Vol.2011-CG-145 No.14
Vol.2011-CVIM-179 No.14
2011/11/17

任意形状の堆積形成手法

櫻井快勢[†]　　宮田一乘[††]

本報告では，任意オブジェクトの堆積を形成する手法を提案する．はじめに，堆積を構成するオブジェクトが堆積形状に従って重なるように配置する．次にオブジェクト同士の領域の重なりを解消するために，堆積形状の法線方向の重なりが少ない位置に各オブジェクトを再配置する．本手法では，比較的少ない計算コストで堆積を表現できることを確認した．本手法を用いることにより，おにぎりなどの凝集体を表現することができる．

A Procedure of Pile Modeling

Kaisei Sakurai[†] and Kazunori Miyata[††]

We propose a method for modeling piles of arbitrary objects. Our method arranges objects that slightly overlap their neighborhoods on user-specified pile-shape to determine the number of the objects. Then, it rearranges the object toward normal on the pile-shape for avoiding intersections of piles. By this procedure, piles or condensations of objects, such as rice-ball, can be represented.

1. はじめに

皿に盛られた料理やぬいぐるみの山など，身近には多くの堆積されたものがあり，CG で現実を表現するためには，それらのモデリングは必須である．ここでの堆積とは，地層ではなく，オブジェクトが積み重なる状態を指す．本稿では，堆積を次の二点を満たす状態と定義する．

・オブジェクト同士が接触する
・地面などの面に投影したオブジェクトの領域が重なる

ただし，投影の方向は投影面に対して法線方向とする．たとえば，図 1 に示すように，接触しているオブジェクト A と B に対し，面に投影された領域 A’と B’に重なりが存在するとき，A と B は堆積しているとする．

また，多くの堆積されたものは，非周期的に配置されているため，モデリングではこの特徴を考慮する必要がある．

粘着性の低い物体が自然に積み上がったものは，ピラミッドのような形成を成すことが多い．しかし，ご飯粒のようにオブジェクトに粘着性があると，堆積の形状はピラミッド形に限らず，任意形状を形成する．CG コンテンツの制作過程で，手作業による堆積の形成は労力と時間を要するため，堆積のモデリングが自動化できると有用であると考える．自動的にモデリングする手法としてプロシージャルモデリングがあるが，任意のオブジェクトを生成することは難しい．たとえば， L-system などの構文によるモデリング法では任意の形状をモデリングできるが，十分なスキルが必要であり，現実的ではない．

本稿では，堆積を形成するための高いスキルを必要としない自動化手法を提案し，任意形状が非周期的に配置された堆積を形成することを目的とする．

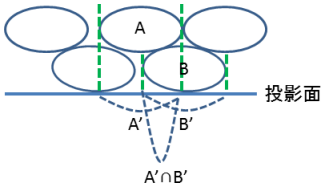


図 1 堆積の一例．

[†] 北陸先端科学技術大学院大学 知識科学研究科
Japan Advanced Institute of Science and Technology, School of knowledge science
[††] 北陸先端科学技術大学院大学 ライフスタイルデザイン研究センター
Japan Advanced Institute of Science and Technology, Research Center for Innovative Lifestyle Design

Vol.2011-CG-145 No.14
Vol.2011-CVIM-179 No.14
2011/11/17

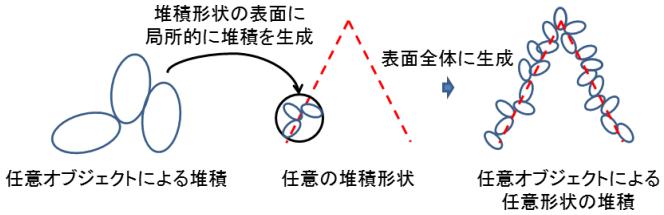


図 2 本アプローチの概略.

