修 士 論 文

3D小石モザイクの生成手法

北陸先端科学技術大学院大学
知識科学研究科知識科学専攻

北　直樹

2011年3月

修 士 論 文

3D小石モザイクの生成手法

指導教員　　宮田一乗　教授

北陸先端科学技術大学院大学
知識科学研究科知識科学専攻

0950018北　直樹

審査委員：　宮田　一乗　教授（主査）
西本　一志　教授
林　　幸雄　准教授
金井　秀明　准教授

2011年2月

目 次

図 目 次

第 1 章
はじめに

本章では，研究背景と本研究の概要を説明する．

1.1 研究背景

コンピュータグラフィックス (CG) における 3D オブジェクトのモデリングは，ハードウェアの性能向上や表示デバイスの高精細化に伴い，その規模やディテールは膨大かつ詳細なものが求められる傾向にある．しかしそれらを手作業で制作すると開発コストが大きくなる．そこで作業の効率化のため，コンテンツの手続き (プロシージャル) 生成，あるいは自動生成と呼ばれる技術が重要になってくる．これらの手法を用いることにより制作の負荷を大幅に低減できるだけでなく，制御パラメータを変化させることで生成モデルにバリエーションを与えることが可能となる．特に，仮想世界における景観シミュレーションやコンピュータゲーム，映画等におけるモデル制作においては，現実に存在する形状の再現だけでなく，デザイナーが意図した形状を反映したモデリングを行いたい場合も多い．また，プロシージャル技術がより有効な場合として，大規模なモデリングや複数のオブジェクトのモデリングが考えられる．例えば，景観シミュレーションでは，水や風といった流体の流れを考慮した設計を行う場合，あるいは都市計画設計を行う場合がそれにあたる．また，コンピュータゲームにおけるマップ制作等ではマップやステージそのものがストーリー上のキーファクターとなる場合や，難易度設定 (レベルデザイン) を行いたい場合などが挙げられる．

著者はこれらの問題意識から，本研究に先立つかたちで図書館シーンのプロシージャル・モデリング手法を提案した [Kita and Miyata 2010]．図書館シーンの描写では多数の本棚をモデリングする必要がある．しかし，本の並び方はそれぞれの本棚で異なっているのが自然であり，モデリングの際，単なるモデルの複製では視覚的な違和感 (いわゆるビジュアルアーティファクト) を伴ってしまうことが考えられる．そこでプロシージャルにランダムさを付加して本のモデルを配置することで，それぞれの本棚で配置の異なったモデルを生成する．他方，本棚のフロア配置を見ると，実際の図書館では多くの場合，対称性や

反復構造などのパターンに従って配置されていることがわかる．このようなパターンは自動生成と相性が良く，生成アルゴリズムに容易に組み込むことができる．結果として図 1.1 に示したように多数の本棚を配置するだけでなく，フロア配置も容易に変更できるため，少ない労力で大きく印象の異なるシーンを同一のアルゴリズムから生成できることを示した．

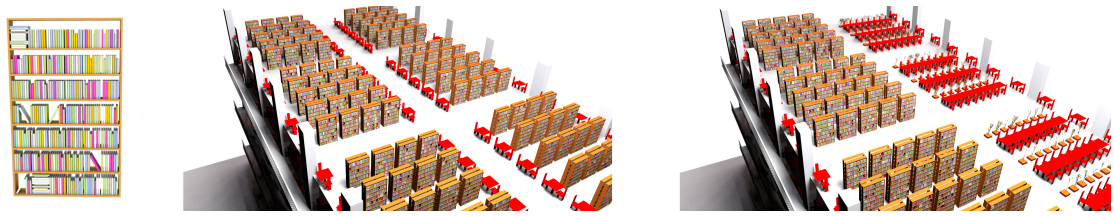


図 1.1: 人工物の構造を利用した図書館シーンのモデリング
[Kita and Miyata 2010] より引用．

このように，プロシージャル技術を用いることでスカルプティングなどのモデルの制作工程は，その多くがパラメータ設定に代替される．さらに，パラメータ設定次第で様々なバリエーションのモデルを生成することが可能となる．しかし，このような利点の反面，しばしば指摘される欠点もいくつか存在する．例えば，所望の形状を得るためにはパラメータ設定の試行錯誤が必要になること，そしてユーザ介入の余地があまりなく，ユーザがインタラクティブに生成プロセスを制御することが困難なことなどがそれにあたる．例えば，山岳形状の生成はこれまでフラクタルノイズを用いる手法が多く用いられてきた [Ebert et al. 2002]．しかし，それだけでは所望の山稜を生成することができない．そのため，近年ではイメージマップ (ユーザスケッチ，Example) を入力とするもの [Zhou et al. 2007] や，スケッチインターフェースで山稜をインタラクティブに指定可能な手法が報告されている [Gain et al. 2009, Hnaidi et al. 2010]．また，同様の傾向は都市のプロシージャル生成にも見受けられる．Parish と Müller は L-System と呼ばれる生成文法を拡張し，都市の道路ネットワークの生成に応用した [Parish and Muller 2001]．彼らのシステムは入力としていくつかのイメージマップ (高度マップや水陸の境界を表す 2 値マップ，人口密度マップなど) を用いて道路ネットワークを適応的に生成する．しかし，制御パラメータや L-System のルールの設定は直感的ではなく，設定が困難である．そのため，Lipp らは視覚的にインタラクティブにルールを設定するインターフェースを提案した [Lipp et al. 2008]．さらに，Chen らはテンソル場をインタラクティブに編集することで直感的に道路ネットワークを生成可能なモデリング手法を提案した [Chen et al. 2008]．そしてこれらの都市のプロシージャル生成の研究成果は CityEngine という商用のソフトウェアとして発表されている [Procedural Inc. 2010]．このような状況から，インタラクティブなプロシージャル技術の重要性が大規模なモデリングをはじめとして，今後ますます増してくる

ことが考えられる．

1.2 研究の概要

本研究では，小石モザイクの生成手法を提案する．小石モザイクは小石を断片とするモザイク画である．モザイク画は石や貝殻，陶磁器等からなる小さな断片の集合であり，全体のアンサンブルとして絵や模様を表現する．本研究では小石モザイクや石畳の模様の生成に主眼を置く．その背景には前章で述べたような理由があるが，具体的にこれらを研究対象とした理由は次の通りである：

1. 構成要素となる石の形状は凸凹の形状が各々異なっているため，手作業でのモデリングは負荷が高い
2. 上記理由から，そのアンサンブルとして絵や模様を表現したい場合，さらに制作が困難になり作業負荷が高くなる
3. これまでにも石モデルのプロシージャルな生成手法についてはいくつか報告されているが，ユーザがその配置をインタラクティブに指定してデザインや模様を表現する手法に関しての報告は，著者の知る限りない
4. またモザイク画生成の観点からは，その生成手法の多くは正方形タイルの敷き詰めに関しての最適化問題として扱われており，石のような異方性形状の配置はあまり報告されていない．さらに，モザイク画生成手法もまた，インタラクティブにユーザが配置パターンを指定する手法はこれまで提案されていない

本研究では小石モザイクのように石を構成要素とし，その配置パターンにより模様を表現するモデルのプロシージャル生成手法を提案する．ユーザが入力した画像の特徴 (エッジ) を抽出し，その特徴を反映した石の配置を計算し，最終的に 3 次元形状の石のモデルを生成する．またユーザはスケッチインターフェースを通して石の配置を制御する “流れ場” の編集を行ったり，各種パラメータにより任意の形状を得ることが可能である．ユーザのコントローラビリティと作業負荷はトレードオフにあると考えられるが，本研究では，ユーザにはパラメータ設定や，全体の配置の流れをスケッチインターフェースを通して制御可能する．その一方で，石の形状の生成等をシステムで計算することでユーザの作業負荷を低減させつつ，ユーザの意思を反映させることが可能になると考える．

1.3 本論文の構成

本論文の構成は以下のようになっている．2 章では実際の小石モザイクに関しての概要と CG における関連研究を紹介する．関連する先行研究は，“プロシージャルモデリング” と

“モザイク画生成手法” に分類する．3 章では本研究での提案手法のメインのアルゴリズムを述べる．4 章では，3 章で生成した小石モザイクに装飾を施す方法に関して述べ，5 章で結果と考察を述べる．最後に 6 章でまとめと今後の課題について述べる．

第 2 章
関 連 研 究

本章では，まず実際の小石モザイクについて述べ，その後プロシージャルモデリング手法とモザイク画生成手法に関する先行研究を紹介する．

2.1 実際の小石モザイク

まず，実際の小石モザイクについてその概要を説明する．小石モザイクは小石を断片とするモザイク画である．文献 [Howarth 2009] では主に動植物や幾何模様をデザインとして用いているのが確認できる．一例を図 2.1 に示す．実際の小石モザイク製作にあたっては，下絵のデザインを決定する必要がある．コンピュータを用いた下絵デザインの方法に関しても，文献 [Howarth 2009] で述べられている．手書きで書いたデザイン案をスキャナでコンピュータに取り込み，ベクター形式で保存する．このとき，下絵の各領域にどのような石 (種類・形状) をどういった配置の仕方で並べるかといった情報も付与する．最終的に実寸で印刷した下絵をもとに石を並べていく．下絵のデザイン例を図 2.2，そしてその一部の領域の拡大図を図 2.3(a)，図 2.3(b) に示す．図 2.3(a)，図 2.3(b) では，石の種類や形状，配置の仕方が書き込まれているのが確認できる．

これらの例から，ユーザは任意の流れを指定して石を並べていくといった工程を踏むであろうことがわかる．それは必ずしも計算機上で得られる最適化された配置とは限らない．また領域ごとに配置方法も異なっている．このような配置パターンはユーザがインタラクティブに指定するのが望ましいと考えられるが，先行研究ではこれを達成することは困難である．

次に，実際の小石モザイクに用いられる石の例を図 2.4 に示す．これらは主に小石モザイクの前景に用いられることが多い．図 2.4(a) や図 2.4(b) のように楕円形をしたものや，図 2.4(c) のように円形のものなどがある．いっぽうで，前景と背景の差別化をはかるために背景には図 2.5 に示すように形の不揃いな石が用いられることがある．図 2.5(左) の石は採石場から切り出された石を砕いて得たもので，領域を敷き詰めるのに適している．図

2.5(右) の石は表面形状がフラットな石であり，図 2.4(c) の丸石とは異なり，不揃いな輪郭をしている．それゆえ，丸石とは異なる視覚効果を与える．

本研究では，丸石と形状の不揃いな石の両方を，アルゴリズムにほとんど修正を加えることなく生成することが可能である．3 章では全体のアルゴリズムを図 2.4(a) や図 2.4(b) に示す丸石の生成手法とともに述べ，4 章では，図 2.5 に示す形状の石を生成し，それを小石モザイクの背景に適用する手法について述べる．

