NICOGRAPH 2012, pp. 79-86

マイクロポイント ベースト シャドウ

高桑昌男 [1)]（非会員）　　**飯倉宏治** [2)]（正会員）

1）フリーランス　　2）静岡理工科大学総合情報学部

MicroPoint Based Shadow

Masao Takakuwa[1)](Non-Member)　**Koji Iigura**[2)](Member)

1) Freelance

2) Faculty of Comprehensive Informatics, Shizuoka Institute of Science and Technology

tacwon @ gmail.com
iigura @ cs.sist.ac.jp

アブストラクト

現代レンダリングに分類されるマイクロポイント法では、高周波成分を低周波成分に正規化することで、エイリアシングのない画像をスーパーサンプリングなしに生成できることが立証されている。同時に、アルゴリズムの単純さ、マイクロポリゴン法およびポイントクラウド法との親和性から、マイクロポイント法が CG 表現のあらゆる領域に拡張可能であることが示唆されている。本研究では、従来のシャドウマップ法が持つ影付け処理の問題点を取上げ、マイクロポイントによる解決方法を提案する。また同時に、提案手法を用いることで、スーパーサンプリングやレイトレーシングを用いる事なく、マイクロポイント法による滑らかで鮮明な陰影を得ることができる事を示す。

Abstract

Micro Point - it will be classified into modern rendering technique, and this new rendering technique generates aliasing-free images without super sampling by normalization a high frequency component into a low frequency component. Due to Micro Point algorithm’s simplicity and affinity for micro-polygon and point cloud, Micro Point is expected to give us a lot of novel aspects in computer generated image area. In this paper, we discuss the shadowing process by existing methods, and we propose a new shadowing algorithm: Micro Point Based Shadow. By using our new algorithm, we also show Micro Point method generates images with clear and sharp shadow and shade.

1. はじめに

ポイントサンプリングに基づくシャドウマップ[1]およびシャドウマップから派生する各種手法[2,3,4] では、一般にカメラスクリーン上のピクセルとシャドウマップ上のピクセルが一対一対応しない。そのため、以下のような大半のケースにおいて正確な影付けを行う事ができない：

（A）カメラスクリーン上の１ピクセルがシャドウマップ上の複数ピクセルと対応する場合。
本来は、シャドウマップ上の複数ピクセルを考慮し、影付けが行わねばならないが、シャドウマップ上の１ピクセルの情報により影付けされる。

（B）シャドウマップ上の１ピクセルがカメラスクリーン上の複数ピクセルと対応する場合。
本来は、カメラスクリーン上の複数ピクセルに対し寄与率に応じ影が投影されなければならないが、カメラスクリーン上の複数ピクセルがシャドウマップの１ピクセルだけで代表されるため正確な影付けが行われない。

仮に、カメラスクリーン解像度を上げると（A）現象が軽減される一方、（B）現象が増加する。同様に、シャドウマップ解像度を上げると、（B）現象が軽減され、（A）現象が増加する。（A）（B）現象は、逆相関関係にあり、シャドウマップ法におけるエイリアシング回避は、原理的に不可能である。
そのため従来手法では、マルチサンプリングなどの近似により、影の境界部分で発生するアーティファクトを軽減する（アーティファクトは、光源の種類によらない）。
こうした近似による影付けは不正確で、ぼやけた影境界を生成するだけである 。本論文では、マイクロポイント[5]を用いた影付け手法を提案する。本手法により、（A）および（B）のケースについても正確で、鮮明な影を生成できる。

2. 関連研究

2.1 シャドウマップ

前述のとおり、シャドウマップ[1] による影表現では、カメラ座標系から見えるオブジェクト全てに対し、正確な光が投影される保証がない。発生するアーティファクトを軽減する手法として、Perspective Shadow Map[2], 光源座標の最適化[3]、カスケード法[4]、深度バイアス法[4]、オクルーダ近似法[4]などが提案されているが、いずれの手法においても、本質的に、正確で、鮮明な影境界を生成することは不可能である。

2.2 レイトレーシング

レイトレーシングの影表現では、オブジェクト形状に対し、スーパーサンプリングが行われる[6]。形状の複雑さは統計的に推測されるが、推測が正しいという保証はない[5]。

2.3 ステンシルバッファ

ステンシルバッファを用いた影付方法は、ゲーム等のリアルタイムレンダリングにて用いられる[7,8]。この方法は原理的には形状の高周波成分を正しく取扱う事ができるが、既存の手法では、ステンシル情報を格納するステンシルバッファで、これらの情報を取りこぼしてしまう。