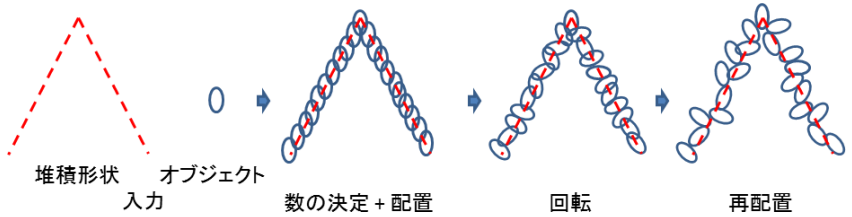


図 3 堆積生成の概略.

2. 関連研究

代表的な既存の堆積の表現手法としては，物理計算の中で所望のピラミッド形を形成するシミュレーション [1]や 3 次元のサンプルから同様の構造を持つ集合体を生成する 3 次元のテクスチャシンセシス [2]，堆積された石のプロシージャルモデリング [3]などが挙げられる．一方，本研究では，任意オブジェクトが任意の堆積形状を形成するように，堆積の構造を自動的に生成する手法を提案する．

一方，オブジェクトの配置手法がいくつか提案されている．隙間なく繰り返しパターンを生成する Escherization [4]や，2 次元の任意オブジェクトのモザイクである Jigsaw image mosaics [5]，指定した形状を近似するようにオブジェクトを配置する 3D collage [6]， Poisson-disk distribution を用いてオブジェクトを一様に配置してテクスチャを生成する Object distribution function [7]などが代表的な手法として挙げられる．しかし，これらの研究では，堆積のように密で非周期的な配置は不可能である．

また，一般的なモザイクを生成する手法も提案されている [8,9]が，任意オブジェクトの配置には不向きである．

自動的にパターンを生成する手法として，いくつかのプロシージャルモデリングが提案されている．そのひとつとして Perlin が提案したテクスチャ生成手法は幅広く活用されており，煙，布地，岩肌などの表現が可能である [10,11]．Worley が提案した細胞のような形状を生成する Cellular texture も活用されている [12]．また，木 [13]や街 [14]などを対象とした手法も提案されている．

以上で示したように，既存手法では堆積形状とオブジェクトの両者を任意に指定することは不可能である．この課題に対し，本手法では，図 2 に示すように，指定した任意の形状の表面全体に，1 章で前述した堆積の定義を局所的に満たすように任意オブジェクトを堆積させる．

3. 堆積生成

本手法では，堆積形状の表面付近にオブジェクトを接触するように配置することで，

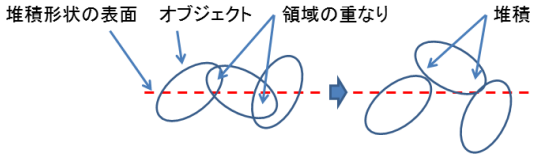


図 4 領域の重なりと堆積の関係.

堆積を表現する．配置には，配置するオブジェクトの数および，個々のオブジェクトの姿勢（ロー・ピッチ・ヨーの回転角）と位置を決定する必要がある．本手法は，これらを図 3 に示すようにオブジェクトの数，姿勢，位置の順で決定する．

まず，配置するオブジェクトの数を決定するために，3.1 節で後述する条件でオブジェクトを配置する．この段階では，概ねの位置を決定するにとどめる．非周期的な見た目を形成するために，姿勢はランダムに回転させる．その後，接触を避けるように個々のオブジェクトを堆積形状の法線方向に従い移動させ,位置を調整することで,堆積を形成する．図 4 に示すように，意図的に重なるように配置することで，再配置で堆積形状の法線方向の移動のみで堆積を構成できる．

本手法では，オブジェクトの重なりを判定するために，配置するオブジェクトは内外が明確な閉じたメッシュで定義する．また，内外を判定するために，このメッシュは，すべての頂点の法線ベクトルが外側，もしくは内側に向くように統一されるとする．また，堆積形状もメッシュで指定するが，こちらは開いたメッシュも扱うことができるものとする．

