社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－CG－124 (5)
2006／8／17

頂点ベースのレイトレーシングを用いた屈折矯正結果の表示

柿本正憲† 立川智章† 西田友是††

メガネレンズ設計検証のために，近視や老視の像が矯正される様子を高速表示する手法を提案する．従来，焦点ぼけ（デフォーカス）をレンズで矯正するシミュレーションでは分散レイトレーシングを用いて計算を行っていたため，処理時間がかかっていた．本研究では，頂点ごとのレイトレーシング結果を使って環境マップを補正することにより，実用上問題ない正確さでリアルタイムの屈折表示を行う．これによってメガネを通したときの像の歪みを再現できた．さらに，視点とレンズの位置関係が固定であることを利用して，レンズ外側の空間におけるデフォーカス量の分布を前処理で計算する．この分布にしたがって実行時に物体頂点をずらして重ね合わせた表示を行うことにより，インタラクティブな性能での屈折矯正表示を実現した．

Rendering Refractive Corrected Image Using Per-Vertex Ray Tracing

Masanori Kakimoto,† Tomoaki Tatsukawa†
and Tomoyuki Nishita††

This paper describes a real-time method to simulate refraction correction of myopia or presbyopia for eyeglass design verification. Conventional defocus correction simulation was done by distribution ray tracing and thus real-time solution was impossible. Our proposed method uses per-vertex basis ray tracing to warp the environment map and acquires real-time refracted image with ray tracing-class precision. It realizes rendering of distorted image through eye-glasses. Since the relative position of the eye against the eye-glass is fixed, the proposed method precomputes the spatial distributions of defocus information in front of the eye-glass. Object vertices are displaced on the fly according to the spatial distributions to make clone objects. Interactive rate display of the defocus correction can be carried out by overlapping the clone objects.

1. はじめに

反射・屈折の表示はコンピュータグラフィクスの研究の中で重要なテーマの一つである．従来からレイトレーシングによってこれらの処理が実現されているが，処理時間が膨大である．そのため近年は，グラフィックスハードウェアの機能である環境マッピングを使ってリアルタイムで反射・屈折を近似的に表示する手法が一般的に使われている．

レイトレーシングはもともと光学系設計のための手法であり，正確な屈折をシミュレートするために現在も使われている．精度が要求される光学系の設計には，現在近似手法は使われていない．

しかし，人間がメガネレンズを通して見える様子をシミュレートする目的であれば，近似手法でも実用上問題ないことが期待できる．リアルタイム表示になることにより，設計検証の効率が上がるだけでなく，販売店での顧客への提示といった応用も可能となる．

本論文では，環境マッピングをベースに，反射・屈折物体の頂点ごとにレイトレーシングを行ってマップ画像を補正する手法を提案する．この手法を用いれば，レンズを通した像の歪みをシミュレートするのに十分な正確さでリアルタイム表示を行うことができる．

また，焦点ぼけ（デフォーカス）をモデル化するために，人間の裸眼での視力特性を，レイをサンプリングした光束として表現した．それらの光束を使って空間内のデフォーカス量の分布を求め，物体の各頂点でデフォーカス処理を行う．これによって，裸眼での見え方をシミュレートでき，レンズによる眼球での屈折の矯正もシミュレートできる．

† 日本 SGI 株式会社
SGI Japan, Ltd.
†† 東京大学大学院新領域創成科学研究科複雑理工学専攻
Department of Complexity Science and Engineering, Graduate School of Frontier Sciences, The University of Tokyo

2. 関連研究

提案手法ではリアルタイムの反射・屈折を考慮した表示を行うため，まずこの分野の関連研究をあげる．さらに，人間の視力矯正のシミュレーションに関する関連研究について紹介する．

2.1 リアルタイムの反射・屈折表示

反射を近似するために Blinn によって提案された環境マッピング[1]は，ハードウェアにより実装できるため[2)3)]，リアルタイム表示の応用に広く用いられている．環境マッピングは映りこむ物体が無限遠にあるという前提が伴うため，特に近くにある（ローカルな）物体に関して誤差が大きいという欠点があった．