図 2.1: 実際の小石モザイク
[Howarth 2009] より引用．

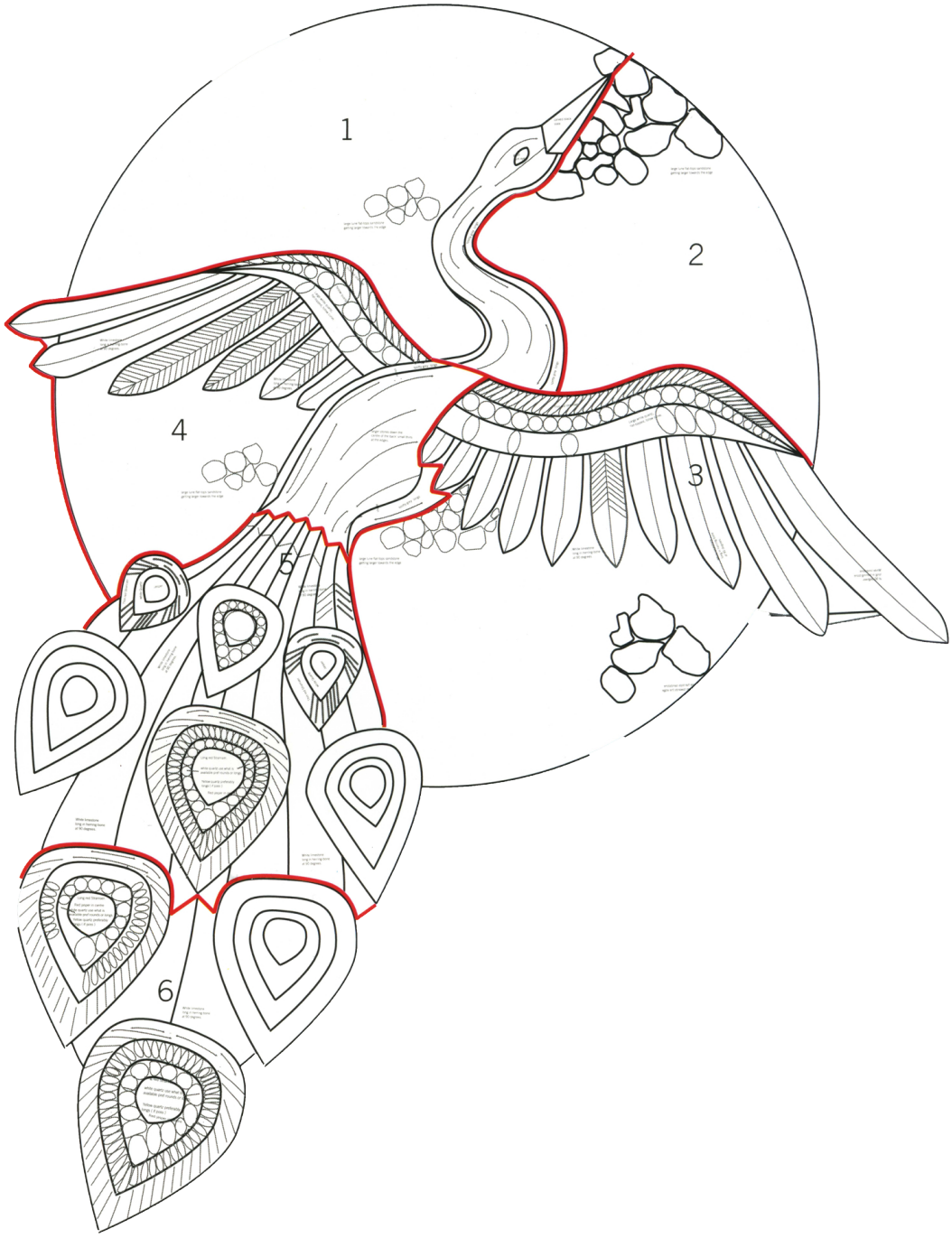
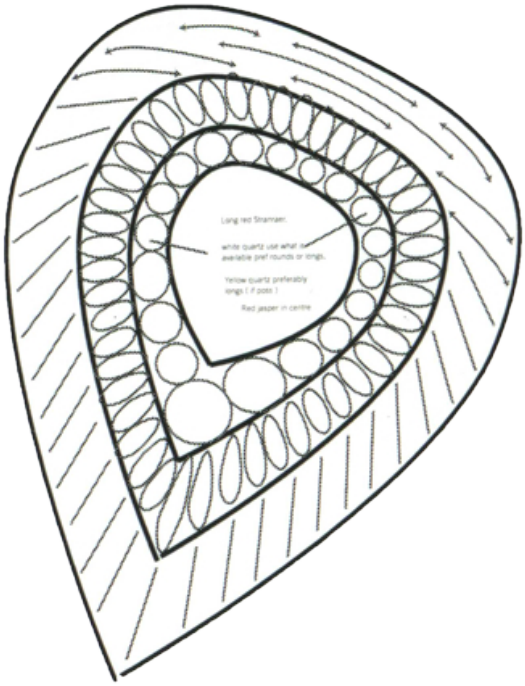
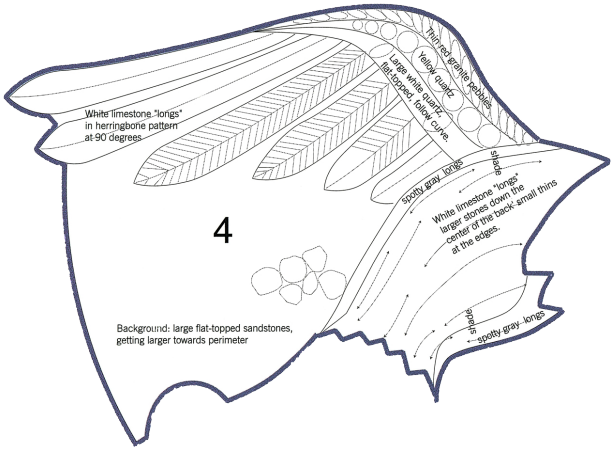


図 2.2: 実際の小石モザイクの下絵デザイン
[Howarth 2009] より引用.



(a)



(b)

図 2.3: コンピュータを用いたモザイク画制作

(a),(b)：[Howarth 2009] より引用.

(a) (b) (c)

図 2.4: 実際の小石モザイクに使われる丸石

(a),(c)：[Howarth 2009] より引用．(b)：注釈参照のこと[1]．

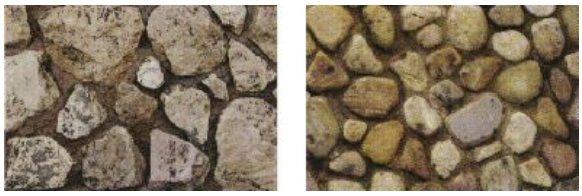


図 2.5: 実際の小石モザイクに使われる石
[Howarth 2009] より引用．

2.2 プロシージャルモデリング

ここでは，プロシージャルモデリングをさらに細分し，ユーザ介入による指定を有する大規模なモデリングに関する先行研究と，石のモデリングに関する先行研究を紹介する．

2.2.1 インタラクティブな大規模モデリング

複数のモデルの集合からなる大規模なモデルの制作に際して，1.1 章で述べたように，ユーザ介入によるプロセス制御は重要な要素となる．

地形生成：Zhou らはシステムの入力として山稜を指定するユーザスケッチと実際の渓谷等の高さマップを与えることで，実際の尾根形状の特徴を反映した山稜をユーザスケッチに沿って生成する手法を提案した [Zhou et al. 2007]．これはテクスチャ合成を応用した手法であるが，同様にテクスチャ合成を応用した手法で，山岳形状生成以外の例としては，Ijiri らは参照画像中のエレメントの配置を解析し，その接続情報を基に，インタラクティブにユーザストロークに沿ってエレメントの配置を行うことが可能なペイントシステムを

[1]©2010 Pandorea... All rights reserved, http://www.flickr.com/photos/jollyroberts/1590802556/.

提案した [Ijiri et al. 2008]．また，インタラクティブにユーザが山岳形状のシルエットをスケッチすることでモデリングする手法 [Gain et al. 2009] や，ユーザが山稜や川，あるいは砂漠の砂紋の様な特徴線を入力することで地形生成が可能な手法 [Hnaidi et al. 2010] などが提案されている．

景観生成：野村らは大規模工場の景観の自動生成手法を提案した [Nomura and Miyata 2010]．この手法では建造物全体のシルエットを一筆書きで入力することで工場の概形を，そしてパラメータにより工場を構成する各部位のレイアウトを制御することができ，生成される工場にバリエーション付けすることができる．

Parich と Müller は都市の道路ネットワークを生成するシステムを提案した [Parish and Muller 2001]．彼らの CityEngine と名付けられたシステムは，陸海の別を指定した 2 値マップや高度マップ，人口密度等を入力として与えることで道路ネットワークを生成する．道路で分割された区画には自動的に高層ビルを生成する．また，Chen らはテンソル場から道路ネットワークを生成する手法を提案した [Chen et al. 2008]．彼らのシステムは陸海の別を指定した 2 値マップから抽出した境界線や高さマップからテンソル場を計算する．ここで，ユーザがインタラクティブにテンソル場を編集することも可能である．そして編集されたテンソル場に沿って配置した hyperstreamline を基に道路グラフを生成する (図 2.6)．

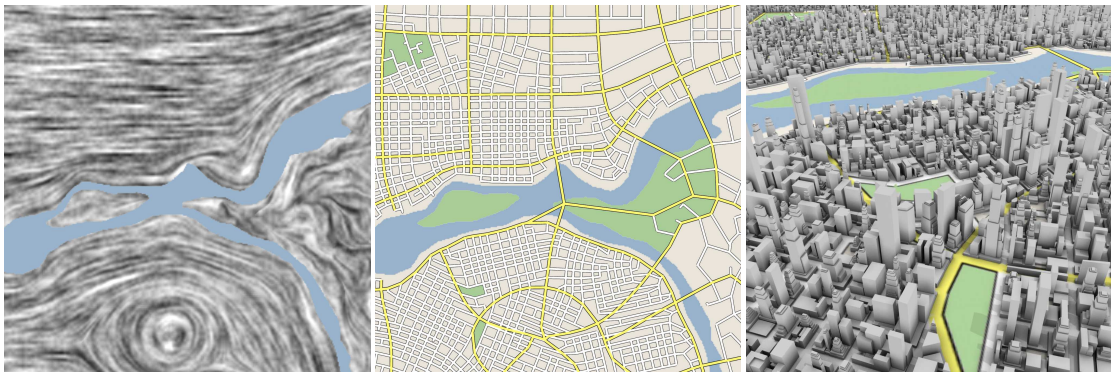


図 2.6: Interactive Procedural Street Modeling

(左)：ユーザはインタラクティブにテンソル場を編集する．(中央)：テンソル場から計算されて道路ネットワーク．(右)：道路ネットワークに基づいて生成された都市景観． 画像は [Chen et al. 2008] より引用．

2.2.2　石のモデリング

石のモデリング手法に関して，これまでにいくつか提案されている．Miyata は 2 次元グラフから石の形状を構成し，石垣テクスチャを生成する手法を提案した [Miyata 1990]．また，Miyata らは力学シミュレーションにより，正方形粒子を与えられた領域内に最密充填することで敷石の敷き詰めパターンを生成し，それに基づいて各四角形粒子に 3 次元形状を付加することで石畳テクスチャを生成する手法を提案した [Miyata et al. 2001](図 2.7)．これらの手法では 2D グラフあるいはシミュレーション結果から石の 2 次元的な配置パターンを生成し，その後その 2 次元形状を基に高さを付加することで 3 次元の石形状を生成している．また，Itoh らは [Miyata et al. 2001] の手法を応用し，与えられた領域を疑似ボロノイ分割することで異方性を持たせ，その上で 3 次元形状を付加することで，ワニやトカゲの皮膚のように有機的な表面形状を生成する手法を提案した [Itoh et al. 2003].

また，Peytavie らは Coner Cube と呼ばれる 3 次元の立方体内部にボロノイ図を生成し，その各セルに侵食処理を施したものを，Wang Tile と似た要領でタイリングすることで非周期的な岩石の積み重なりを再現するモデルの生成手法を提案した [Peytavie et al. 2009]. 彼らの手法では，Coner Cube との交差判定を行うことで任意の 3 次元形状を表現するかたちで岩石をインスタンス化することが可能である．Peytavie らの手法では大きさの大きく異なる岩石の混在をうまく表現することができなかったが，櫻井らは複数のボロノイ図から排他的にセルを選択することでこれを達成する手法を提案した [櫻井 and 宮田 2010].

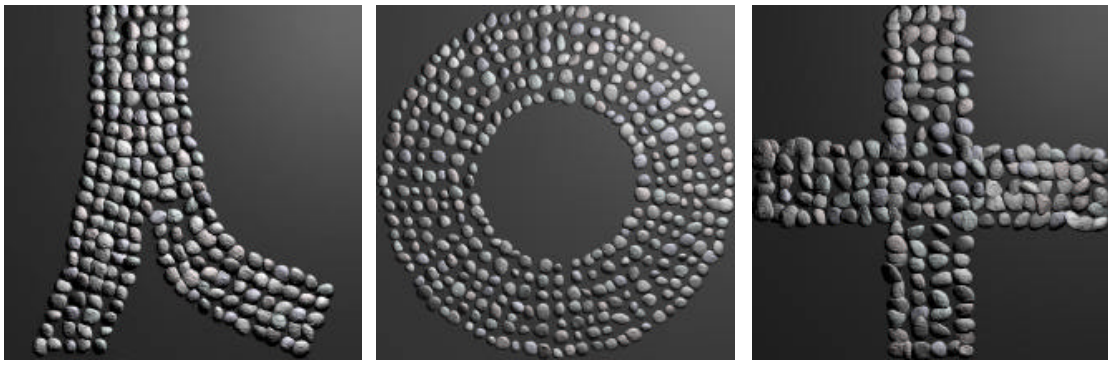


図 2.7: A Method for generating pavement texture using the square packing technique

力学シミュレーション結果から生成された配置パターンから石畳モデルを生成する．画像は [Miyata et al. 2001] より引用.