3.1 オブジェクト配置による数の決定とオブジェクトの姿勢

本節では，オブジェクトの数と姿勢の決定方法を詳しく述べる．まず数の決定のために，以下の条件で配置を行う．

条件 1：偏りなく配置する
条件 2：オブジェクト同士に重なりを付ける

条件 1 は，堆積に穴を作らないために必要な条件であり，条件 2 は，図 4 に示すよ

Vol.2011-CG-145 No.14
Vol.2011-CVIM-179 No.14
2011/11/17

うに再配置での移動を法線方向に限定するために必要な条件である．これらのオブジェクトの数を決定する配置の十分条件を満たすために，堆積形状の表面でポワソンディスクサンプリングのダートスローイング法を用いて，オブジェクトを配置する．

ダートスローイング法は，ブルーノイズ特性を持った 2 次元の点群の配置手法である [15]．点を中心とする排他的な円を設定し，その円同士が重ならないように点を配意する．円の半径を r としたとき，点と近隣点の間には[r,2r]のランダムの距離が保たれる．この分布法を用いれば，点の分布に偏りが発生せず，条件 1 を満たす．また，円の半径 r をオブジェクトに内接する円の半径 r'の 1/2 とすると，その円の中心同士の距離は[r'/2,r']となり，オブジェクトは近隣のオブジェクトと接触もしくは重なり，条件 2 を満たす．

曲面上で，ダートスローイング法を用いるためには，比較的高コストで実装が複雑なジオデジック距離を計算に用いることができるが，ユークリッド距離のほうが，高速かつ，実装が簡単であるため，本手法ではユークリッド距離を用いる．本手法で用いるダートスローイング法では，排他的な領域として円ではなく，球を用いる．堆積形状上に，ユークリッド距離で点間の距離を計算し，配置済みの球との内外判定によって点の配置の可否を決定する．

ダートスローイング法で用いる球には，オブジェクトに内接する最大の球を用いる．条件 1 を満たすために，球の大きさは一様である必要がある．オブジェクトに対して内接する最大の球は一意に決定でき，一様の大きさで配置する十分条件を満たす．また，小さな球を用いて配置するとオブジェクト数が多くなり，計算コストが膨大になるため，大きな球を用いることで配置するオブジェクト数を少なくし，計算コストを小さくする．内接する最大の球は，3.2 節で後述する陰関数で用いる球から最大の半径を持つ球を選択する．

一般的なダートスローイング法の実装に画像を用いる方法がある．点を配置するたびに，その点を中心とする円内の画素にフラグを立てることで，画素のフラグの有無で配置済みの円の内外判定ができる．ただし，円の半径は画素よりも大きい必要がある．本手法では，堆積形状のメッシュの表面で同様の処理を行う．堆積形状の表面に密な点群を生成し，それらを上述したアルゴリズムにおける画素と同等に扱う．堆積形状の点群からフラグがない点を 1 つ選択し，オブジェクトを配置する．その後，選択した点を中心に球を配置し，球内の点群にフラグを立てる．すべての点にフラグが立つとオブジェクトの配置を終了する．

堆積形状表面に生成する密な点群は，堆積形状のメッシュを細分割して得られる頂点を用いる．細分割を十分に行うことで，堆積形状の表面に多くの点を生成できる．本手法は，Catmull-Clark 細分割 [16]を応用した細分割で，面と辺の中心に新しい頂点を作る．Catmull-Clark 細分割で滑らかな曲面に近似するとき，特異点（メッシュのリンクが 4 頂点以外の頂点）周りの分割で尖る問題があるが，本手法では，滑らかな曲面に近似しないため，この問題は無視できる．図 5 に示すように，ポリゴンの重心と辺の中心に新しい頂点を付加する．