3. マイクロポイント

マイクロポイントに関しては既に発表済み[5]であるが、新しい概念のため、本論文においても再度説明を行う。
ポイントサンプリングに基づく手法はスーパーサンプリングに依拠し発展してきた[9,10,11]。そしてスーパーサンプリングが、所詮、統計的手法に過ぎないことは暗黙の了解事項とし黙殺され、これまで踏込んだ議論がなされることは殆どなかった。
マイクロポイントでは高周波成分を低周波成分に正規化することでスーパーサンプリングなしにエイリアシングのない画像を生成する事が可能である。

3.1 基本概念

不確定性原理の基では、プランク長以下の領域で、運動量と位置を同時に決定できない。このことが、量子力学の根本原理となっている。不確定性原理のプランク長をピクセル領域に、運動量を不透明度、位置を占有率に置換えたのがマイクロポイントの基礎概念である。
ピクセルを観測する際、観測者はピクセル内部で何が起きているのかを知ることはできない。観測者が知り得るのは、ピクセル内に存在するオブジェクトの色および不透明度(=α)と専有面積との積のみである（以下の例題を参照）。

> 問：　背景が RGB=(0,0,0) の時に RGB=(128,0,0) なる値がピクセル値として観測された。この時、ピクセル内部にはどのようなオブジェクトが存在するのか以下の選択肢より選べ。
>
> （1）占有率 100% の RGB=(128,0,0) の不透明オブジェクト
>
> （2）占有率 100% の RGB=(255,0,0) の α＝0.5 のオブジェクト
>
> （3）占有率 50% の RGB=(255,0,0) の不透明オブジェクト
>
> （4）占有率 10% の RGB=(1280,0,0) の高輝度な球形オブジェクト
>
> 答：　すべて正解

例題：ピクセルの色とは何か？

マイロポイントにおいては、ピクセル領域内で、不透明度=0.1、占有率=50% の三角形と、不透明度=1.0、占有率=5%の球体は、同一のものとして扱われる。
ピクセルの観測値からピクセル内部でどのような事が起きているか分からないのであれば、観測者の都合で、解釈しなおせば良いーこれがマイクロポイントの基本概念である。
併せて、従来のレンダリング手法においては、輝度、法線ベクトル、接ベクトル、u,、v、du、dvなどの値は、オブジェクト形状の特徴を示す従属値として定義されるが、マイクロポイントにおいては、ピクセル領域内のオブジェクト形状は不確定として扱わず、ピクセル領域内の独立した分散確率として、輝度、

法線ベクトル、u、v値などを定義する。

3.2 正規化

上述のとおり、観測者に都合の良い様に、ピクセルの観測値を解釈しなおす。これがマイクロポイントを用いた高周波成分の低周波成分化である。

ピクセル内部に存在するオブジェクトの占有率を Coverage、透明度を Alpha、色を Color とすれば、ピクセルの観測値は Coverage × Alpha × Color と表される。この式を次のように変形する：

$$\text{ピクセルの観測値} = \text{Coverage} \times \text{Alpha} \times \text{Color} = \underbrace{1.0}_{\text{正規化された占有率}} \times \underbrace{(\text{Coverage} \times \text{Alpha})}_{\text{正規化された不透明度}} \times \text{Color}$$

マイクロポイントでは、不透明度の値を変化させることで常に占有率 ＝100% になるような正規化が行われる。不透明度 ＝1.0、占有率 ＝5% の高周波成分は、不透明度 ＝0.05、占有率 ＝100%と低周波成分化される。結果、原理的にサンプリングミスが発生することがなくスーパーサンプリングが不要となる。このスキームはテクスチャマッピングにも拡張され、エイリアシングの原因となるピクセルに対する高周波成分を持つテクスチャについても、それぞれのテクセルのピクセルへの占有率として同様に扱われる。

3.3 ダイシング

マイクロポイントへのダイシングは形状および法線、テクスチャ座標等の分散に応じて行われる。例えばテクスチャ座標に関しては､uv 座標の分散が１テクセル以内に収まるようダイシングされる。その為、テクスチャサンプリング時におけるテクスチャフィルタリングは全く不要となり、テクスチャサンプリング時にテクスチャ情報がぼやけるといった現象は発生しない。

3.4 サンプリング定理

サンプリング定理に基づき、マイクロポイントは、ピクセルを2×2に分割したサブピクセル領域で定義される（実際には縦横２倍の解像度でレンダリングを行い、それを縮小する事となるので、以下、ピクセルとサブピクセルの区別は行わず、単に「ピクセル」と呼ぶ事とする）。