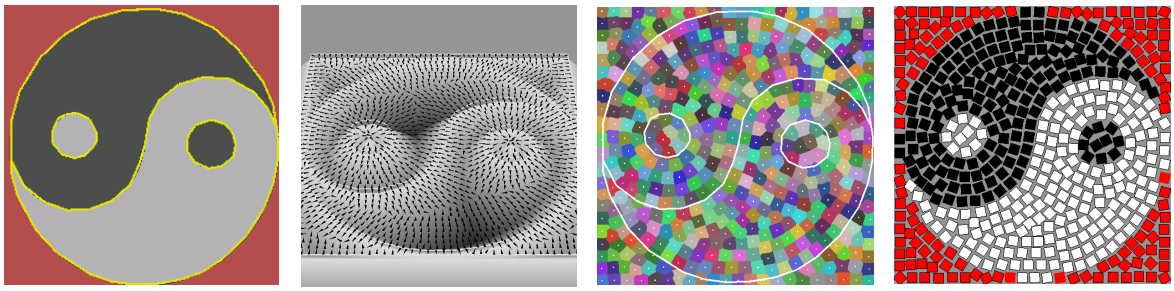
2.3　モザイク画生成手法

CG において古典的モザイク画生成手法を扱う際，次のような最適化問題として捉える場合がしばしばある：

問題 2.3.1 (CG におけるモザイク画生成) 平面中の領域 $\boldsymbol{R} \subset \mathbb{R}^2$ と，制約条件が与えられたとき，次の条件が成り立つ N 個の母点 $\boldsymbol{P}_i \subset \boldsymbol{R}$ に N 個のタイルを配置する．

- すべてのタイルは互いに素である *(*交わらない*)*
- 覆う面積が最大になる
- 制約条件を可能な限り満たす

このような最適化に関するモザイク画の生成手法に関して，これまでにいくつかの手法が考案されてきた．Hausner は重心ボロノイ (CVD) からモザイク画を生成する手法を提案した [Hausner 2001]．重心ボロノイをマンハッタン距離で計算することで四角形タイルを形成する．各タイルは入力画像から得られたエッジ情報を基に生成された距離場により方向付けされる．さらに，重心ボロノイを最適化する上で，エッジ周辺を計上しないステップを取り入れることでボロノイセルがエッジに沿うようにしている (図 2.8)．また Liu らはグラフカット法に基づいた最適化手法により，陽にエッジ検出を用いないでモザイク画を生成する手法を提案した [Liu et al. 2007, 2010]．Battiato らはエネルギー関数を最適化することで方向場を計算し，その向きにタイルを沿わせる手法を提案した [Battiato et al. 2008]．



(a)　(b)　(c)　(d)

図 2.8: Simulating Decorative Mosaics
(a)：入力画像．(b)：(a) から生成した方向場．(c)：重心ボロノイ (CVD) の最適化．(d)：モザイク画生成結果．画像は [Hausner 2001] より引用．

また，Dobashi らはボロノイセルと入力画像の色の SSD(Sum of Squared Difference) を計算し，ボロノイ図の母点を移動するという最適化により結果を得る [Dobashi et al. 2002]．

これは上記の様な最適化問題ではなく，不定形タイルを用いて，そのタイルの輪郭を参照画像の色領域に沿うような最適化を行っている．最適化を行わない手法として，Blasi らは入力画像から抽出したエッジとボロノイ図を合成することで最終的なモザイクセルを生成した [Blasi et al. 2005, di Blasi and Gallo 2005]．Battiato らはマスク画像を入力とすることで前景・背景で異なる並べ方をする種のモザイク画 (Opus Vermiculatum) を生成する手法を提案した [Battiato et al. 2006]．Faustino らは重み付きボロノイを計算することで，入力画像の特徴を反映したモザイク効果を生成している [Faustino and de Figueiredo 2005]．多くの手法がボロノイ図を利用しているが，Elber らは画像から得られた特徴線に平行なオフセットラインを算出し，この曲線にモザイク片を並べることでモザイク画を生成する手法を提案した [Elber and Wolberg 2003]．また，Noda らは 3D モデルから抽出したエッジからオフセット曲線を求めてモザイク片を並べる手法を提案している [Noda and Miyata 2009]．3D モデルを入力とすることでユーザは任意のアングルの下絵を容易に得ることができるため，下絵デザインの候補選定に便利であると言える．

他方で，3D モデルの表面にモザイクタイルを装飾する手法についてもいくつか報告されている [Lai et al. 2006, Dos Passos and Walter 2008, 2009, Battiato and Puglisi 2010]．Lai らは 3D モデルの表面に正方形タイルを配置する手法を提案した [Lai et al. 2006]．これを応用して，Dos Passos らにより表面の曲率を考慮して配置するタイルの大きさや配置を可変にする手法，あるいは正方形タイルとボロノイ図から生成した不定形タイルの混在したモザイクタイルを配置する手法などが提案されている [Dos Passos and Walter 2008, 2009]．

第 3 章
アルゴリズム

本手法で提案するアルゴリズムの概要は次にようになっている：

1. 入力画像からエッジを抽出する
2. エッジ情報から距離場を求める
3. 方向場を生成し，流れ場を初期化する
4. ユーザによる流れ場の対話的編集
5. Poisson-Disk 分布により石ボリュームの生成点を分布させる
6. 配置した生成点に石ボリュームを生成する

以下，アルゴリズムの詳細について述べる．

3.1 エッジ抽出と距離場の生成

まず，入力画像 (図 3.1) からエッジを抽出する．エッジの抽出は入力画像に対してラプラシアンフィルタを適用することで行う．画像 I の各ピクセル $(x, y) \in \mathbb{R}^2$ におけるラプラシアン $\triangle$ は次のように書かれる：

$$\triangle I(x,y) \equiv \nabla^2 I(x,y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) I(x,y) \tag{3.1}$$

画像処理においては，あるピクセルにおける 8 近傍のラプラシアンを計算する際には，次のような 3×3 のカーネル

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.2}$$

を計算する．さらにラプラシアンフィルタにより得られた画像を 2 値化しておく．このようにして抽出されたエッジを図 3.2 に示す．次に，抽出したエッジから距離場 $D_{distance}(\boldsymbol{p}) = D(x, y)$ を生成する．距離場は，各ピクセル $\boldsymbol{p} = (x, y) \in \mathbb{R}^2$ とそこからエッジの最近接点までのユークリッド距離を計算して求める．生成した距離場を図 3.3 に示す．距離場はグレースケールの画像として表示され，その値の範囲を $D_{distance}(\boldsymbol{p}) \in (0, 255)$ とすると，値が 0 に近いほど黒く，逆にエッジから遠ざかるほど白く表示される．

図 3.1: 入力画像

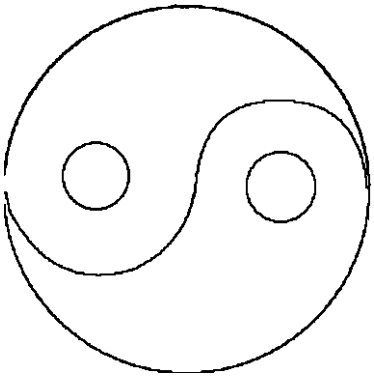


図 3.2: 抽出されたエッジ

図 3.3: 距離場

3.2 流れ場の生成

ここでは，流れ場の生成手法について述べる．

1. 入力画像から方向場を計算する (方向場生成)
2. ユーザが任意で流れ場を編集する (ストロークからの流れ場生成)

流れ場とは，石の形状の異方性をその向きに沿わせるための場のことである．流れ場は上記のどちらか一方，もしくは両方を用いて生成することができる．すなわち，システム側で自動的に算出する方法とユーザが編集する方法がある．また，双方を用いる場合は，システム側で自動的に算出したベクトル場に対してストロークインターフェースを用いて編集する．システムが自動的に算出する場合には方向場を用いる．これはモザイク画の生成手法ではたびたび用いられる方法であり，モザイク片の方向を求めるために使用される．

3.2.1 方向場生成

方向場 $\phi(\boldsymbol{p})$ は，各ピクセル $\boldsymbol{p} = (x, y)$ におけるエッジの接線方向を向くベクトルで構成される場である．エッジの接線方向を計算するために，まず距離場 (図 3.3) からグラディエントを計算することで，エッジの法線方向を向くベクトルを求める．

$$V(\boldsymbol{p}) = V(x, y) = e^{-dD_{distance}(\boldsymbol{p})} \nabla \boldsymbol{p} \tag{3.3}$$

ここで，係数 $exp(-dD_{distance}(\boldsymbol{p}))$ は，エッジからの距離 $D_{distance}(\boldsymbol{p})$ に応じてベクトルの大きさを減衰させるための係数であり，d は減衰定数である．図 3.4 にエッジ法線方向ベクトル場を可視化した例を示す．また，エッジの接線方向は， 法線方向ベクトルの向きを反時計まわりに回転することで得る．図 3.5 にエッジ接線方向ベクトル場を可視化した例を示す．ただし図 3.4，図 3.5 の矢印の色はベクトルの大きさを表しており，ベクトルの大きさを正規化したものを HSB 色空間の色相に割り当てたものである．この際，彩度と明度は 100% で固定値とした．多くのモザイク画生成手法では法線方向を方向場として採用している．これはタイル (モザイク片) の方向をシステムで計算するためのものであるが，本手法では接線方向を用いる．これは，後にユーザが編集する際，ユーザはエッジに平行にストロークを描くことで流れを指定することが予想されるためである．

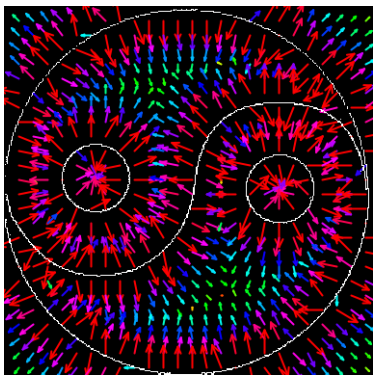


図 3.4: エッジの法線方向を向くベクトル場

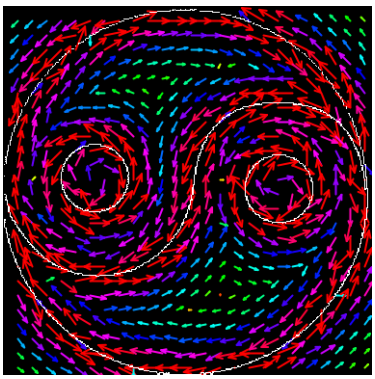


図 3.5: エッジの接線方向を向くベクトル場

3.2.2 ストロークからの流れ場生成

ユーザは任意のストロークを描くことで流れ場 (ベクトル場) を編集することが可能である．ユーザが描いたストロークは等間隔にサンプリングすることで等しい長さを持つセグメントの集合として表現することができる (以下，ストロークセグメントと呼ぶ)，すなわちストローク S を n 個のストロークセグメント s_i の集合 $\{s_i \mid 0 \leq i \leq n-1\}$ とする．さらに，ストロークセグメントの始点をデザインエレメントと呼ぶことにする．各デザインエレメントにおけるベクトルの大きさは，セグメントの始点から終点の向きでセグメントの長さに比例する値として設定する．そして，各ピクセルにおけるベクトル場の値は [Zhang et al. 2006] と同様に式 (3.4) で計算する．

$$V(\boldsymbol{p}) = \sum_i e^{-d\|\boldsymbol{p}-\boldsymbol{p_i}\|^2} V_i(\boldsymbol{p}) \tag{3.4}$$

ここで，$\boldsymbol{p}$ は注目するピクセル，$\boldsymbol{p_i}$ は i 番目のデザインエレメント，$V_i(\boldsymbol{p})$ は $\boldsymbol{p}$ における $\boldsymbol{p_i}$ のベクトル場の値，d は減衰定数である．これは，あるピクセルのベクトル場の値が各デザインエレメントの寄与を合計することで決定されることを表している．図 3.6 に生成例を示す．