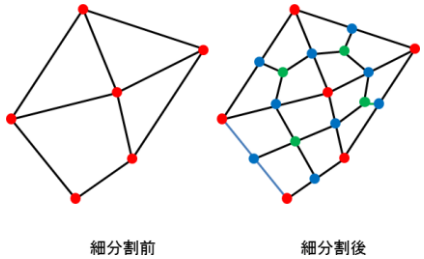


図 5 細分割による点の配置．
赤点は細分割前の頂点．緑点はポリゴンの重心．青点は辺の中心．黒線分は辺．

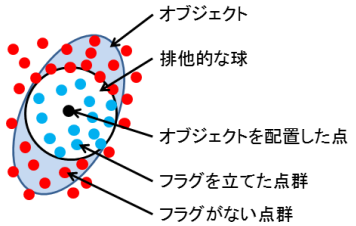


図 6 堆積形状の表面上におけるオブジェクトの配置．黒点はオブジェクトを配置した点．青点はフラグを立てた点群．赤点は，フラグなしの点群．

点を密に配置した後，オブジェクトを 1 つずつ点上に配置する．図 6 に示すように，1 つのオブジェクトを配置するたびに，配置した点の周辺でユークリッド距離 r'/2 内の点を排他的な領域内の点とする．

複数種のオブジェクトを配置する場合，指定した割合で配置のオブジェクトを替える．

3.2 オブジェクトの再配置

本節では，堆積を形成するための再配置方法を詳しく述べる．再配置は，堆積形状の法線方向に各オブジェクトを移動し，初期配置の重なりを解消する．オブジェクトを陰関数で表現して，重なりを関数化し，その関数を指定範囲内の移動で最小にことで，重なりを解消する．重なりを完全に解消すると，オブジェクトが堆積形状の表面から遠く離れる可能性があるため，移動の範囲を指定して堆積形状を保つこととした．

重なりを解消することは，オブジェクトの重なる部分の体積を最小化するとも言い

Vol.2011-CG-145 No.14
Vol.2011-CVIM-179 No.14
2011/11/17

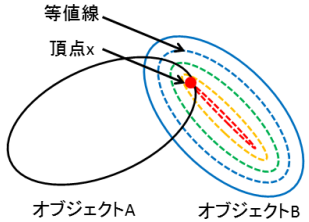


図 7 陰関数を用いた深度の取得.
関数の値を色で表現（赤黄緑青の降順）.
オブジェクト A 上の頂点 x はオブジェクト B に対して赤色の深度.

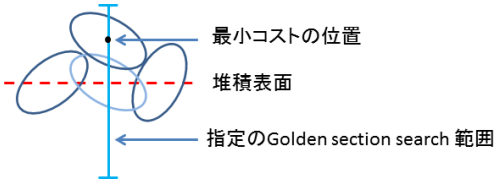


図 8 堆積形状の表面付近での golden section search.

換えることができるが，一般的に，任意のオブジェクト同士の重なりの体積を求めるためには, ブール計算で任意オブジェクト同士の積集合を求めて体積を計算するため, 複雑な計算が必要となる．これを避けるために，本手法では陰関数を用いて，重なりの深度を求める．陰関数は面からの距離に依存した値を返すため，図 7 に示すようなオブジェクト A, B の重なりがある場合, 重なる頂点 A での陰関数値を求めることで, オブジェクト A の頂点 x におけるオブジェクト B に対する深度がわかる. 本手法では, オブジェクトの重なりの量を，重なる可能性がある近隣のオブジェクトの陰関数に対して，各オブジェクトの頂点の陰関数値の総和として求める．陰関数のサポートサイズはオブジェクトの内部に限定されるように設定し，陰関数はオブジェクトの外部では 0 の値を返す．オブジェクトのコスト関数 c は式(1)で定義する.

$$c(t) = \sum_{\mathrm{x}\in\mathrm{X}} f(\mathrm{x} + t\mathrm{n}) \quad (1)$$

このとき，X はあるオブジェクトの頂点 **x** の集合，**n** はオブジェクトが配置された堆積形状上の位置の法線ベクトル，t は法線上の移動量，f は 3.2.1 項に後述の陰関数である．$c(t)=0$ を満たすとき，オブジェクト同士が重なることはなくなるが，オブジェクト同士が離れる可能性がある．できるだけ接触させるために，堆積形状の表面に

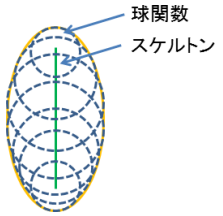


図 9 スケルトンと球関数を用いた陰関数表現.