4. 提案手法 － MicroPoint Based Shadow

4.1 基本となる考え方

最終画像に直接関連するマイクロポイントはカメラ座標系で定義されたスクリーン解像度に応じ生成される。マイクロポイントは生成されると形状を持たない概念となるが、生成前までは形状情報を保持している。

この形状情報をオブジェクト片とし光源座標系で定義されたスクリーン(ライトスクリーン)でマイクロポイント化を行う。これらライトスクリーン上のマイクロポイントをシャドウマイクロポイントと呼ぶ。

シャドウマイクロポイントはマイクロポイントへのリンクを保持し、該当マイクロポイントが光源に照らされているか否かを判定するために利用される。

4.2 レンダリング手順

マイクロポイントベーストシャドウによる影付け方法は以下の手順にて行う。

4.2.1 ダイシング

オブジェクトに付随する分散情報（輝度、法線、U,V・・・）を用いオブジェクトをオブジェクト片へとダイシングを行う。3.3 で述べたように、ダイシングによって各種の分散確率が十分に小さくなるよう行われる。その結果、mipmap[12]や異方性フィルタ[13]のような恣意的手法も不要となる。

4.2.2 マイクロポイント用微小オブジェクト片の生成

オブジェクト片を視点座標系に変換し、スクリーン格子を用いてさらに小さなオブジェクト片を生成する（詳細については付録を参照）。この生成した小さなオブジェクト片＝微小オブジェクト片はピクセル(サブピクセル)に収まる大きさであり、１つの微小オブジェクト片より１個のマイクロポイントが生成される大きさである。

4.2.3 マイクロポイントの生成

生成した微小オブジェクト片よりマイクロポイントを生成する。この時、生成したマイクロポイントと微小オブジェクト片とを紐づけしておく。

マイクロポイントの生成時、画像生成のために必要なシェーディング関連情報や奥行情報の最大値および最小値を各マイクロポイントの属性値として保持する。また、それぞれのマイクロポイントにおける、各光源からの光が

”当たっている“、”当たっていない“

を表す記憶領域を合わせて初期化しておく。

4.2.4 シャドウマイクロポイントの生成

光源から見た形状情報を格納するスクリーンをライトスクリーンと呼び、ライトスクリーンに格納される影付け用のマイクロポイントをシャドウマイクロポイントと呼ぶ。

4.2.2 で生成した微小オブジェクト片を光源座標系に変換し、ライトスクリーン用にダイシングしマイクロポイントを生成する。シャドウマイクロポイント生成時に微小オブジェクト片に紐づけられたマイクロポイントの情報をリンクとして設定しておく。このリンクは次の照光処理で使用する。

このリンク情報により、以下の事柄が実現される事に注意：

（A’）ライトスクリーン上の複数のシャドウマイクロポイントとカメラスクリーン上のマイクロポイントとが紐づけられる。

（B’）ライトスクリーン上の各格子（＝ライトスクリーンにおけるピクセルに相当する概念）には複数のシャドウマイクロポイントが格納される。またそれらシャドウマイクロポイントはカメラスクリーン上のマイクロポイントと紐づけられている為、結果として、ライトスクリーンの１格子とカメラスクリーン上の複数のピクセルとを関連付けることが可能となる。

4.2.5 ライトスクリーンによる照光

ライトスクリーンの各格子にはシャドウマイクロポイントが格納されている。このシャドウマイクロポイントを光源に近い方から順次列挙していき、リンク先のマイクロポイントに照光されているという情報を設定していく。この処理をシャドウマ

イクロポイントの不透明度の累計が 1 になるまで続ける。
ライトスクリーン全体に上記の処理を行う事により、各マイクロポイントに対し、各光源からの光が当たっている・いないという情報が設定される。

4.2.6 マイクロポイント列から色およびα値を決定

最後にスクリーンに格納されているマイクロポイント列より最終的な画像の色およびα値を決定する。
一般的に各ピクセルには複数のマイクロポイントが格納される。奥行情報の範囲が重複するものは1つのマイクロポイントへと合成を行い、手前から奥へと、マイクロポイントの不透明度情報を用いて半透明合成を行う。
それぞれのマイクロポイントが影になっているか否かで色が変わるが、それらの処理はシェーダにて行う。シェーダでは該当マイクロポイントが照光されているか否かを判断して、適切な色を返すようにする。

5. 結果

マイクロポイントベーストシャドウにより影付けされた画像を以下に示す（図 1、図 2）：

図 1 Stanford “Happy Buddha”