近年，これを改良するための手法が考案されている．Hakura らは，ローカル物体に関しては別のレイヤーとし，個別の環境マップをレイトレーシングによる前処理で用意することでこれを解決した[4]．しかしながら，この方法では反射・屈折物体が動いてローカル物体が頻繁に変わる場合，前処理やり直しのコストが大きい．Szirmay-Kalos らは，画素ごとに奥行き値 z も持つキューブマップを求め，反射・屈折方向へのレイと物体の交点を，繰り返し計算で近似することで，ローカル物体のより正確な反射・屈折をリアルタイムで実現した[5]．この手法の目的はゲームであるため，屈折はシーンに置かれた透過物体の雰囲気を出すことが目的であった．

本研究では，レンズを通して見たシーン全体の様子を検証する目的であるため，より正確な手法が必要となる．そこで，レンズ各頂点を通してレイとシーンの交差計算を行い，レイトレーシングと同等の正確さを実現した．

2.2 視力矯正のシミュレーション

人間の眼の特性を考慮した描画手法として，被写界深度のシミュレーションは以前から行われている[6)7)]．眼球をレンズ系としてとらえ，視力の矯正まで考慮したシミュレーションモデルは Mostafawy らによって提案された[8]．Loos らは，波動光学的なアプローチである wavefront tracing を活用し，累進焦点レンズ（境目のない遠近両用レンズ）による視力矯正のシミュレーションを行った[9]．

これらの従来手法はいずれも分散レイトレーシングを使って処理を行っているため，表示速度は遅くリアルタイム処理は不可能である．

3. 頂点ベースのレイトレーシング

本節では，レンズによる歪みを正確に表示するための屈折処理に用いる頂点ベースのレイトレーシングについて説明する．なお，本研究では，対象となる物体はすべてポリゴンモデルとしている．

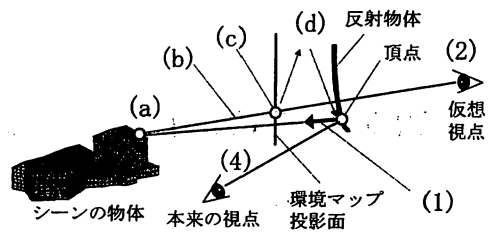


図 1 頂点ベースのレイトレーシングの処理手順

3.1 アルゴリズム

毎フレームの処理手順は以下のとおりで，図 1 と対応させて説明する．

(1) 反射・屈折物体表面の頂点での反射・屈折ベクトルをすべて求める
(2) 仮想視点と視野を求め，環境マップを描画する
(3) 反射・屈折物体の各頂点について
  (a) 反射・屈折ベクトルの延長がシーンと交差する点を求める
  (b) 上記交点と仮想視点を結ぶ直線を求める
  (c) 上記直線と環境マップ投影面との交点を求める
  (d) 上記交点の環境マップ画像上での座標を頂点のテクスチャ座標とする
(4) 本来の視点から反射・屈折物体とシーン全体を描画する

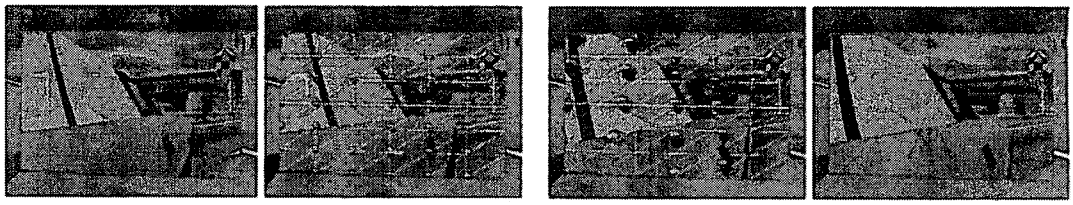
上記 (2) の仮想視点は，反射・屈折ベクトルをいくつか選び，それらの延長線に最も近くなる点として求める．