近い位置を低コストとする式(2)のコスト関数 c’を用いる.

$$c'(t) = c(t) + a|t| \quad (2)$$

このとき，a は距離を操作するための係数とする．各オブジェクトに対して，式(2)のコスト関数 c’が最小となる t を求める．最小値の解法は，Golden section search を用いる. Golden section search は, 指定した範囲における最適解を求める. 図 8 のように, 範囲を指定し, オブジェクトを最小コストの位置に移動する. 範囲を指定することで, 堆積形状からの距離に制限を付ける．一般的には大域解が望ましいが，計算コストが膨大である上に，解が確実に求まらない可能性があるため，本手法では局所解を求めることとした.

3.2.1 球を用いた陰関数

本手法では，陰関数に球状の関数群を用いる．この関数群は，任意の点に対して，内側に限定してオブジェクトの表面との距離を返す.

関数群を構成する球状の関数は, 図 9 に示すように, スケルトン（もしくは medial axis と呼称される幾何情報）上に中心点をとり，中心点からオブジェクトの表面までの最短距離を半径として持つ．本手法では， Amenta らの手法[17]を用いて，スケルトンに近似したメッシュを取得する．スケルトンのメッシュの頂点を中心とし，オブジェクトに接触する球は，必ず内側にある．これらの球状の関数の 0 セットサーフェイスの和集合は，オブジェクトの表面を近似する．そのため，オブジェクトの表面からの距離を返す関数として用いることができる．この陰関数 f は式(3)で定義する.

$$f(\mathrm{x}) = \max_{g\in\mathrm{G}} g(\mathrm{x}) \quad (3)$$

$$g(\mathrm{x}) = (r - ||\mathrm{x} - \mathrm{c}||)s(\mathrm{x}, \mathrm{c}, r)$$

$$s(\mathrm{x}, \mathrm{c}, r) = \begin{cases} 1 & \text{if } ||\mathrm{x} - \mathrm{c}|| < r \\ 0 & \text{otherwise} \end{cases}$$

このとき，G は g の集合であり，r は球の半径，c は球の中心（スケルトン上の点），s はサポート関数である. 陰関数 f は, x がオブジェクトの内側のとき, 正の値を返す.

Vol.2011-CG-145 No.14
Vol.2011-CVIM-179 No.14
2011/11/17

4. 結果

本手法は, 幾何計算ライブラリ CGAL と並列計算ライブラリ OpenCL を用いて C++で実装した. 実験の環境は, CPU : Intel Core i7 950 (3.07 GHz), メモリ : 12.0 GB RAM, OS : Windows7 64bit, GPU : NVIDIA GeForce GTX 570 の構成である. 実験に用いる図 10 に示すモデルは, Maya を用いて球を変形させてモデリングした. それぞれの制作時間は概ね 2 分程度であった. また, Stanford dragon と Stanford bunny も実験のモデルとして使用した. ただし, Stanford bunny は配置するには頂点が多すぎるため, 頂点数を削減して使用した. モデルの頂点数と面の数を表 1 に示す.

単一のオブジェクトを用いた結果を図 11 に示す. コスト関数式(2)内の a は 0.1 とし, 再配置の範囲は[-b,b]とした. ここで, 各オブジェクトの中心から最遠点の距離を b とする. 図 11(a)と(c)は, 図 10(a)のおにぎり形にオブジェクトを配置した結果である. オブジェクトはそれぞれ図 10(b)のご飯粒と Stanford bunny である. また, 図 11(b)と(d)は, Stanford dragon 形にオブジェクトを配置した結果である. ただし, 図 10(a)は 3.1 節の細分割 2 回で頂点を 6242 点にした. また, 図 11(d)では, 堆積形状を視認しやすくするためにオブジェクトのサイズを図 11(b)より小さくした.