図 2 Stanford “Bunny”

6. まとめと今後の課題

本研究で提案したマイクロポイントベーストシャドウを用いる事で、冒頭で述べた（A）および（B）のケースついても正しく影付けを行える事を示した。また合わせてマイクロポイント法で影の付いた画像を生成できることを示した。
シャドウマイクロポイントとライトスクリーンによる照光処理では、カメラスクリーンにおける被照光対象領域がライトスクリーン上でどのような広がりを持つのかを考慮して照光処理が行われる。一方、ディープシャドウマップ[14]を含む従来のシャドウマップでは、あくまでも点対点の関係のみであり、この部分は従来の手法と大きく異なる点である。
本研究では点光源の影に限定しライトスクリーンを用い実装および検証を行ったが、本手法は全方位の点光源や線光源、面光源によるソフトシャドウ、反射/屈折の効果を考慮したコースティクス表現にも拡張可能と考えられる。
オブジェクト片、マイクロポイント用微小オブジェクト片、シャドウマイクロポイントの各生成処理は並列性が高い。実際、本研究の為に作成したレンダラでは、一部の処理に関しては既に並列化を行っており、複数 CPU コアの活用を実現している。Out of core 化とあわせて、今後さらなる並列化を行う予定である。

参考文献

[1] L. Williams, Casting curved shadows on curved surfaces, Proc. SIGGRAPH, Vol.12, No.3, pp.270-274, 1978.
[2] M. Stamminger, G. Drettakis, Perspective shadow maps, ACM TOG - Proc. SIGGRAPH, Vol. 21, Issue 3, pp.557-562, 2002.
[3] M. Wimmer, D. Scherzer, W. Purgathofer, Light Space Perspective Shadow Maps, Proc. Eurograpihcs Symposium on Rendering, pp.143-152, 2004.
[4] E. Eisemann, U. Assarsson, M. Schearz, M. Wimmer, Casting Shadows in Real Time, SIGGRAPH Asia 2009 Course Notes, 2009.

[5] 高桑、飯倉、マイクロポイント、映像情報メディア学会 2012 年次大会予稿集, 2012.
[6] R. Cook, T. Porter, L. Carpenter, Distributed Ray Tracing, Proc. SIGGRAPH, Vol.18, No.3, pp.137-145, 1984.
[7] F. C. Crow, Shadow algorithms for computer graphics, Proc. SIGGRAPH, Vol. 11, Issue 2, pp.242-248, 1977
[8] M. J. Kilgard, Robust Stencil Shadow Volumes, CEDEC 2001 Presentation, 2001
[9] L. Carpenter, The A-buffer, an Antialiased Hidden Surface Method, Proc. SIGGRAPH, Vol.18, No.3, pp.103-108, 1984.
[10] R. Cook, L. Carpenter, E. Catmull, The Reys Image Rendering Architecture, Proc. SIGGRAPH, Vol.21,No. 4, pp.95-102, 1987.
[11] W. Jakob, S. Marschner, Manifold Exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport, Proc. SIGGRAPH (To appear) , 2012.
[12] L. Williams, Pyramidal parametrics, Proc. SIGGRAPH, Vol. 17,No. 3, pp.1-11, 1983.
[13] J. McCormack, R. Perry, K. I. Farkas, N. P. Jouppi, Feline: Fast elliptical lines for anisotropic texture mapping, Proc. SIGGRAPH, pp.243-250, 1999.
[14] T. Lokovic, E. Veach, Deep shadow maps, Proc. SIGGRAPH, pp.385-392, 2000.
[15] J. M. Lane, L. C. Carpenter, T. Whitted, J. F. Blinn, Scan Line Methods for Displaying Parametrically Defined Surfaces, CACM 23(1), pp.23-34, 1980.
[16] J. C. Morrison, Fast anti-aliasing polygon scan conversion, Graphics Gems, pp.76-83, 1990.

付録：格子による形状の分割

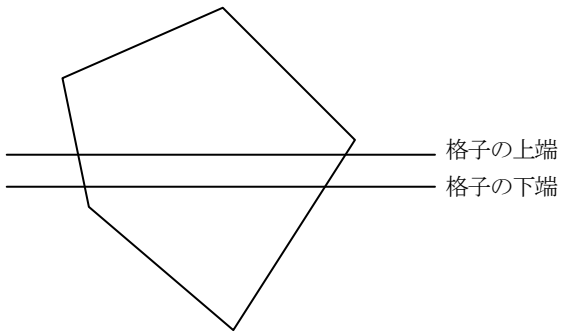
ここではオブジェクトからオブジェクト片へのダイシングや、オブジェクト片からマイクロポイント用の微小オブジェクト片を生成する方法について述べる。具体的には、オブジェクトに付随する UV 座標値を用いて、それぞれのオブジェクト片に関係するテクセルが 1 テクセル以下になるようダイシングする方法や、オブジェクト片をピクセルの大きさで分割しマイクロポイントを生成する方法に関して説明を行う。