本アルゴリズムでは，反射・屈折物体の各頂点について正確な位置が映りこむことが基本的に保証される．例外として，頂点と仮想視点のずれに起因して遮蔽物体の食い違いが起こる場合がある．これは反射・屈折物体が大きく，かつレイの広がりが極端に大きい場合にほぼ限られ，レンズのシミュレーションでは問題にならない．

図 1 では反射の例について示しているが，レンズのような屈折物体の場合は，入射と出射の 2 回の屈折を経たあとのレイを求めて同様の手順を実行する．

3.2 反射と屈折の表示結果

頂点ベースのレイトレーシングの反射の表示例を図 2 に示す．キューブマッピングではずれが生じるローカル物体の映りこみも，頂点ベースのレイトレーシングでは正確に処理されていることがわかる．



頂点ベースのレイトレーシング　　キューブマッピング

図 2　車のドアミラーへの映りこみの例．真ん中の二つの画像では，ミラーのワイヤーフレームモデルを重ね合わせている．多数の球は反射ベクトルとシーンとの交点に配置した．頂点ベースのレイトレーシングでは交点が正確に頂点部分に映りこんでいることがわかる．

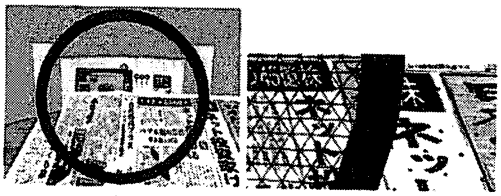


図 3　凹レンズを通した屈折の表示結果．右図は左図の右端付近を拡大したもので，レンズ頂点をワイヤーフレームで表示し，屈折ベクトルとシーンの交点に小さな球を配置した様子．屈折ベクトルの交点が頂点に正確に映りこんでいる．

同様に，レンズを通した屈折の表示例を図 3 に示す．

4. デフォーカスの補正

本節では，デフォーカスのモデル化とメガネレンズによる矯正後のデフォーカスを考慮した表示処理について述べる．提案手法では幾何光学を利用した簡易モデルを用い，波動光学の要素は考慮しない．

4.1 視覚モデル

まず，裸眼でのデフォーカスについて考えてみる．眼光学では，最も近くに焦点を合わせたとき，眼球からその点までの距離を調節近点（あるいは単に近点），逆に最も遠くに合わせたときの距離を調節遠点（遠点）と呼ぶ．近点が大きい人ほど老視（老眼）が強く，遠点が小さい人ほど近視が強いことになる．

図 4 は，提案手法で用いる簡易モデルである．眼球の屈折は本来角膜と水晶体の 2ヶ所で起こるが，ここではこれらを一つの薄いレンズとみなす（図 4(a)）．近点に合わせた状態で近点よりもさらに近くの被写体を見た場合にデフォーカスが起こる．瞳の開口の直径を A，瞳から近点までの距離を n とし，近点よりもさらに z_n だけ内側に点状の被写体があったとする（図 4(b)）．この被写体を見ようとしても，開口からのレイは一点に収束せず，被写体を中心として視線に垂直な直径 D_n の円盤状に拡がる．ここで

$$D_n = \frac{Az_n}{n} \tag{1}$$

である．提案手法では，この直径 D_n の円盤を点拡がり関数（Point Spread Function）とみなし，被写体を円盤内に多数サンプリング配置したのちに通常の CG のカメラモデルで描画することによりデフォーカスを実現する．各サンプルの描画結果はアキュムレーションバッファ[7] を使って合成する．

図 4(c) に示すように，遠点よりさらに遠くの被写体を見たときにもデフォーカスが起こるが，このときも被写体を直径

$$D_f = \frac{Az_f}{f} \tag{2}$$

の円盤内でサンプリング配置することによりデフォーカスをシミュレートする．

4.2 頂点の変位処理

実際の被写体であるポリゴンモデルに対しては次のように考える．まず，サンプリング数 S を決め，すべての与えられたモデルについてクローンを作る．各クローンにはクローン番号 i（$i = 0, 1, ..., S-1$）がつく．便宜上，もとのモデルのクローン番号は 0 とす