また, 図 10(b) のご飯粒と(c)の栗のオブジェクトを用いた結果を図 12 に示す. ご飯粒と栗の配置の割合は, 100:1 とした. 再配置の範囲は, ご飯粒が[-b,b], 栗が[-0.1b,0.1b]とした. 栗の範囲が小さい理由は, ご飯粒に比べてサイズが大きく, 再配置で栗が外に出やすくなるのを防ぐためである.

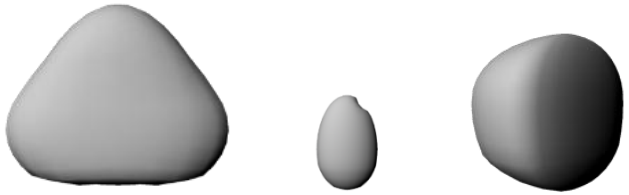
以上の結果の配置数と計算時間を表 2 に示す. 陰関数化の計算時間は, スケルトンのメッシュと球の半径を求めた時間の総和である. 陰関数化の計算時間の差異は, スケルトンの計算がオブジェクトの形状に依存するためと考えるが, 原因は特定できていない. 図 11(a)と(c)は同じ堆積形状を用いたにも関わらず, 配置数が異なっている. これは bunny よりご飯粒の方が小さいため, ご飯粒の方が多く配置された. また, 配置の計算時間も, 配置数に依存している. 図 11(a)は(c)のおおよそ 2 倍の配置数のオブジェクトで構成され, 配置にかかる時間もおおよそ 2 倍である. 配置のダートスローイング法での終了条件である,「配置可能な頂点がなくなること」を満たすまで配置を繰り返すため, 配置に対する計算時間は配置数に依存する. 一方, 再配置では, 図 11(a)と(c)の計算時間が概ね同じである. 再配置は, 配置されたオブジェクトを陰関数で計算するため, 陰関数の数が計算時間に影響する. 陰関数の数は, オブジェクトの頂点と同数であるため, bunny のほうが時間を要する. そのため, 配置数に違いがあるにも関わらず, 概ね同じ程度の時間を要したと考える. 図 11(b)と(d)の比較でも同様のことが言える. また, 再配置では, 図 11(b)と(d)では, 配置数の割合に比べて, 図 11(d)のほうが時間を要しており, 陰関数の数が多いためと考える.

表 1 使用するモデルの頂点数と面数.

	頂点数	ポリゴン数
図 10(a)	382	400
図 10(b)	382	400
図 10(c)	382	400
Dragon	50000	100000
Bunny	546	1088

表 2 配置数と計算時間. ()内は堆積形状の頂点数

	配置数[個]	陰関数化[s]	配置[s]	再配置[s]
図 11 (a)	5788 (6242)	0.24	6.71	272.86
図 11 (b)	16396 (50000)	0.24	33.27	628.28
図 11 (c)	3096 (6242)	0.39	3.37	254.35
図 11 (d)	23031 (50000)	0.39	86.80	1365.07
図 12	5365 (6242)	ご飯粒:0.24 栗:0.12	6.27	194.30



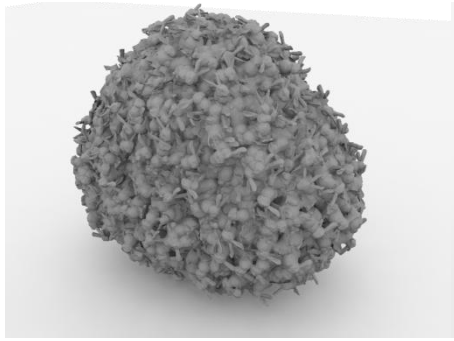
(a) おにぎり形　(b) ご飯粒　(c) 栗

図 10 制作した堆積形状と配置するオブジェクト.
(a)は堆積形状, (b)と(c)は配置するオブジェクト.