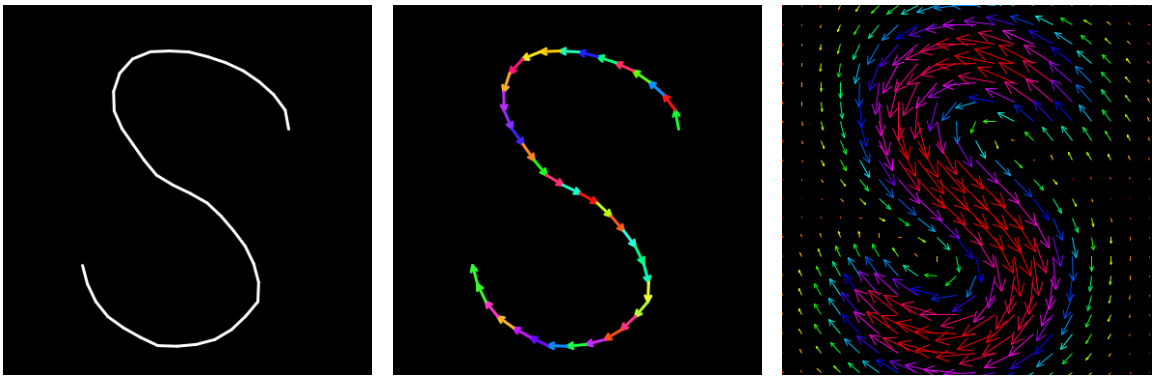


図 3.6: ストロークインターフェースによるベクトル場の編集
左：ユーザ入力によるストローク．中央：ストロークセグメントの集合表現．右：ストロークから生成されたベクトル場の可視化．

また，ユーザはベクトル場の影響範囲を制限することが可能である．本手法では，0 番目のデザインエレメント (p_0) の属する色領域と同じ色領域の寄与のみ計算に考慮することでベクトル場の影響範囲を制限することができる．これにより領域別に流れを形成することが可能となる．図 3.7 に例を示す：

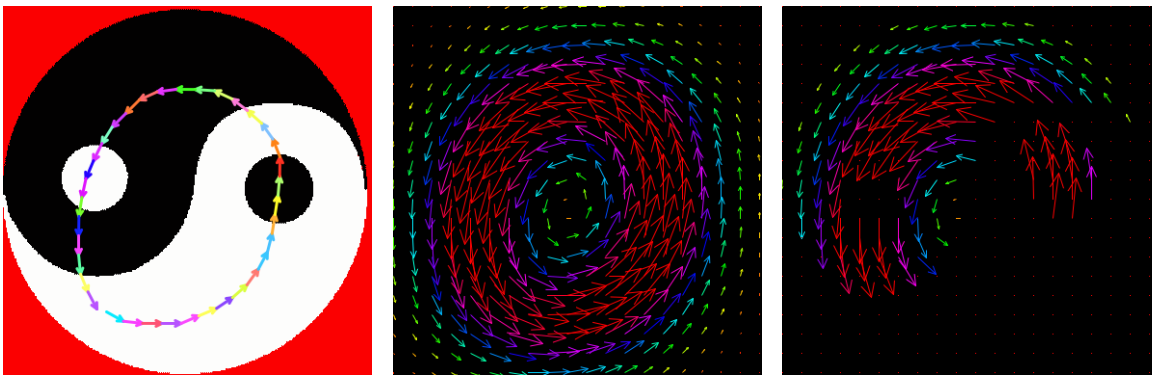


図 3.7: 領域制限の有無
左：ユーザ入力によるストローク．中央：領域制限がない場合のベクトル場の可視化．右：領域制限 (黒色領域) がある場合ベクトル場の可視化．

3.2.3 回転場の生成

通常のストロークから生成されるベクトル場はそれぞれのデザインエレメントからの足し合わせであり，回転場を生成したい場合に問題になることがある (図 3.8)．

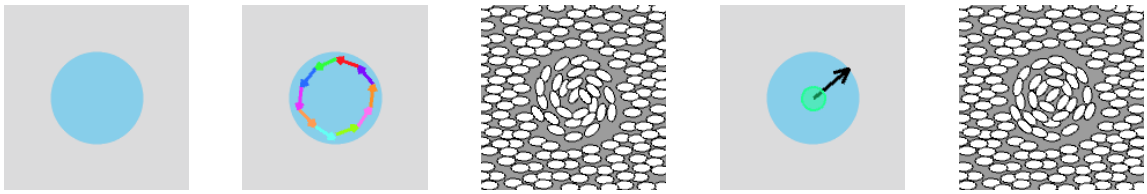


図 3.8: 回転場の導入
(a)：参照画像．(b)：通常のストロークセグメントから生成したデザインエレメント．(c)：(b) のデザインエレメントの寄与の足し合わせにより生成した小石の配置結果．(d)：回転場．(e)：(d) の回転場から生成した楕円 (小石) の配置結果．

図 3.8(a) を参照画像とした場合を考える．このとき，領域制限のもと中央の円に沿うような流れ場を生成したい場合，これまでの様にストロークを描く (図 3.8(b)) ことで図 3.8(c) のような配置結果を得る (配置手法に関しては後述)．この方法で生成される流れ場は各デザインエレメントの寄与の足し合わせとなるため，中央付近では意図した流れを作る

ことが困難である．図 3.8(c) では，円の内側の配置結果が渦を巻いたような状態になっているのが確認できる．ユーザの意図としては円の特徴を表現したいので，その流れ場は同心円状になっていることを期待する場合もある．ここでは，そのような流れ場を生成するため，“回転場” を生成する方法を考える．図 3.8(d) は回転場のユーザ指定の方法を表している．通常のストロークと区別するため，ストロークの始点を中心に半透明の円を合わせて表示することでこれが回転場を指定するものであることを表す．図 3.8(e) は以下で説明する生成方法で生成した回転場により得られた配置結果である．図 3.8(c) と比較して，中心付近でも楕円 (小石) の向きが同心円状になっているのが確認できる．

いま，ユーザの 1 ストローク (図3.9) において，ストロークの始点を回転場の中心点 $q(x_q, y_q)$，ストロークの始点から終点までの直線距離を回転場の影響半径 r とする (図 3.10)．

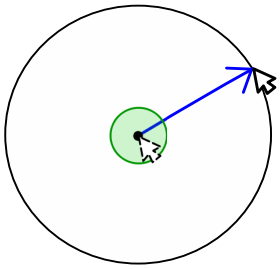


図 3.9: 入力ストローク

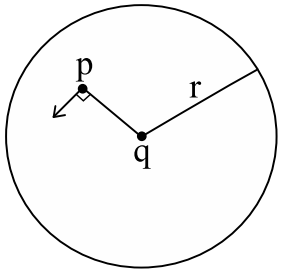


図 3.10: 点 p における回転場の計算

このとき，点 q を中心とした半径 r の円内の各ピクセル $p(x_p, y_p)$ におけるベクトル場を考える．点 p における法線方向単位ベクトルを $norVec(p) = (x_{norVec}, y_{norVec})$ とすると，$norVec(p) = \overrightarrow{qp}/|\overrightarrow{qp}|$ となる．また，点 p における回転場を $tanVec(p) = (x_{tanVec}, y_{tanVec})$ とすると，$tanVec(\boldsymbol{p})$ は点 $\boldsymbol{p} = (x, y)$ において $norVec(\boldsymbol{p})$ と直交する．すなわち，点 $\boldsymbol{p}$ における回転場の方向 θ は，

$$\theta = tan^{-1}\left(\frac{y_{tanVec}}{x_{tanVec}}\right) \tag{3.5}$$

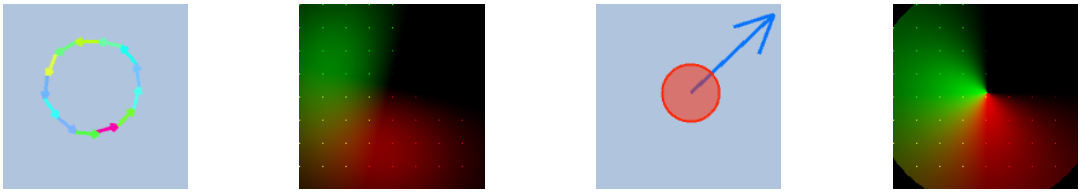
$$= tan^{-1}\left(\frac{y_{norVec}}{x_{norVec}}\right) - \frac{\pi}{2} \tag{3.6}$$

となる．また大きさに関しては式 (3.7) により決定する：

$$\|tanVec(\boldsymbol{p})\| = s \cdot exp\left(-\frac{d}{r}\right) \tag{3.7}$$

ここで，s はスケールファクター，d は点 p と点 q の間の距離とする．これは，回転場が中心から遠ざかるに従い，その値の大きさが減衰していくことを表している．これはストロークセグメントで円を描いた場合に得られるベクトル場とは減衰の仕方が逆である．

また，矢印を用いた可視化ではそのベクトル場がなめらかに変化しているかどうかは確認が困難である．そこで，その効果を検証するために色で可視化した例を次に示す (図 3.11)：

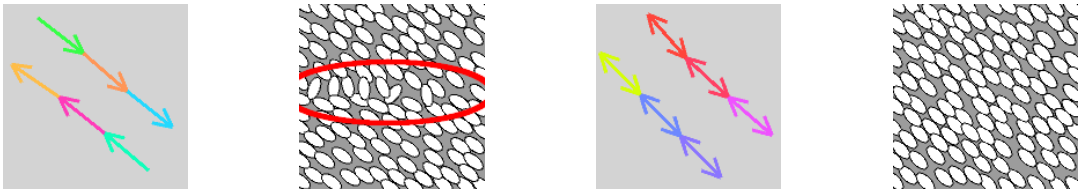


(a) (b) (c) (d)

図 3.11: 回転場の効果の色による検証
(a)：通常のストロークで描いた円．(b)：(a) のベクトル場の色による可視化．(c)：回転場．(d)：(c) のベクトル場の色による可視化．

これは，各ピクセルにおけるベクトル場の大きさ $V(V_x, V_y)$ を $(0, 255)$ に 規格化し (それぞれ V'_x, V'_y)，そのピクセルの RGB カラーに $(R, G, B) = (V'_x, V'_y, 0)$ を代入して表示したものである．図 3.11(b) より図 3.11(d) の方が色がなめらかかつリニアに変化している様子が確認できる．

3.2.4　ベクトル場を用いる際の問題点



(a)　(b)　(c)　(d)

図 3.12: ベクトル場による編集の問題点とテンソル場による解決策
(a)：ベクトル場のデザイン．(b)：ベクトル場による配置結果．(c)：テンソル場のデザイン．(d)：テンソル場による配置結果．

これまでベクトル場を用いた流れの生成手法に関して説明してきた．しかし，ベクトル場を式 (3.4) に従って足し合わせる際問題になることがある．それは，互いに反対方向を向くベクトル場をデザインした場合，双方の足し合わせにより中間部分で意図しない方向のベクトル場を生成してしまうことである．例えば，図 3.12(a) のようなストロークを入力とした際，生成されるベクトル場に従い算出した配置結果は図 3.12(b) のようになる．図中の中央部分で意図しない方向を向く配置結果になっているものが見られる．これはベクトル値の足し合わせの際，反対の成分同士が打ち消し合うなどの結果として生成されるもので，不可避的に発生すると考えられる．この問題は [Zhang et al. 2007] でも指摘されているが，本研究においても同様の問題が発生することが発覚した．さらに，[Zhang et al. 2007] では絵画調レンダリングの例を取り上げているが，絵画調レンダリングではストロークを重ねているため，違和感はそれほど大きくない．一方で本研究の場合には要素は排他的に配置するため，その見た目の違和感は大きくなることが予想される．

そこで，本手法でも [Zhang et al. 2007] を参考にテンソル場の導入を試みた．図 3.12(c) のようなテンソル場をデザインした際，図 3.12(d) では，図 3.12(b) の問題が解決されているのが確認できる．これはテンソル場の性質に起因する．テンソル場は双方向的な大きさと向きから成るため，各々のデザインエレメントの足し合わせの際，ベクトル場のように一方向の向きを持つデザインエレメントの足し合わせで発生するような，意図しない生成結果を生じることはない．以下では，テンソル場の生成手法に関して説明する．

3.2.5　テンソル場の生成

本項では，まずテンソル，特に本手法で用いる 2 階の対称テンソルの概要を説明したあと，本手法での応用について説明する．

テンソル場の概要

2 階のテンソルは次のように定義される：

定義 3.2.1 (2 階テンソル) 任意のベクトル $\boldsymbol{x}$ に対してベクトル $T(\boldsymbol{x})$ を対応させる対応 T について線形性 *(linearity)*

$$T(a\boldsymbol{x} + b\boldsymbol{y}) = aT(\boldsymbol{x}) + bT(\boldsymbol{y}) \tag{3.8}$$

が任意の数 a, b と任意のベクトル $\boldsymbol{x}, \boldsymbol{y}$ に対して成り立つとき，T をテンソル *(Tensor)* という *[石原 1991]*.