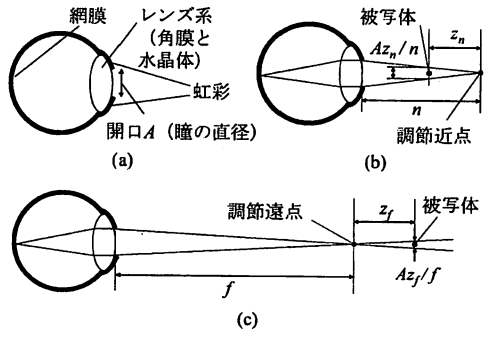


図 4　眼球によるデフォーカスの簡易モデル．(a):簡易眼球モデル．(b):限界まで近く（調節近点）に焦点を合わせた状態．(c):限界まで遠く（調節遠点）に焦点を合わせた状態．

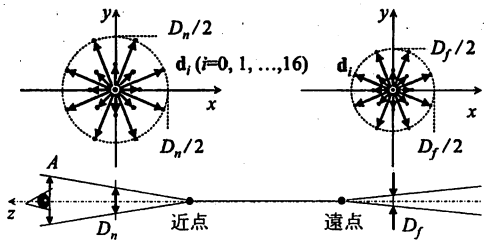


図 5　視点からの距離によって決まる頂点変位ベクトルの例.

る．表示処理の際には，S 個のクローンすべてを表示するが，全頂点について，視点からの距離に応じて，式 (1) または式 (2) の範囲内で視線と垂直方向に変位させる．頂点が近点と遠点の間にある場合は変位は 0 となる．

変位の方向と大きさは，あらかじめ定めた S 個の変位ベクトル $\mathbf{d}_i$ $(i = 0, 1, ..., S-1)$ からなる変位ベクトルセットのうち，その頂点の属するモデルのクローン番号にしたがって選択する．図 5 に変位ベクトルの例を示す．図では近点の内側と遠点の外側とについてそれぞれ一つずつの変位ベクトルセット $(S = 17)$ を例示している．

実装時は，変位ベクトルは正規化したものを 1 セットだけ保持し，頂点位置での D_n または D_f の値に応じてその場で大きさを決める．近点と遠点の間では，変位は 0 なので，すべての変位ベクトルが $\mathbf{0}$ であるような変位ベクトルセットとする．これは，焦点が完全に合わせられ，デフォーカスがまったくないことを意味する．

4.3　デフォーカス量の空間分布

前節の方法では，頂点が決まれば視点からの距離が決まり，変位量を式 (1) または式 (2) によって計算して変位ベクトルを求めた．裸眼の視覚モデルとしてはこれでよいが，レンズを通した場合は各頂点について計算で変位ベクトルを求めるのは困難である．そこで，図 6 のように立方体のボクセルによる空間分割を行い，各ボクセルについて変位ベクトルセットを前計算する方法を提案する．

ここでは，眼とメガネレンズの位置関係が固定という前提条件を用いる．また，ボクセル空間はシーン全体を包含するのに十分な大きさで設定する．

前計算の概略は以下の通りである．まず全ボクセルの変位ベクトルを初期化したあと，近点に合わせた状態での変位ベクトルを，次に遠点に合わせた状態での変位ベクトルを各ボクセルに記録していく．

一回の変位ベクトルの記録の処理は次のようになる．

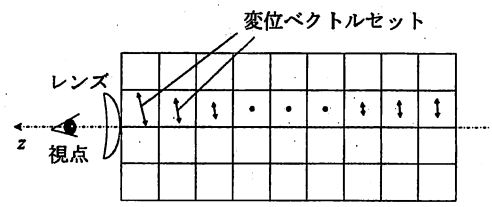


図 6　デフォーカス量（変位ベクトル）の空間分布の例.