それぞれのオブジェクト片に関係するテクセルが 1 テクセル以下になるようダイシングする場合はテクセルの格子にてオブジェクトを分割し、またマイクロポイント用の微小オブジェクト片を生成する場合はオブジェクト片をピクセルの格子にて分割する事となる。どちらの場合においても、形状を格子にて分割する処理に帰着される。

これらの処理については様々な方法が提案されているが[10,15]、本付録では我々が検証用レンダラで用いた方法について説明する。

対象オブジェクトはそれぞれ適切な方法にて多角形の集合として表現されているものとする。オブジェクト片も多角形として処理を行う。この条件の下では、オブジェクトからオブジェクト片へのダイシングとは、多角形（オブジェクト）に付随する情報を用いて多角形（オブジェクト片）へと分解する事に他ならない。

我々はスキャンコンバージョン[16]に良く似た方法にて本問題を処理した。スキャンコンバージョンにおいては水平方向のピクセルの列（格子の列）を直線として考え、多角形と水平な直線との関係によりレンダリングを行う。我々が用いた方法では水平な格子の列を水平な直線ではなく、水平な格子の集合として扱う。



多角形の格子による分割手順

A.1 多角形を格子が定義されている空間に投影

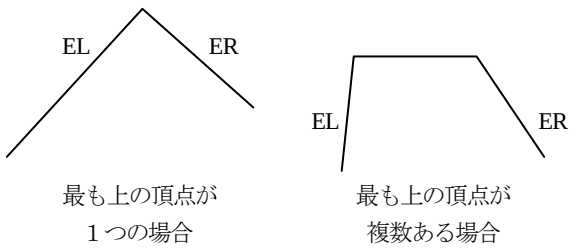
まず、多角形を格子が定義されている空間に投影する。分割後の形状はワールド空間での座標値が必要になる為、ワールド座標における座標値も保持しておく。また、以下の処理で必要となる為、格子座標系の上下左右方向を決めておく。

A.2 格子座標系にて頂点を並び替える

多角形を構成する頂点を、格子の座標系にて上から下へと並び替え、格子座標系における形状の最上端および最下端の位置を求めておく。

A.3 左稜線、右稜線を求める

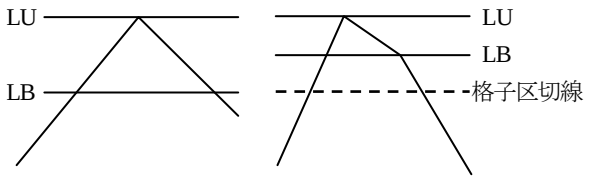
A.2 で求めた最も上に存在する頂点を端点に持つ、左右の稜線を求める。最も上に存在する頂点が複数ある場合（最も上にある稜線と格子の水平線が平行な場合など）は、もっとも上に位置する稜線に接続している左右の稜線を求める。それらをそれぞれ EL,ER とする。



A.4 分割を行う上下の水平線を求める

分割を行う上下の水平線を LU, LB とし、それぞれの初期値を、LU は最も上にある頂点を通過する水平な線、LB は最も上にある頂点を含む格子の下端を通過する水平線とする。ここで、もし LB が稜線 EL または ER と交わらない場合は、EL、

ER の下方の端点でより上の方にあるものと交わる水平線を LB とする。

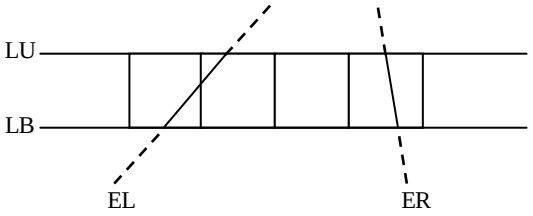


EL,ER が下の格子にかかる場合　　ER の下の端点が格子区切線よりも上に存在する場合

※通常のスキャンラインコンバージョンと同様に、この LU,LB を上から下に順次下げていきつつ分割処理を行う（新しい LU = 古い LB)。基本的に LU および LB は格子座標における格子の上端と下端となるが、EL または ER と交点を持たない場合は、交点を持つように LB を上の方にずらす。