また，直交座標系 Σ の基本ベクトル $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ とすると，3 つのベクトル $T(\boldsymbol{e}_1), T(\boldsymbol{e}_2), T(\boldsymbol{e}_3)$ を $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ の 1 次結合として次のように表すことができる.

$$\begin{cases} T(\boldsymbol{e}_1) = T_{11}(\boldsymbol{e}_1) + T_{21}(\boldsymbol{e}_2) + T_{31}(\boldsymbol{e}_3) = T_{i1}(\boldsymbol{e}_i) \\ T(\boldsymbol{e}_2) = T_{12}(\boldsymbol{e}_1) + T_{22}(\boldsymbol{e}_2) + T_{32}(\boldsymbol{e}_3) = T_{i2}(\boldsymbol{e}_i) \\ T(\boldsymbol{e}_3) = T_{13}(\boldsymbol{e}_1) + T_{23}(\boldsymbol{e}_2) + T_{33}(\boldsymbol{e}_3) = T_{i3}(\boldsymbol{e}_i) \end{cases} \tag{3.9}$$

このとき，$9(= 3^2)$ 個の数の組 $T_{11}, T_{12}, ..., T_{33}$ をテンソル T の直交座標系 Σ に関する成分 (component) という．T の成分は行列では次のように書かれる：

$$[T_{ij}] = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \tag{3.10}$$

以降では 2×2 の行列で書かれる次のような 2×2 テンソルを考える：

$$[T_{ij}] = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \tag{3.11}$$

ここで，$T_{ij} = T_{ji}$ のときテンソル $[T_{ij}]$ を対称テンソルと呼ぶ.

対称テンソルの性質 (固有ベクトルの直交性)

対称テンソルの性質として，対称テンソルの固有ベクトルは直交する．2 階のテンソル T_{ij} の固有値を λ，対応する固有ベクトルを $\boldsymbol{A}$ と書くと，これらは次式を満たす.

$$T_{ij}A_i = \lambda A_j \tag{3.12}$$

ここで，2 階の対称テンソル T_{ij} を考える．

$$T_{ij} = T_{ji} \tag{3.13}$$

T_{ij} の相異なる 2 つの固有値を λ_p, λ_q，そのそれぞれに対応する固有ベクトルを A_i^p, A_i^q と書く．定義より

$$T_{ij}A_j^p = \lambda_p A_i^p \tag{3.14}$$
$$T_{ij}A_j^q = \lambda_q A_i^q \tag{3.15}$$

式 (3.14) の両辺に A_i^q，式 (3.15) の両辺に A_i^p を掛け，辺々引くと，

$$T_{ij}A_j^pA_i^q - T_{ij}A_j^qA_i^p = (\lambda_p - \lambda_q)A_i^pA_i^q \tag{3.16}$$

ここで，左辺は式 (3.13) より 0 となる．ところで右辺では $\lambda_p \neq \lambda_q$ を仮定していたので，$A_i^pA_i^q = 0$ でなければならない．これは $\boldsymbol{A^p}$ と $\boldsymbol{A^q}$ の内積が 0，すなわち固有ベクトルが直交していることを意味している □

これは，可視化の際に都合がよい．すなわち，2 つの固有ベクトルは互いに直交しているので主固有ベクトルのみを可視化すれば，他方はそれに直交する方向となるが，これはちょうど楕円の長軸と短軸と対応付けて考えられるためである (図 3.13(b))．本手法では，図 3.13(a) に示す v_1 を楕円の長軸方向とし，v_2 を楕円の短軸の方向とする．

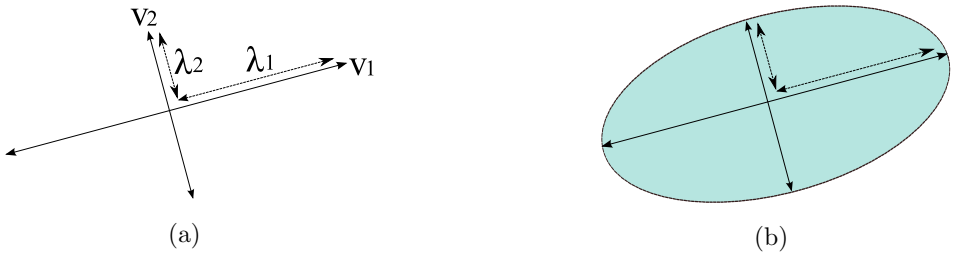


図 3.13: テンソル場の直交する 2 固有ベクトルと楕円形 Disk の対応
(a)：テンソル場の直交する 2 固有ベクトル．(b)：テンソル場の固有ベクトルと楕円形 Disk．

本手法で用いるテンソル場

本手法では次のような 2×2 の対称テンソルを用いる:

$$\mu \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix} \tag{3.17}$$

ここで $\mu \geq 0$，$\theta \in [0, 2\pi)$ である．このとき，2 つの相異なる固有ベクトルに関して，主固有ベクトルは，

$$\left\{ \lambda \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \middle| \lambda \neq 0 \right\} \tag{3.18}$$

また他方の固有ベクトルは，

$$\left\{ \lambda \begin{pmatrix} \cos(\theta + \frac{\pi}{2}) \\ \sin(\theta + \frac{\pi}{2}) \end{pmatrix} \middle| \lambda \neq 0 \right\} \tag{3.19}$$

となる．これらの固有ベクトルは $\pi/2$ の角度を成す，すなわち直交していることが確認できる．

生成するテンソル場の種類

テンソル場 T は画像中の各々のピクセル $\boldsymbol{p} = (x, y)$ で値 $T(\boldsymbol{p})$ を持つ．$T(\boldsymbol{p}) = 0$ のとき $\boldsymbol{p}$ を縮退点 (degenerate point) と呼ぶ．またそれ以外は正則 (regular) と呼ぶ．本手法ではベクトル場のときの場の生成パターンと似たパターンを生成するため，Grid パターンと Radial パターンのテンソル場を用意する [Chen et al. 2008]．これらのテンソル場は [Zhang et al. 2007] で提案されたものであるが，Chen らはこれらのテンソル場を道路網の生成に適用した．Chen らは道路網が多くの場合，街区ではグリッド状，またパリの凱旋門のようなモニュメントの周辺では放射状になっている様子から特にこれらのパターンを用いた．本手法では，ベクトル場の生成手法に対応させるかたちで Chen ら同様，Grid パターン (通常のベクトル場生成) と Radial パターン (回転場生成) を用いるのが適切だと判断した．

Grid パターン

Grid パターンは点 $\boldsymbol{p}_0$ において方向ベクトル (u_x, u_y) が与えられたとき，$l = \sqrt{u_x^2 + u_y^2}, \theta = tan^{-1}\left(\frac{u_y}{u_x}\right)$ として，次のようなテンソル場から生成される [Zhang et al. 2007]：

$$T(\boldsymbol{p}) = l \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix} \tag{3.20}$$

次に Grid パターンの生成例を示す．図 3.14 に示すように，テンソル場は双方向的であるため，片矢印ではなく両矢印を用いる．このようなユーザストロークからは図 3.15 のように，ストロークに並行な方向に主固有ベクトルを持つテンソル場が生成される．このテンソル場により生成される配置結果を図 3.16 に示す．なお，図 3.15 は 2 種類の方向の流れを持つノイズを合成して作成した概念図である (図 3.18 も同様に Radial パターンの概念図)．

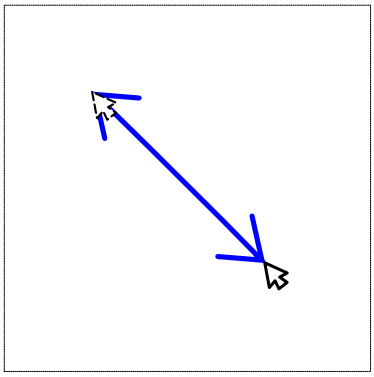


図 3.14: ユーザストローク

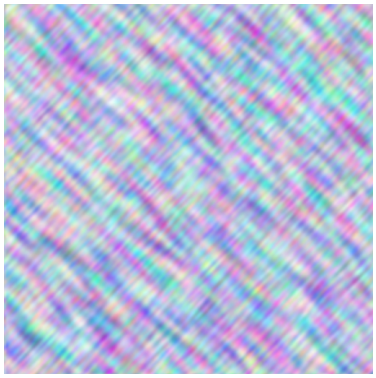


図 3.15: Grid パターン

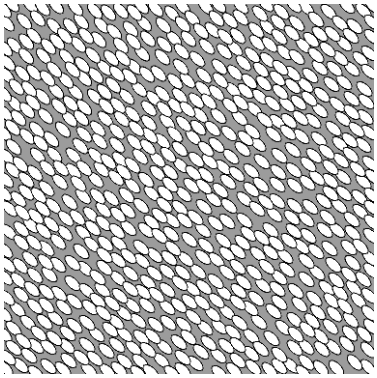


図 3.16: 配置結果

Radial パターン

Radial パターンは本手法におけるベクトル場生成手法の 1 つである回転場に相当する．Radial パターンでは，その主固有ベクトルが円の接線方向となる．Radial パターンのテンソル場を生成するために，ユーザが指定した点，すなわちデザインエレメント $\boldsymbol{p}_0$ をその中心点に設定する．このとき，Radial パターンのテンソル場は次のような形式になる [Zhang et al. 2007]：

$$T(\boldsymbol{p}) = \begin{pmatrix} y^2 - x^2 & -2xy \\ -2xy & -(y^2 - x^2) \end{pmatrix} \tag{3.21}$$

ただしここで，$x = x_{\boldsymbol{p}} - x_0, y = y_{\boldsymbol{p}} - y_0$ である．
次に Radial パターンによる生成例を示す．図 3.17 に示すように，両矢印を用いるのは Grid パターンと同様である．このようなユーザストロークからは図 3.18 のように $\boldsymbol{p}_0$ を中心と

する円に対して接線方向に主固有ベクトルを持つテンソル場が生成される．このテンソル場により生成される配置結果を図 3.19 に示す．

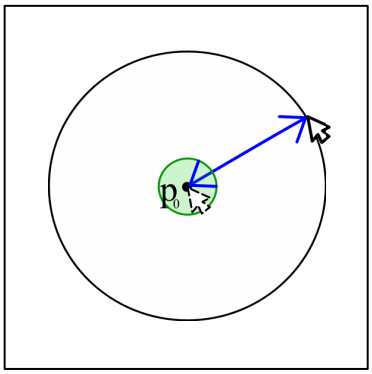


図 3.17: ユーザストローク

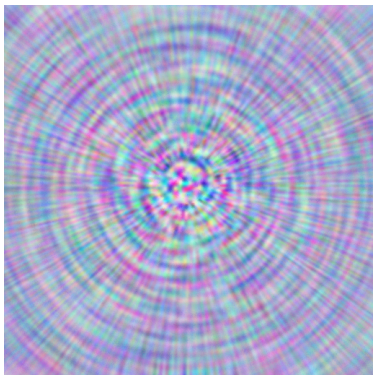


図 3.18: Radial パターン

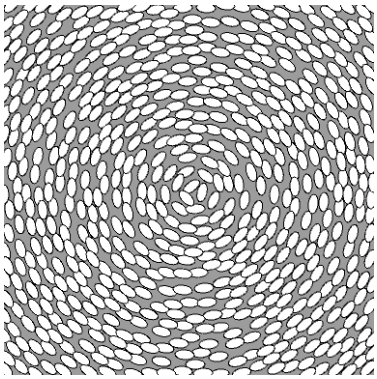


図 3.19: 配置結果

3.2.6 ストロークインターフェースによるテンソル場の編集

Chen らはさらにブラシインターフェースを導入している [Chen et al. 2008]．彼らのブラシインターフェースではユーザのストロークをいくつかのセルのストリップとして抽出し，各々のセルを構成する 4 頂点にテンソルの主固有ベクトルを設定している．本手法では，3.2.2 章で導入したストロークから流れ場を生成する手順をテンソル場に置き換えて適用する．すなわち，各ピクセルにおけるテンソル場の値は，

$$T(\boldsymbol{p}) = \sum_i e^{-d\|\boldsymbol{p}-\boldsymbol{p_i}\|^2} T_i(\boldsymbol{p}) \tag{3.22}$$