視点からレンズ面にある一点に対して近点用または遠点用のレイの束（光束）を発し，その束が通るボクセルにおけるレイの拡がりを変位ベクトルセットとして蓄積する．この処理を，レンズ面上の点を一定間隔で走査する形で光束の生成と処理を繰り返す．各ボクセル側から見ると光束が複数回通ることになるが，複数回蓄積した変位ベクトルセットから最後に平均の変位ベクトルセットを計算する．

以下，近点用の光束を処理する方法について詳細に説明する（図 7）．

まず，視点からレンズ上の与えられた一点に対して光線を生成する．この光線を中心として光束を生成するため，これを基準光線とよぶことにする．光束の中の他の光線の開始点は視点の近傍に設ける．この開始点は瞳孔の表面に対応するもので，視点を中心とし基準光線に垂直な直径 A（4.1 節で述べた瞳の開口直径）の円盤内でほぼ一様に分布させる．

次に，基準光線上の近点を求め，基準光線以外の光線もすべてそこを通るようにする．結果として，光束は近点で収束する細長い円錐状の形となる．

さらに，基準光線を含む各光線についてレンズで 2 回屈折を行い，屈折後の光束が最も収束する点を求め，これを矯正近点とする．矯正近点よりも視点に近い場所について，屈折後の基準光線が通るボクセルをたど

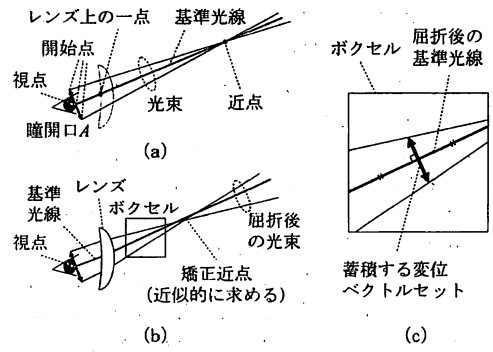


図 7　変位ベクトルセットのボクセルへの蓄積処理．(a)：光束の生成 (b)：屈折後の光束 (c)：屈折後の基準光束が通るボクセルに蓄積する変位ベクトルセットの計算

り[10]，そのボクセルにおける屈折後の光束の拡がりを，変位ベクトルセットとして蓄積する．具体的には，ボクセルで切り取られた基準光線（屈折後）の中点を求め，その中点を通り基準光線に垂直な平面を求める．屈折後の光束の各光線がその垂直平面に交わる点に，前記中点から引いたベクトル群を変位ベクトルセットとする（図 7(c) 参照）．

以上の処理をレンズ上の各点（通常は頂点でよい）について繰り返す．遠点に関しても同様の処理を行い，最後に各ボクセルについて蓄積した変位ベクトルから平均を求め，前処理を終了する．

表示処理の実行時には，すべてのクローンモデルの頂点について次のような手順を行う．

まず，その頂点の属するボクセルを求める．これは一回の座標変換によって求まる．次に，そのボクセルに記録された変位ベクトルセットから，頂点のクローン番号に応じて変位ベクトルを選び，4.2 節で述べたのと同様に実際に頂点を変位させる．実際の表示は，4.1 節の後半で述べたように各クローンモデルの描画結果を合成する．

5. 実験結果

本提案手法によりデフォーカス処理を施し，レンズによる矯正も考慮して描画した結果を図 8 に示す．

本実験では，CPU として Pentium4 の 3.8GHz，GPU として GeForce 7800GTX を使用し，表示性能については以下のような結果を得た．

まず，図 3（左）のように，レンズによる屈折処理だけを行いデフォーカスをしない場合は，30〜40fps で表示できる．これは眼とレンズの相対位置を固定にした場合（屈折ベクトルの再生成が不要）で，この前提を解除すると表示レートは 6fps に落ちる．使用したレンズのポリゴン数は表面と裏面それぞれ約 1,500，シーンのポリゴン数は 10,000 程度である．