A.5 LU-LB 間の格子にて形状を分割する

LU と LB に挟まれた領域に対し、格子単位で分割を行う。まず、LU－LB 間の形状領域を含む格子を求める。

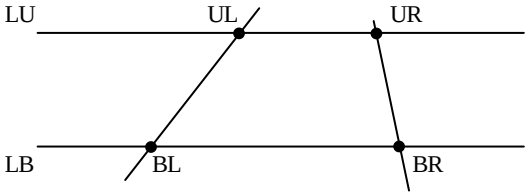


A.5.1 分割の準備

これらの格子に対して順次以下の処理を行う：
該当格子の左側垂直線および右側垂直線をそれぞれ LL, LR とし、格子座標系における点 p に対し、以下のように関数 f(p, LL, LR) を定義する。

$$f(\mathrm{p}, \mathrm{LL}, \mathrm{LR}) = \begin{cases} 0, & \mathrm{p}\ \text{は LL より左側} \\ 1, & \mathrm{p}\ \text{は LL と LR の間} \\ 2, & \mathrm{p}\ \text{は LR より右側} \end{cases}$$

LU, LB と EL, ER のそれぞれの交点を以下の様に UL, BL, UR, BR と名付ける：



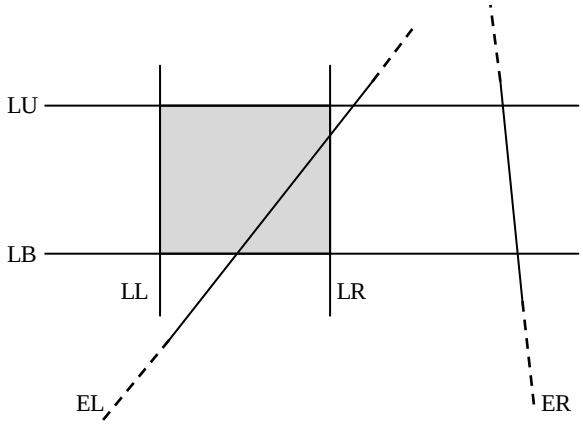
A.5.2 3 進 4 桁コード

A.5.1 で定義した関数 f と UL, BL, UR, BR を用いて di (i=1, …,4) を計算する。

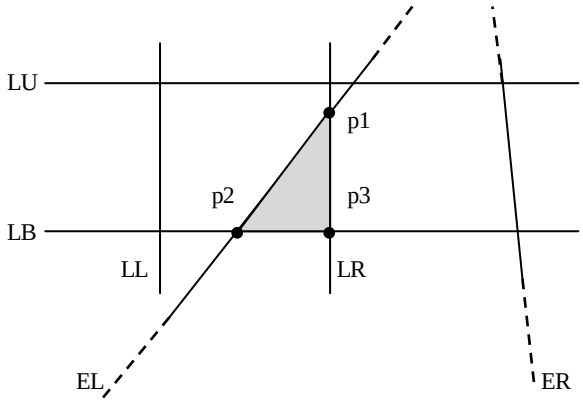
$$\begin{cases} \mathrm{d1} = f(\mathrm{UL}, \mathrm{LL}, \mathrm{LR}) \\ \mathrm{d2} = f(\mathrm{BL}, \mathrm{LL}, \mathrm{LR}) \\ \mathrm{d3} = f(\mathrm{UR}, \mathrm{LL}, \mathrm{LR}) \\ \mathrm{d4} = f(\mathrm{BR}, \mathrm{LL}, \mathrm{LR}) \end{cases}$$

di ∈ {0,1,2} であり、これら４つの数字を d1 から d4 まで並べて 4 桁の 3 進数として考える。この数値を 3 進 4 桁コードと呼ぶ事にする。

形状と格子との関係により、 3 進 4 桁コードは様々な値をとる。例えば以下の様な配置状況の場合、3 進 4 桁コードは 2122 （＝10 進数で 71）となる。



3 進 4 桁コードが分かれば､形状の格子により切り取られる領域が、LU, LB, LL, LR, EL, ER それぞれの直線のどの交点により構成されるかが分かる。例えば、上図の場合は、



となり、EL と LR の交点 p1 と、EL と LB の交点 p2 と、LB と LR の交点 p3 からなる 3 角形が該当格子により切り取られる領域である事が分かる。

3 進 4 桁コードは値としては 81 種類の値をとるが､そのうち有効な値は 34 種類である。 0000 および 2222 の２つは有効な領域を切り取らない値である。また、BR は BL よりも右側に配置されなければならないが、このような条件に矛盾する 0100 のような値（エラーコード）が 45 通りある（UL と UR の関係に矛盾するものも含む)。