となる．テンソル場に関してもベクトル場のときと同様に各デザインエレメントの寄与を足し合わせることで最終的なテンソル場の値とする．このとき，$T(\boldsymbol{p})$ の固有値 λ と対応する固有ベクトル (主固有ベクトル) $\boldsymbol{A}$ からそのピクセルの流れ場の方向を $\boldsymbol{A}$ の向きとし，大きさは λ から決定する．

3.3 Poisson-Disk による点分布

次に，これまでで得られた流れ場に沿うかたちで点を分布させる手法について説明する．点分布には Poisson-Disk 分布において基本的な手法であるダートスローイング法を用い

る [Cook 1986] ．ダートスローイング法による Poisson-Disk サンプリングでは，サンプル点に Disk と呼ばれる排他領域を設け，サンプル点が他のサンプル点の Disk 領域に入らないように排他的に点を分布させる．試行回数には，[櫻井 and 宮田 2010] と似た計算式を用いる．櫻井らは試行回数を，

$$N_p = \frac{Ac_d}{\pi(0.5r)^2} \tag{3.23}$$

で与えている．ここで，A は画像の全ピクセル数，c_d は disk の充填率，r は disk の半径である．本手法では Disk の形状を楕円にして分布させるため，r の代わりに楕円の長軸の半径を a，短軸の半径を b として，

$$N_p = \frac{Ac_d}{\pi ab} \tag{3.24}$$

とする．また，本手法では Disk 領域どうしも排他的に点分布を行うので $r \to 0.5r$ のようなスケーリングも施さないこととした．

分布に際して，計算機のメモリ上に入力画像と同じ画素数分のメモリ領域を確保する．各画素を表す構造体に有効／無効のフラグ情報を保持するためのデータ構造を持たせる．試行の際，そのフラグ情報を参照して新しい点の位置を決定する．すなわち，1 つの点が配置される毎にその点の Disk 領域にあたる画素に有効フラグを格納する．N_p 回試行を繰り返し行うことで画像の全領域を覆うように点が配置される．次に配置される新しい点は有効フラグの格納されていない領域を配置の候補とする．Algorithm1 に疑似コードを示す (10 行目-18 行目)．ただし，疑似コード中の ♠ の $IsInside()$ 関数は，p を中心 (原点) としその Disk が，長軸の半径が a，短軸の半径が b の楕円の内部にある場合にのみ真となる関数である (図 3.20)．つまり p の Disk に属する集合は，

$$\left\{ (x,y) \in \mathbb{R}^2 \,\middle|\, \left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 \leq 1 \right\} \tag{3.25}$$

となる．ここでさらに，3.2 章で生成した流れ場 $\phi(p)$ を考慮した楕円形 Disk の分布を考える．点 $p(x,y)$ におけるベクトル値を $V(p) = (V_x, V_y)$ とすると，この点における流れ場が x 軸となす角度は $\alpha = \tan^{-1}(V_y/V_x)$ となる．ダートスローイングで打つ点 p での試行の際に用いる Disk は，図 3.20 の楕円を, 点 p を中心に角度 α だけ回転させた楕円形の Disk とする (図 3.21)．この場合，極座標系を用いると，式 (3.25) は次のように修正される：

$$\left\{ (r,\theta) \in \mathbb{R}^2 \,\middle|\, \left(\frac{r\cos(\theta-\alpha)}{a}\right)^2 + \left(\frac{r\sin(\theta-\alpha)}{b}\right)^2 \leq 1,\ 0 \leq \theta < 2\pi \right\} \tag{3.26}$$

ここで，(r,θ) は図 3.21 中の q の座標値であり，r は動径，θ は偏角である．式 (3.25) の (x,y) と (r,θ) の関係は，$q = re^{(\theta-\alpha)} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}\begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix}$，$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix}$ となる．
この集合に対して有効フラグを格納することで，流れ場を考慮した点の異方性分布を行う．

Algorithm 1 Edge-aware Poisson-Disk Distribution with ellipsoidal disk

```
I ← Image stored in memory
L ← List which stores successfully plotted pixels
for all p ∈ I do
    if −δ ≤ D(p) ≤ δ then
        p.flag ← true
    else
        p.flag ← false
    end if
end for
for i ← 0 to Np do
    p ← choose pixel randomly
    if p.flag = false then
        L.append(p)
        for all {p | IsInside(p)} do //  ····· ♠
            p.flag ← true
        end for
    end if
end for
```

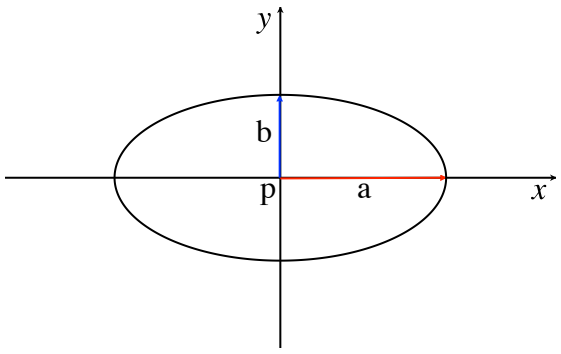


図 3.20: 楕円

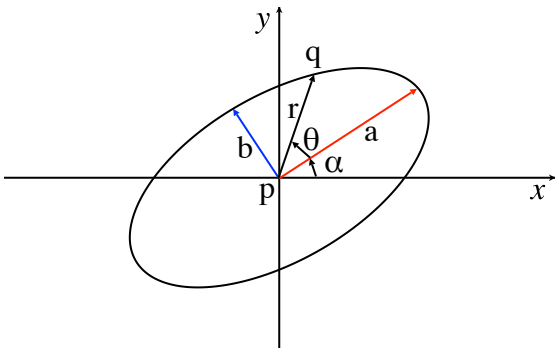


図 3.21: α だけ楕円を回転した場合

ところで，モザイク画生成手法では入力画像の特徴を生成結果に反映させるためにエッジ領域を避けるような行程をアルゴリズム中に組み込んでいる場合が多い．本手法でも入力画像の特徴を反映させる手段として，入力画像のエッジ領域を避けるようにダートスローイング法で点分布を行う．本手法では Algorithm1 に示した通り，各画素を表すデータ構造にはフラグ情報が保持されており，これに基づいて試行の成否を判定している．図 3.2 のエッジから図 3.3 で $\pm\delta$ の範囲のピクセルに対してフラグを有効にする (3-9 行目) ことで試行の際，エッジ領域を Disk 領域と同様に扱うことができ，点はエッジの周囲 $\pm\delta$ の領域には打たれない．以下に，Poisson-Disk 分布で分布させた結果を示す．

図 3.22: 図 3.1 に対し，テンソル場を用いて流れ場を指定

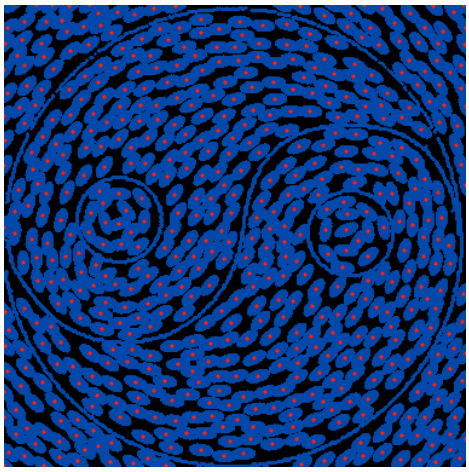


図 3.23: エッジと流れ場を考慮した Poisson-Disk 分布結果

図 3.22 は図 3.1 に対してストロークインターフェースを通して流れ場を指定したものである．このストロークから式 (3.22) に従いテンソル場が計算される．図 3.23 はエッジと，図 3.22 で指定した流れ場の向きを考慮して Poisson-Disk 分布を行った結果である．分布結果から，各分布点の Disk が流れ場の向きを反映している様子が確認できる．

3.4 点分布からの石ボリュームの生成

本章では，Poisson-Disk 分布で分布させた点から石形状を生成する手順について説明する．石形状の生成には大きく 2 つのステップがある．1 つ目は石の基本ボリュームを生成するステップ，2 つ目は基本ボリュームをスムージングするステップである．以下，手順の詳細について説明する．

3.4.1 石の基本ボリュームの生成

基本ボリュームの生成は次のような手順で行われる：

1. 楕円に外接する長方形を石の基本ボリュームの底面を構成する基礎形状とする
2. 長方形内部に新たに点を追加する
3. 長方形を構成する 4 頂点と，点 p の複製である点 p' を含む内部の点に対して三角分割を行い，メッシュを作成する
4. 内部に追加した (重心点の除く) 点を長方形の外側に移動する
5. 内部の点を高さ分だけスイープする．この際，必要に応じてスイープする高さにランダムさを加える

楕円に外接する長方形の 4 頂点

まず，楕円に外接する長方形の 4 頂点を求める．点 $p = (p_x, p_y)$ を中心 (重心) とする長軸の半径 a，短軸の半径 b の楕円を考える．このとき，楕円に外接する長方形の 4 頂点 $\{ q_i \,|\, 0 \leq i \leq 3 \}$ は，

$$\begin{cases} q_0 = (p_x + a, p_y + b) \\ q_1 = (p_x - a, p_y + b) \\ q_2 = (p_x - a, p_y - b) \\ q_3 = (p_x + a, p_y - b) \end{cases} \tag{3.27}$$

となる (図 3.24).

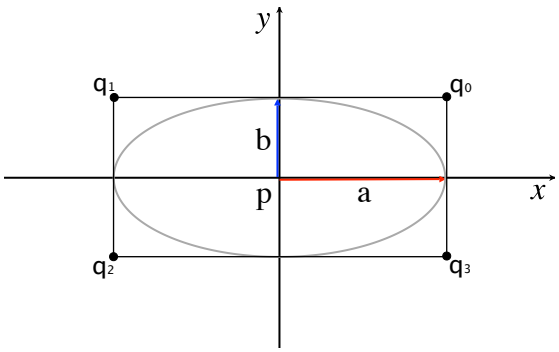


図 3.24: 楕円に外接する長方形

これらの頂点で石の基本ボリュームの底面を構成する (図 3.25).

長方形の内部に新たに点を追加

次に長方形の内部に新たな点を追加する．この際，上で求めた 4 頂点と重心のそれぞれに対して

$$\boldsymbol{q'} = \boldsymbol{q} + d \cdot \frac{\boldsymbol{q} - \boldsymbol{p}}{\|\boldsymbol{q} - \boldsymbol{p}\|} \tag{3.28}$$

を計算する．上式により点 $\boldsymbol{q}$ は点 $\overrightarrow{qp}$ の方向に d(ここでは $d < 0$) だけ移動する．これにより新たな点 $\boldsymbol{q'}$ の位置が求まる (図 3.26).

頂点群の三角形分割

次に，これらの点群に対して三角形分割を行う．三角形分割にはドロネー三角形分割を用いる．平面上の点集合 $\boldsymbol{P}$ の任意のドロネー三角形分割は，$\boldsymbol{P}$ のあらゆる三角形分割の中で最小角度を最大にする [de Berg et al. 2008]，すなわち極端に細長い三角形が生成されにくいため，CG の分野では広く利用されている．三角形分割した結果を図 3.27 に示す.

追加した点を長方形外部へ移動

長方形内部に追加した，重心点を除く点群を長方形外部へ移動する．これには式 (3.28) で $d > 0$ とすればよい．このようなステップを踏むことで，長方形の内部は三角形分割されずに，外側のみに三角メッシュを構成することができる (図 3.28)．これにより後にメッシュのサブディビジョン (再分割) を行う際，半球状の形状を得ることが可能となる.