次に，デフォーカス処理については，約 1,500 の光束（光線数 10）を屈折させ $110 \times 77 \times 42$ のボクセルについてサンプル数 10 の変位ベクトルセットを前計算するのに約 3 秒かかった．デフォーカス矯正の表示性能（図 8(c)(d)）は 12〜15fps であった．レンズを使わない裸眼でのデフォーカスの表示性能（図 8(a)(b)）は 30fps である．

6. 考　　察

6.1 モデルデータの前処理

提案手法の最も大きな制限は，表示対象物体をボクセル間隔程度に頂点が配置されるぐらいの細かさでポリゴン化しておく必要がある，という点である．たとえ一枚の四辺形であっても，大きなものであれば細かくメッシュ化する必要がある．そのため，データの前処理の手間がかかることは確かである．

しかし，表示性能への影響についていえば，細かいメッシュ化が必要なのは比較的視点に近い部分なので，広域データを扱う場合は LOD 処理を使えばポリゴン増加による性能低下はほとんどないと考える．

6.2 視線に対する考え方

人間の視覚は，ある瞬間を考えると，広い視野のうち視線方向の限られた範囲だけに高い視力を集中し，周囲の視力は極端に低い．本研究のように，人間の眼球特性まで考慮した描画を行う場合でも，このような視線を考慮した処理は二つの理由で困難である．一つは，広い視野の視力分布のモデル化が困難である点で，もう一つは，結果の提示の際，実際のユーザの視線を検出してそれに即応した表示を行う難しさがともなう点である．

本研究では，描画すべき画像の各部分（各ボクセル）についてそれぞれ個別に最良の調節を行った場合の結果を集積して出力画像とし，広域の視力分布は無視している．このように，出力画像の各部分について入力条件が異なるのは一見奇異に思える．しかし，最終的に結果の画像を見るユーザも，その画像のごく一部にしか高い視力を発揮できないから，画像全体を同一の条件でシミュレートすることはあまり意味がない．少なくとも，メガネレンズによる矯正結果の検証の応用では，各部分で最良の調節を使って描画することが有用である．

7. おわりに

レイトレーシングと実用上同じ精度の反射・屈折をリアルタイム処理で実現し，従来分散レイトレーシングで多くの時間をかけていた屈折矯正のシミュレーションもインタラクティブな表示レートで実現した．提案手法は，その精度の高さにより，実際のメガネレンズ設計検証に十分活用できる見通しを得ている．

今後は，累進焦点レンズなど，検証が重要となる実用的なレンズの設計に適用していくとともに，光束の生成時に乱視を考慮したり屈折時にレンズの色収差を考慮するなど，より現実に近いモデルを確立したい．また，空間分割を視点座標系で行うことにより，特に視点の近くでのデフォーカス画像の画質向上が見込める．デフォーカス処理における頂点のクローン化処理は頂点シェーダによって効率化できる可能性があり，これも今後の課題である．