A.5.3　格子よる分割

3 進 4 桁コードにより処理を分岐し、分割される具体的な領域を計算する。格子により切り出される領域の形は 3 角形から 6 角形までの多角形である。

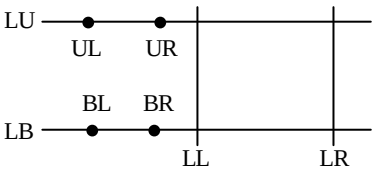
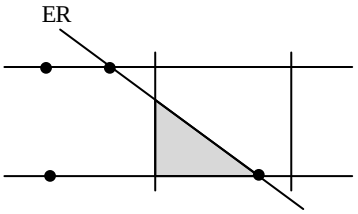
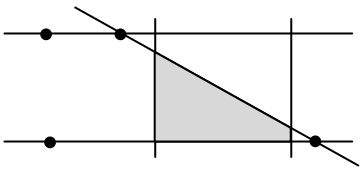
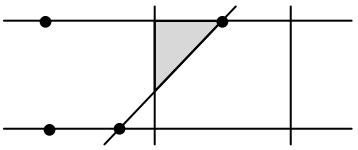
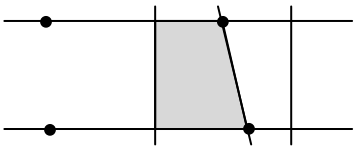
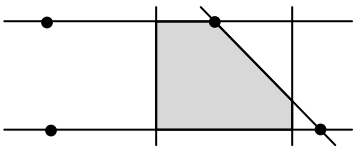
それぞれの頂点の座標はまず格子座標系にて計算される。その後、格子座標系からワールド座標系に変換する。

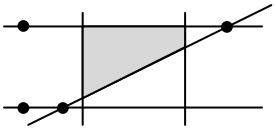
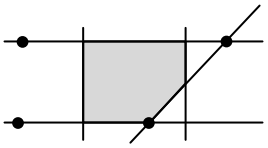
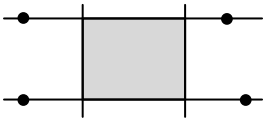
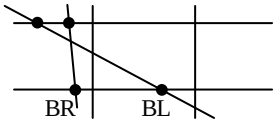
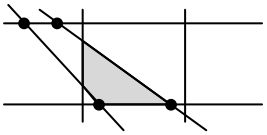
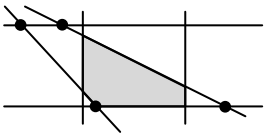
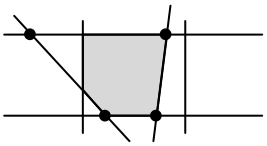
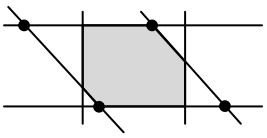
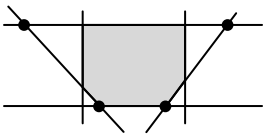
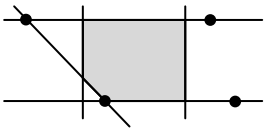
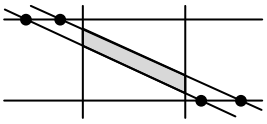
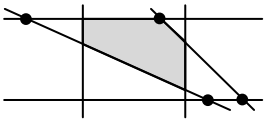
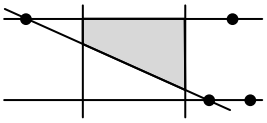
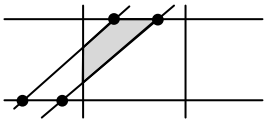
A.6 LU, LB, EL, ER の更新