移動した頂点を高さ分だけ上方に持ち上げる

外部へ移動した 4 頂点を，$d_{height} + Rnd$ だけ上方へ持ち上げる．Rnd は一様乱数とする．また，重心に生成した点の持ち上げ量は $d \cdot (d_{height} + Rnd)$, $d \approx 1.2$ とした．以上により小石の基本ボリュームを生成する．生成結果を図 3.29 に示す．

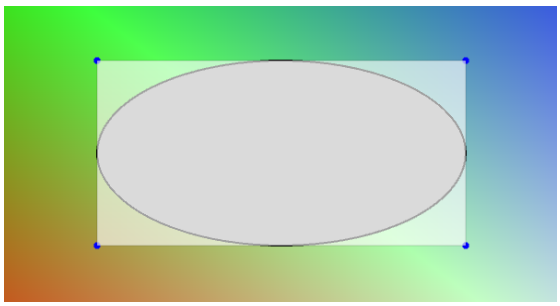


図 3.25: 楕円に外接する長方形

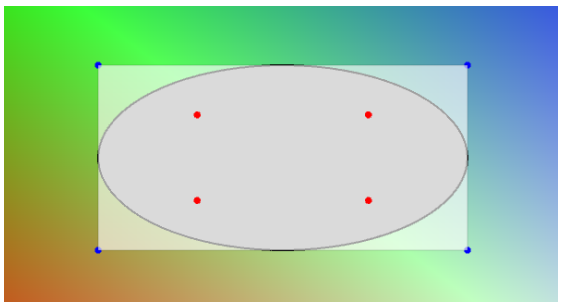


図 3.26: 内部に 4 頂点を追加

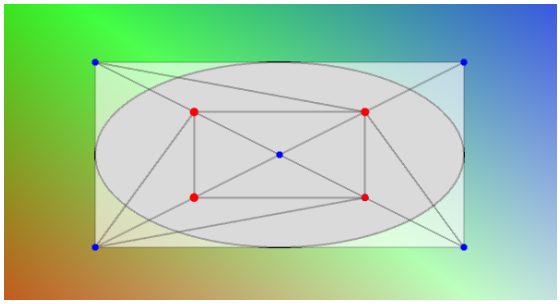


図 3.27: 頂点群を三角形分割

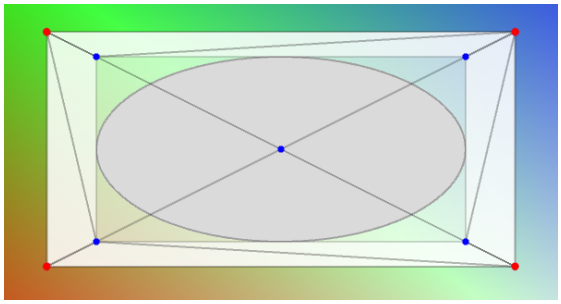


図 3.28: 内部の頂点を外側に移動

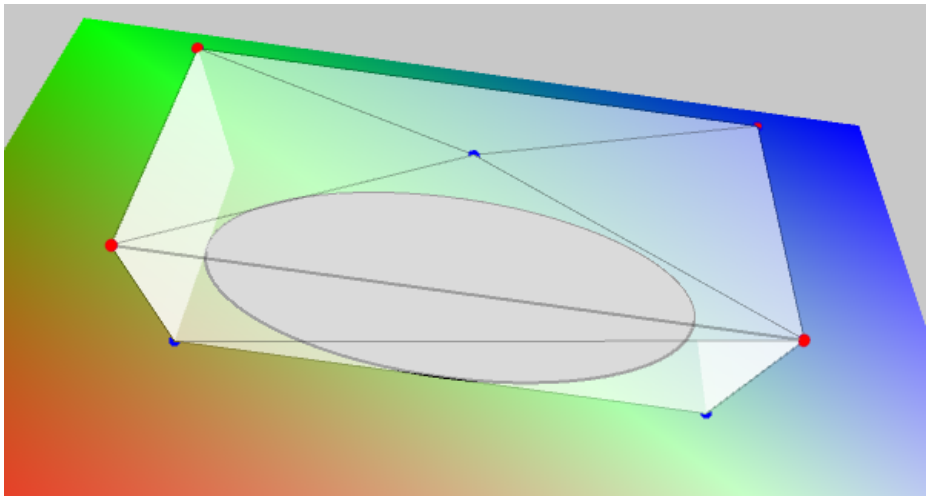


図 3.29: 移動した頂点を高さ分だけ上方に持ち上げる

3.4.2 基本ボリュームのスムージング

生成した基本ボリュームをスムージングし，形状により丸みを持たせる．本手法ではスムージングに Loop サブディビジョン (再分割) を用いる．Loop サブディビジョンでは，1 つの三角形メッシュを 4 つの新しい三角形に再分割する．よって n 回再分割を行うと 1 つの三角形は $4n$ 個の三角形に分割されることになる [Akenine-Möller et al. 2008]．

Loop サブディビジョンでは，次の再分割規則を用いる：

$$\boldsymbol{p}^{\boldsymbol{k+1}} = (1-k\beta)\boldsymbol{p}^k + \beta(\boldsymbol{p}_0^k + \cdots + \boldsymbol{p}_{n-1}^k) \tag{3.29}$$

$$\boldsymbol{p}_i^{k+1} = \frac{3\boldsymbol{p}^k + 3\boldsymbol{p}_i^k + \boldsymbol{p}_{i-1}^k + \boldsymbol{p}_{i+1}^k}{8},\ i = 0...n-1 \tag{3.30}$$

ここで，

$$\beta(n) = \frac{1}{n}\left(\frac{5}{8} - \frac{(3+2\cos(2\pi/n))^2}{64}\right) \tag{3.31}$$

式 (3.29) は既存の頂点 $\boldsymbol{p}^k$ を $\boldsymbol{p}^{k+1}$ に更新するための規則，式 (3.30) は $\boldsymbol{p}^k$ と $\boldsymbol{p}_i^k$ の間に新しい頂点 $\boldsymbol{p}_i^{k+1}$ を追加するための規則である．ただし，n は $\boldsymbol{p}^k$ の価数である．図で表すと次のようになる．これらをマスクあるいはステンシルと呼ぶ．式 (3.29) は図 3.30，式 (3.30) は図 3.31 にあたる．また，境界の場合はそれぞれ図 3.32，図 3.33 のステンシルを用いる．

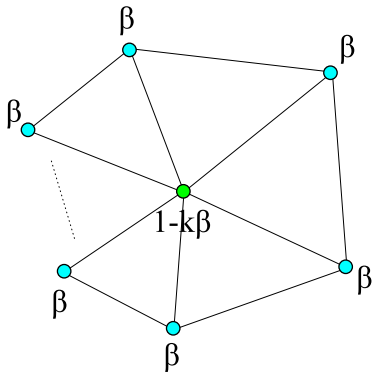


図 3.30: 頂点ステンシル

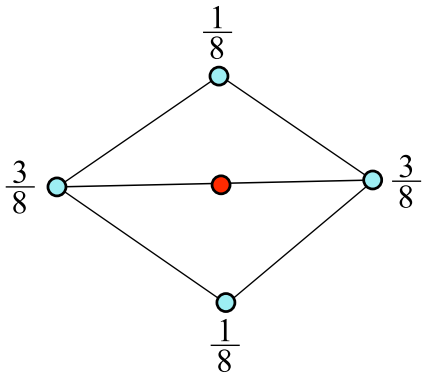


図 3.31: エッジステンシル

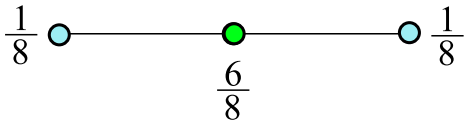


図 3.32: 境界ステンシル (更新)

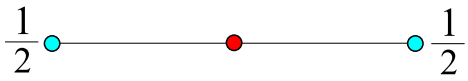


図 3.33: 境界ステンシル (追加)

以下に，図 3.25〜図 3.29 で作成した基本ボリューム (図 3.34) に Loop サブディビジョンを適用した結果を示す (図 3.35).

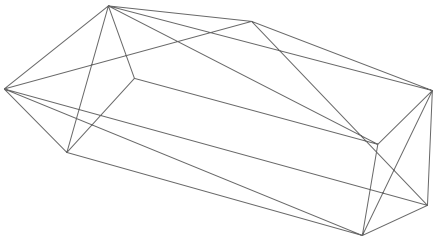


図 3.34: 図 3.29 で作成した基本ボリューム

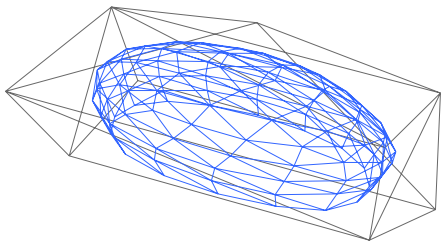


図 3.35: 図 3.34 に再分割を 2 回適用した結果

また，次に図 3.36 から生成した基本ボリューム (図 3.37) と，それに対して Loop サブディビジョンを適用した結果を示す (図 3.38).