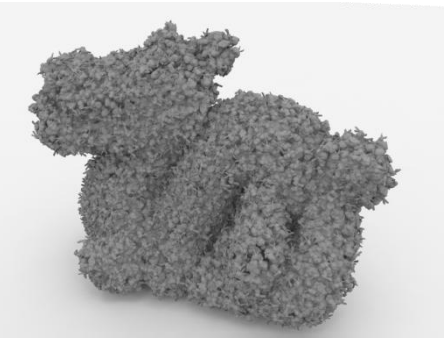
Vol.2011-CG-145 No.14
Vol.2011-CVIM-179 No.14
2011/11/17

(a) おにぎり

(b) dragon 形おにぎり



(c) bunny のおにぎり



(d) bunny の dragon 形堆積

図 11 堆積の生成結果.

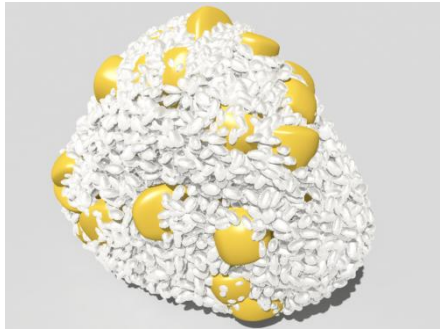


図 12 複数のオブジェクトでの生成結果.

5. 考察

本手法は，3 次元の堆積をできるだけ簡単にモデリングするために，パラメータの設定が難しい物理計算や L-system を避けて幾何的な処理のみで自動生成した．実験により，堆積形状とオブジェクト，およびオブジェクト間の距離と再配置の範囲を指定する 2 パラメータのみを設定するだけで所望の形状を得ることを確認した．

最近の堆積生成の手法である 3 次元のテクスチャシンセシス[2]では，入力のサンプルで小さいながらにも手作業で堆積を形成する必要があるが，本手法では，その必要はないため，手作業の量は減ると考える．

本手法は，非周期的な堆積を対象としており，粘着性があるオブジェクトの表現に適していると考える．一方，粘着性が低く，球に近いオブジェクトが形成するような周期的な配置は不可能である．しかし，周期的な配置はコピー＆ペーストで形成できるため，手作業でも大きな労力を要しないと考える．

再配置の最適化で，大域解を避けた理由は，高次元の計算をなくすためであるが，今後コンピュータの計算能力，またはアルゴリズムの向上により，可能となると考える．本手法での最適化に局所解を用いたのは，計算コストの問題のみであるため，大域解を用いても堆積を形成できると考える．

図 11 に示す結果からも視認できるように, オブジェクトの大きさ以下の粒度を堆積形状にて表現することはできない．今後，堆積形状を表現するための適切な粒度を明らかにすることで，生成された堆積が指定の堆積形状に近似すると考える．

オブジェクトの姿勢は，現状ではランダムに決定しているが，ベクトル場などの方向の指定をすることで，操作できると考える．

本手法では，陰関数に Amenta らの手法で求めたスケルトンを用いるため，陰関数の数はモデルの頂点数と同数となる．そのため，頂点の数が多い場合，陰関数の計算回数が増加する．また，陰関数のサポートサイズの重なりも多くなるため，無駄な計算が多くなる．実験では，並列処理を用いて陰関数値を求めているため計算時間に大きな影響を与えないが，CPU-GPU の転送に時間を要するうえに，計算時にメモリを圧迫し，計算が不可能になることがあるため，今後，適切な削減が必要である．