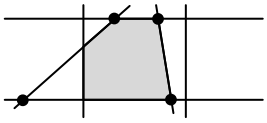
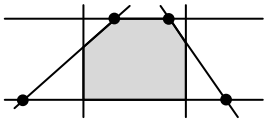
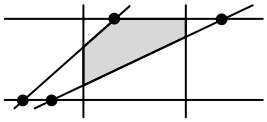
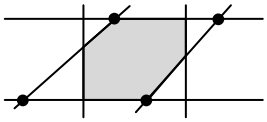
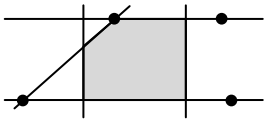
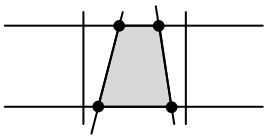
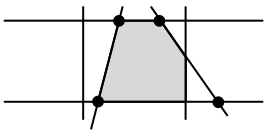
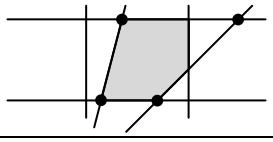
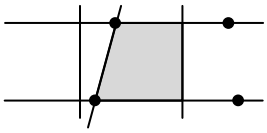
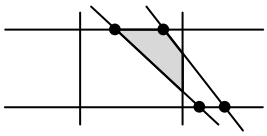
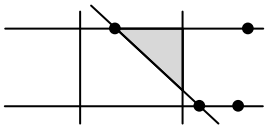
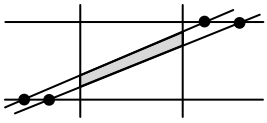
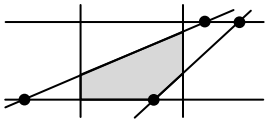
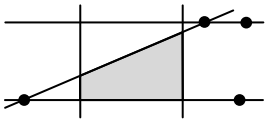
LB を新たな LU とし、新しく定めた LU と EL, ER との下端との関係により LB を求める。LB が EL または ER の下端と交わっている場合は、次の、左もしくは右稜線を求めそれを新しい EL もしくは ER として使用する。

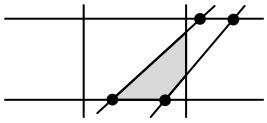
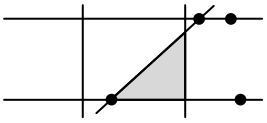
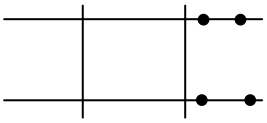
LB が分割対象形状の下端にまで達するまで、A.5 および A.6 の処理を順次繰り返す。

付録： 3 進 4 桁コード表

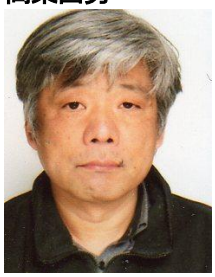
コード値： 0000 UL, BL, UR, BR が対象領域の外側にある為、この格子では分割されない	
コード値： 0001 このコードでは、該当の格子により分割される領域は LL と ER の交点、LL と LB の交点、ER と LB の交点からなる三角形となる。	
コード値： 0002 同様に、このコードでは格子により分割される領域は四角形となる。	
0010	
0011	
0012	

コード値： 0020 	0021 
0022 	0100 （エラー） ※ BL と BR の位置が矛盾 
0101 	0102 
0110 （エラー） ※ BL, BR の位置が矛盾	0111 
0112 	0120 （エラー） ※ BL, BR の位置が矛盾
0121 	0122 
0200 および 0201 （エラー） ※ BL, BR の位置が矛盾	0202 
0210 および 0211 （エラー） ※ BL, BR の位置が矛盾	0212 
0220 および 0221 （エラー） ※ BL, BR の位置が矛盾	0222 
1000, 1001, 1002 （エラー） ※ UL, UR の位置が矛盾	1010 

1011 	1012 
1020 	1021 
1022 	1100 ～ 1110 （エラー） ※ UL, UR や BL, BR の位置が矛盾
1111 	1112 
1120 （エラー） ※ BL, BR の位置が矛盾	1121 
1122 	1200 ～ 1211 （エラー） ※ UL, UR や BL, BR の位置が矛盾
1212 	1220 および 1221 （エラー） ※ BL, BR の位置が矛盾
1222 	2000 ～ 2012 （エラー） ※ UL, UR の位置が矛盾
2020 	2021 
2022 	2100 ～ 2120 （エラー） ※ UL,UR や BL, BR の位置が矛盾

2121 	2122 
2200 ～ 2221 （エラー） ※ UL, UR や BL, BR の位置が矛盾	2222 （格子は形状を切取らない） 

高桑昌男



1980 年日本大学生産工学部数理工学科卒。1983 年トーヨーリンクスにて、CG制作、開発に従事。1988 年よりフリーランス。1992 年慶應義塾大学SFC講師。1996 年岐阜県立国際情報科学芸術アカデミー教授。2012.年 4 月より現職。

飯倉宏治

1992 年豊橋技術科学大学工学部知識情報工学科卒。1994 年豊橋技術科学大学大学院工学研究科修士課程修了。1998 年岐阜県立国際情報科学芸術アカデミー アートアンドメディアラボ科卒業。多摩美術大学情報デザイン学科非常勤講師、マイクロソフトプロダクトディベロップメントリミテッド等を経て 2011 年 3 月より現職。