粒子数	本手法	レイトレーシング
雲	75290	11.5秒	約20時間
稲妻	287	0.1秒	100秒
爆発火炎	13896	2.2秒	約10時間

画像サイズ：　1600×1200ピクセル

使用計算機はCPU PentiumⅢ 933MHz，メモリ 512MB，VGA　GeForce3 である．

本手法のレンダリング時間は，レイトレーシング法に比べ160倍から16000倍程度高速であるという結果が得られた．また，レイトレーシング法では画像サイズに応じてレンダリング時間が大きく増減するが，本手法ではほとんど変化は見られず，視点の位置を変えてもレンダリング時間に大きな差は見られなかった．

５．　まとめ

本論文では，一次散乱を考慮したOpenGLベースの高速な３Ｄテクスチャレンダリング法を提案した．提案手法では，例えば雲のレンダリングについて，従来のレンダリング法で数時間かかっていた計算時間を，数十秒程度にまで改善することができた．

なお，ここではボリュームデータにより表現される３Ｄテクスチャに限って説明したが，現在，異方性３ＤテクスチャについてもOpenGLベースのレンダリング法の開発を行っている．

６．　謝辞

本研究の一部は，通信・放送機構「地域提案型研究制度」に関わる研究開発課題による．

参考文献

[1] Y. Dobashi, T. Nishita, and T. Okita. Animation of Clouds Using Cellular Automaton, Computer Graphics and Imaging'98, pp.251-256, 1998

[2] 菊池 司，村岡 一信，千葉 則茂，積雲型の雲のビジュアルシミュレーション，画像電子学会誌，Vol.28, No.2，pp.140-151，1999

[3] J.Takahashi, H.Takahashi and N.Chiba, Image Synthesis of Flickering Scenes Including Simulated Flames, IEICE Transactions on Information Systems, Vol.E80-D, No.11, pp.1102-1108, 1997

[4] Y. Dobashi, T. Yamamoto, and T. Nishita. E.cient Rendering of Lightning Taking into Account Scattering Effects due to Clouds and Atmospheric Particles, Pacific Graphics 2001, pp.390-399, 2001

[5] B.Sosorbaram, T.Fujimoto, K.Muraoka and N.Chiba, Visual Simulation of Lightning Taking into Account Cloud Growth, Computer Graphics International, pp.89-95, 2001

[6] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures, SIGGRAPH'89, pp.271-280, 1989

[7] N. Chiba, K. Muraoka, A. Doi and J. Hosokawa, Rendering of Forest Scenery Using 3D Textures, The Journal of Visualization and Computer Animation, Vol.8, pp.191-199, 1997

[8] 村岡 一信, 千葉 則茂，異方性３Ｄテクスチャ法，画像電子学会誌，Vol.28，No.2，pp.131-139，1999

[9] 高倉章, 河合直樹，室田秀樹，茂出木敏雄，CGによる炎の3次元シミュレーション，情報処理学会「グラフィクスとCAD」研究会報告，No.065 － 002，1993

[10] 小笠原 祐治，村岡 一信，千葉 則茂：“複数の投影画像を用いた高速化３Ｄテクスチャ・レンダラ”，情報処理学会「グラフィクスとCAD」研究会報告，97-5，pp.25-30，1999

[11] 大田敬太、藤原俊朗、松田浩一、土井章男，テクスチャバッファを用いた3次元ボリューム表示システムの開発，第188回画像電子学会研究会予稿01-03-04 pp21-27，2001

[12] Y.Dobashi, K.Kaneda, H.Yamashita, T.Okita, T.Nishita, "A Simple, Efficient Method for Realistic Animation of Clouds," SIGGRAPH2000, pp. 19-28, 2000

[13] 竹下 大樹，藤本 忠博，村岡 一信，千葉 則茂，爆発火炎の粒子ベースビジュアルシミュレーション法，情報処理学会東北支部研究会，山形大学，2002.3.14