サイズが大きく異なる複数のオブジェクトをダートスローイング法で配置するとき，小さなオブジェクトの排他的な球の距離で配置されることがある．この原因は内外判定のみで配置の有無を決定しているためである．オブジェクトの重なりが多くなりすぎるため，今後，オブジェクトの削除を考慮する必要がある．

6. まとめ

本稿では，非周期的に堆積されたオブジェクトを表現する手法を提案した．オブジェクトで任意の堆積を形成するために，配置する数，個々のオブジェクトの姿勢，位

Vol.2011-CG-145 No.14
Vol.2011-CVIM-179 No.14
2011/11/17

置の順で決定した．配置する数は，堆積形状の表面に，ダートスローイングの要領でオブジェクトを配置することで決定した．これにより，概ねの位置を決定することができ，再配置では，移動方向を堆積の法線方向のみに限定することができた．姿勢をランダムに決定することで構成要素が非周期的に配列した堆積を表現した．再配置の移動量は，陰関数を用いて定めたコスト関数を最適化することにより求めた．最適化では，大域解は計算コストが高すぎるため，それを避けるために局所解を用いた．

今後，計算コストの無駄を減らしつつ，堆積形状により近づくような幾何計算を試みる．

参考文献

1) Shu-Wei Hsu, John Keyser: Piles of objects, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2010, Volume 29 Issue 6, (December 2010)
2) Chongyang Ma, Li-Yi Wei, Xin Tong: Discrete element textures, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2011, Volume 30 Issue 4, (July 2011)
3) Adrien Peytavie, Eric Galin, Stephane Merillou, Jerome Grosjean: Procedural Generation of Rock Piles Using Aperiodic Tiling, Computer Graphics Forum (Proceedings of Pacific Graphics), Volume 28, Number 7, pp 1801--1810, (2009)
4) Craig S. Kaplan, David H. Salesin: Escherization, SIGGRAPH '00 Proceedings of the 27th annual conference on Computer graphics and interactive techniques, (2000)
5) Junhwan Kim, Fabio Pellacini: Jigsaw image mosaics, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002, Volume 21 Issue 3, (July 2002)
6) Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, Daniel Cohen-Or: 3D collage: expressive non-realistic modeling, NPAR '07 Proceedings of the 5th international symposium on Non-photorealistic animation and rendering, (2007)
7) Ares Lagae, Philip Dutre: A procedural object distribution function, ACM Transactions on Graphics (TOG), Volume 24 Issue 4, (October 2005)
8) Alejo Hausner: Simulating decorative mosaics, SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques, (2001)
9) Yoshinori Dobashi, Toshiyuki Haga, Henry Johan, Tomoyuki Nishita: Method for creating mosaic images using voronoi diagrams, Proceedings of Eurographics 2002 Short Presentations, pages 341--348, Saarbrucken, Germany, (2002)
10) K. Perlin, E. M. Hoffert: Hypertexture, ACM SIGGRAPH Computer Graphics - Special issue: Proceedings of the 1989 ACM SIGGRAPH, Volume 23 Issue 3, (July 1989)
11) David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley: Texturing and Modeling A Procedural Approach Third Edition, Morgan Kaufmann; 3 edition (December 16, 2002)
12) Steven Worley: A cellular texture basis function, SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, (1996)
13) Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomir Mech, Przemyslaw Prusinkiewicz: Self-organizing tree models for image synthesis, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2009, Volume 28 Issue 3, (August 2009)
14) Yoav I. H. Parish, Pascal Muller: Procedural modeling of cities, SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques, (2001)
15) Daniel Dunbar, Greg Humphreys: A spatial data structure for fast Poisson-disk sample generation, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2006, Volume 25 Issue 3, (July 2006)
16) Edwin Catmull, James Clark: Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes, Computer-Aided Design, pages 350--355, (1978)
17) Nina Amenta, Marshall Bern, Manolis Kamvysselis: A new Voronoi-based surface reconstruction algorithm, SIGGRAPH '98 Proceedings of the 25th annual conference on Computer graphics and interactive techniques, (1998)