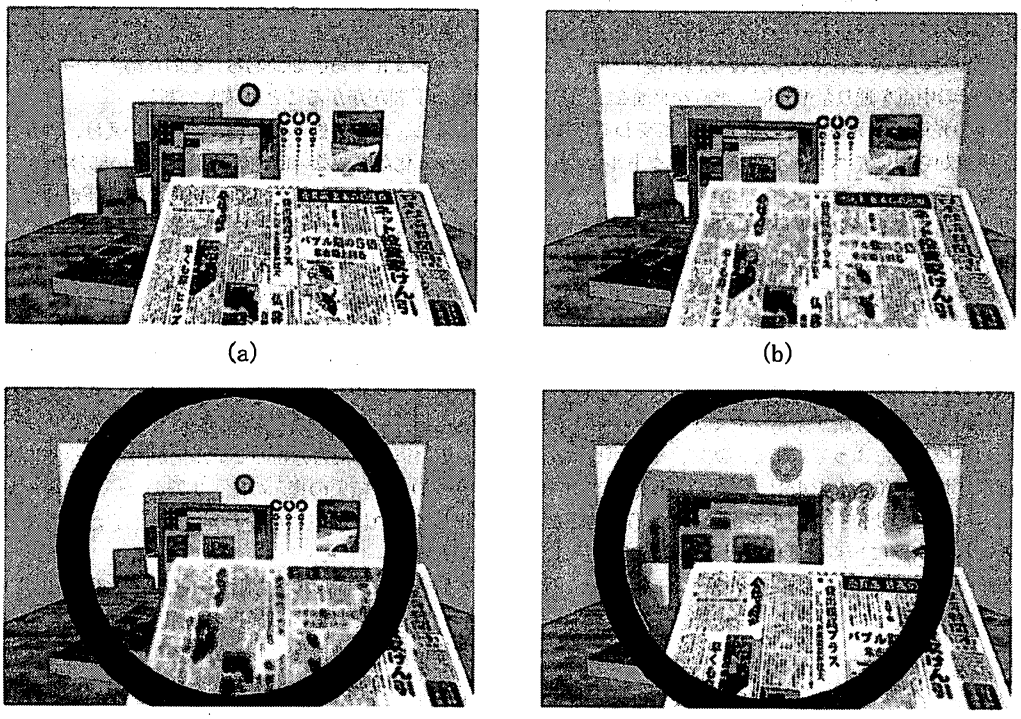


図 8 視力を考慮したデフォーカスの表示とその矯正. (a)：正視の画像 (近点 0.2m, 遠点 10m). (b)：近視かつ老視の画像 (近点 0.5m, 遠点 1.3m). (c)：上記 (b) を近視用の凹レンズで矯正した結果. (d)：上記 (b) を老視用の凸レンズで矯正した結果. いずれも, 瞳孔径は $A = 0.012$m, ボクセル数 $110 \times 77 \times 42$, デフォーカスのサンプル数 $S = 10$, 屈折率 1.66 (ガラス).

参 考 文 献

1) J.F. Blinn and M.E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, Vol. 19, No. 10, pp. 542–547, 1976.
2) P.Haeberli and M.Segal. Texture mapping as A fundamental drawing primitive. In *Fourth Eurographics Workshop on Rendering*, pp. 259–266, 1993.
3) D. Voorhies and J. Foran. Reflection vector shading hardware. In *Proc. SIGGRAPH '94*, pp. 163–166, 1994.
4) Z.S. Hakura, J.M. Snyder, and J.E. Lengyel. Parameterized environment maps. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 203–208, 2001.
5) L. Szirmay-Kalos, B. Aszodi, I. Lazanyi, and M. Premecz. Approximate Ray-Tracing on the GPU with Distance Impostors. *Computer Graphics Forum (Proc. Eurographics 2005)*, Vol.24, No.3, pp. 685–704, 2005.
6) R.L. Cook, T.Porter, and L.Carpenter. Distributed Ray Tracing. In *Proc. SIGGRAPH '84*, pp. 137–146, 1984.
7) P.Haeberli and K.Akeley. The accumulation buffer: hardware support for high-quality rendering. In *Proc. SIGGRAPH '90*, pp. 309–318, 1990.
8) S.Mostafawy, O.Kermani, and H.Lubatschowski. Virtual Eye: Retinal Image Visualization of the Human Eye. *IEEE CG&A*, Vol.17, No.1, pp. 8–12, January-February 1997.
9) J.Loos, Ph. Slusallek, and H.-P. Seidel. Using wavefront tracing for the visualization and optimization of progressive lenses. *Computer Graphics Forum (Proc. Eurographics 1998)*, Vol.17, No.3, pp. 255–263, 1998.
10) J.Amanatides and A.Woo. A fast voxel traversal algorithm for ray tracing. In *Proc. Eurographics 1987*, pp. 3–10, 1987.