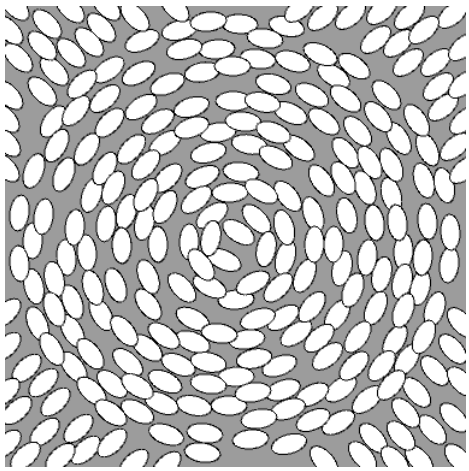


図 3.36: 配置バターン

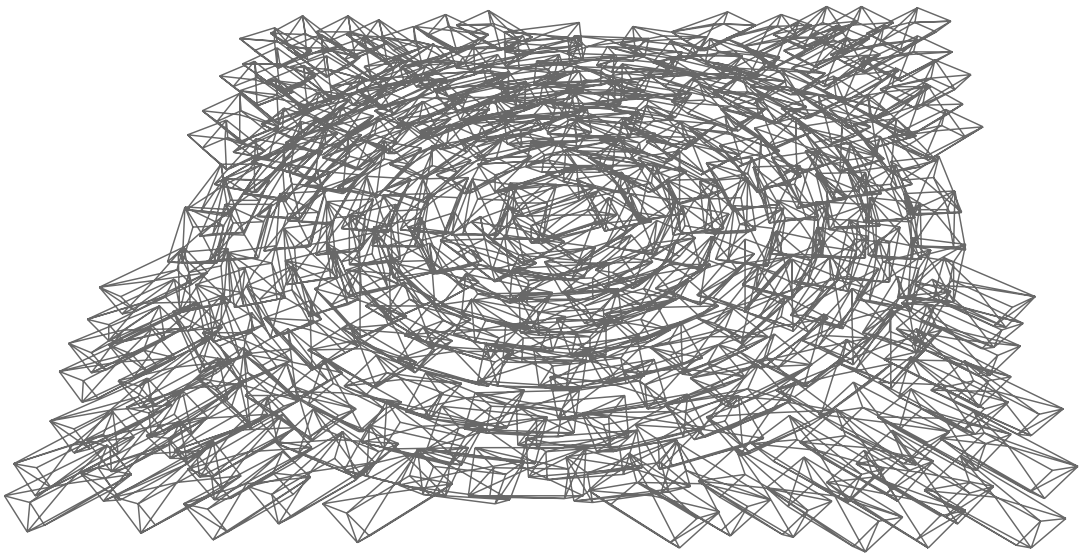


図 3.37: 図 3.36 から生成した基本ボリューム

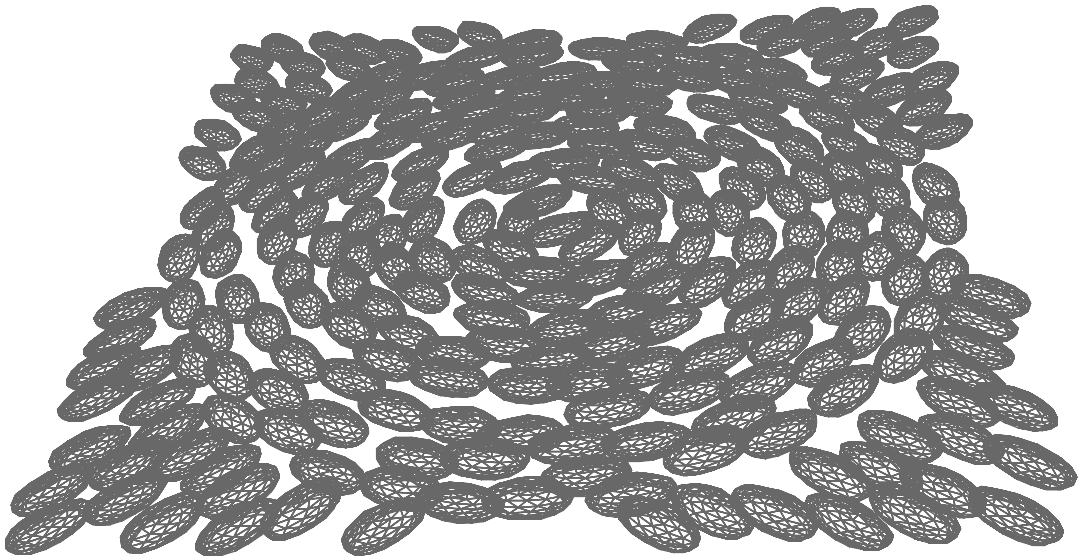


図 3.38: 図 3.37 に再分割を 2 回適用した結果

第 4 章
小石モザイクの装飾

3章で石の配置パターンの生成手法，そして石形状の生成手法について説明した．ここでは実際の小石モザイク風の装飾を施すための手法について説明する．

実際の小石モザイクでは，前景と背景の差を明確に区別可能とするために前景と背景で形状の異なる石を用いる場合がある．その一例として，前景に丸石を用い，背景に表面形状がフラットな石を用いる場合がある (図 2.2, 図 2.3(b))．3章で生成した小石モデルで前景は表現できるため，ここでは背景の表面形状がフラットな石の生成手法，そして前述の小石モデルと組み合わせる手法について説明する．

また，実際の小石モザイクでは 3D 効果を施したものが見られる (図 4.1)．

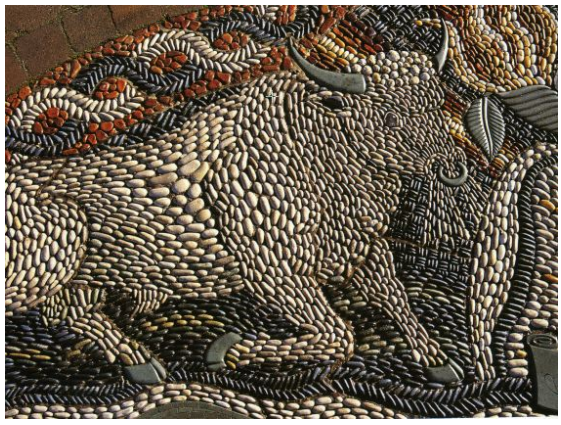


図 4.1: 3D 効果を施した小石モザイク

図 4.1 では，エッジから前景中心に向かうにしたがって，小石の大きさが徐々に大きくなっ

ている．これは横から見たときに前景の動物の体躯の丸みを 3 次元的に表現するため，エッジ付近では小さい石を用い，中央付近で大きい石を用いることで演出している．本手法では 3D モデルを入力とすることでこの効果のように高さを付与することが可能である．このことについても説明する．

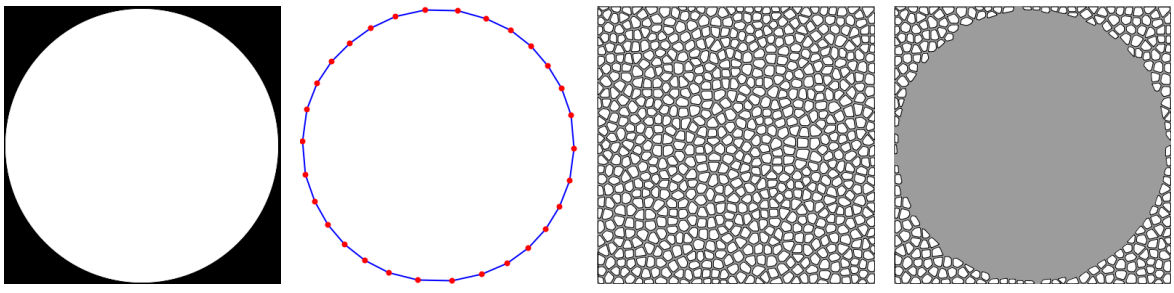
4.1 小石モザイクの背景の生成手法

4.1.1 手法の概要

小石モザイクの前景，背景を表現するため，本手法はマスク画像を追加の入力とする．マスク画像は前景部分を白色，背景部分を黒色で表現した 2 値画像である．また本手法は，背景にあたる表面形状のフラットな石を生成するため，3.3 節の点分布からボロノイ図を構成する．構成したボロノイセルは 3.4 節に従い，小石の場合と同様に石ボリュームを構成することが可能である．つまり，背景モデルの生成は 3 節で石の基本ボリュームの構成方法が異なるだけで，あとは全く同等の手法により構成することが可能である．手法の概要は以下のようになっている：

1. 3.3 節の手法で分布させた点群からボロノイ図を構成する
2. ボロノイセルからフラットな石形状を生成する
3. マスク画像を用いて，その前景部分に小石モデル，背景部分にフラットな石を配置する

以下では，3 章との差分として，ボロノイ図の構成，フラット形状の生成，マスク画像による石モデルの配置方法について説明する．



(a) (b) (c) (d)

図 4.2: 小石モザイクの背景の生成手法
(a)：マスク画像．(b)：輪郭線追跡．(c)：Poisson-Disk 分布から構成したボロノイ図．(d)：マスク画像によりクリッピングされたボロノイ図．

4.1.2 マスク画像から境界線を抽出する

まず，マスク画像から前景と背景の境界線を抽出する．これは後にボロノイセルをクリッピングするために必要な処理である．

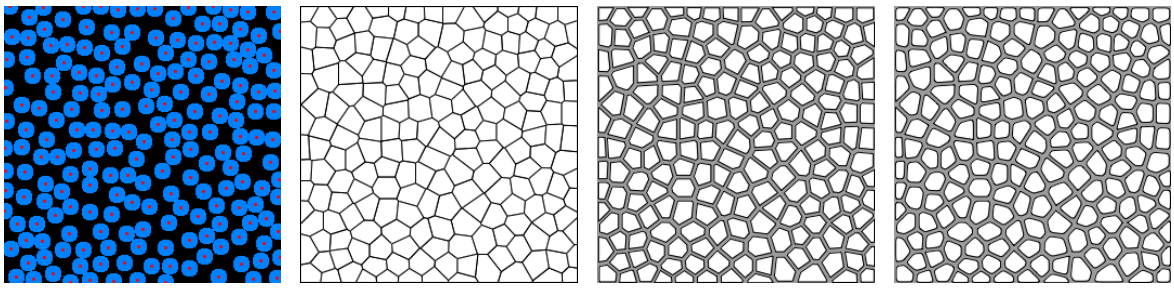
境界線の抽出は輪郭線追跡で行うことが可能である．2 値画像における輪郭線追跡では，前景 (ピクセル値 1) と背景 (ピクセル値 0) の境界線を追跡することになる．アルゴリズムの概要は次のようになっている：

1. ピクセルを画像左上から順にラスタースキャンする
2. 初めてピクセル値が 1 になったピクセルを追跡の開始地点とする
3. 開始地点からスタートし，近傍のピクセル値を調べる．ピクセル値が 1 ならそのピクセルを境界として登録し，そこを新たな追跡点とし，近傍を調べる．ピクセル値が 0 なら反時計周りに更に近傍を調べる．
4. 以降，次ピクセルが開始地点のピクセルになるまでこれを繰り返す．

近傍の調べ方は前回の追跡点との位置関係を考慮することで効率的に調べることができる．アルゴリズムの詳細は文献 [昌達 2008] 等に譲り，ここではこれ以上触れない．以上により，境界線を構成するピクセル群を得ることができる．なお，複数の境界領域を走査する際には上記の手順を領域の分だけ繰り返せばよい．次に，これらのピクセル群を一定間隔でサンプリングする (図 4.2(b) 中の赤丸)．サンプリングした点群を $\{ c_i \mid 0 \leq i \leq n \}$ とすると，線分 $\overline{c_i c_{i+1}}$ を得る (図 4.2(b) 中の青線)．後ほど，ボロノイセルを線分 $\overline{c_i c_{i+1}}$ でクリッピングし，セルを削っていくことになる．

4.1.3 Poisson-Disk 分布からボロノイ図を構成する

Poisson-Disk 分布で分布させた点群からボロノイ図を構成する．ここでは 3.3 節の手法において，楕円ではなく，円 ($a = b$) を Disk 形状として Poisson-Disk 分布を行う．これにより得られた点群をボロノイセルの母点 (site) として，ボロノイ図を生成する．



(a) (b) (c) (d)

図 4.3: Poisson-Disk 分布から生成したボロノイ図
(a)：Poisson-Disk 分布．図中の赤点が母点 (site) を表し，青色領域は Disk 領域を表す．(b)：(a) から生成したボロノイ図．(c)：(b) の各セルをその重心方向に縮小した図．(d)：(c) の各セルをなめらかにした図．

まず，Poisson-Disk 分布で分布させた点からボロノイ図を構成する (図 4.3(a))．このとき，これらの点 (p_i) は母点 (site) と呼ばれる．ボロノイ図はグラフで表され，p_i の面は，他のどの p_j よりも p_i に近い 2D のすべての点を表す．ボロノイ図の形式的な定義を次に示す [Langetepe 2006]：

定義 4.1.1 (ボロノイ図) $p_i, p_j \in S$ に対し，

$$Bis(p_i, p_j) = \{\, x \mid d(p_i, x) = d(p_j, x) \,\} \tag{4.1}$$

で p_i と p_j の二等分線 *(bisector)* を示す．$Bis(p_i, p_j)$ は線分 $\overline{p_i p_j}$ の中点を通る垂直線を表す．二等分線は平面を *2* つの開いた半平面

$$H(p_i, p_j) = \{\, x \mid d(p_i, x) < d(p_j, x) \,\} \tag{4.2}$$
$$H(p_j, p_i) = \{\, x \mid d(p_j, x) < d(p_i, x) \,\} \tag{4.3}$$

に分離し，$H(p_i, p_j)$ は p_i を含み，$H(p_j, p_i)$ は p_j を含む．

S に関する p_i のボロノイ領域 *(Voronoi region)* は，次のように $n-1$ 個の半平面の交差で定義される．

$$VoR(p_i, S) = \bigcap_{p_i \in S, p_j \neq p_i} H(p_i, p_j) \tag{4.4}$$

S 自身のボロノイ図 $VD(S)$ は，次で定義される．

$$VD(S) := \bigcup_{p_i,p_j \in S, p_i \neq p_j} \left(\overline{VoR(p_i, S)} \bigcap \overline{VoR(p_j, S)} \right) \tag{4.5}$$

Poisson-Disk 分布で生成した母点から構成したボロノイ図を図 4.3(b) に示す．図 4.2(c) や，図 4.2(d) に示した図は，石形状を表現するために各ボロノイセルを式 (3.28) を用いてその重心方向に縮小し (図 4.3(c))，さらに 2 次のベジェ曲線を用いてなめらかなセルとして表現したものである (図 4.3(d))．

4.1.4 ボロノイセルのクリッピング

次に，ボロノイ図を構成する各々のボロノイセルに対して，線分 $\overline{c_i c_{i+1}}$ と 2 点で交わっているかどうかを判定する．もし 2 点で交わっていれば，線分 $\overline{c_i c_{i+1}}$ に対してボロノイセルの重心と反対側の頂点を削除し，かわりにボロノイセルと線分 $\overline{c_i c_{i+1}}$ の 2 つの交点をそのボロノイセルを構成する頂点群に加える．つまり，クリッピング後のボロノイセルは線分 $\overline{c_i c_{i+1}}$ に対して，元のボロノイセルの重心側の頂点群と 2 つの交点から構成される．

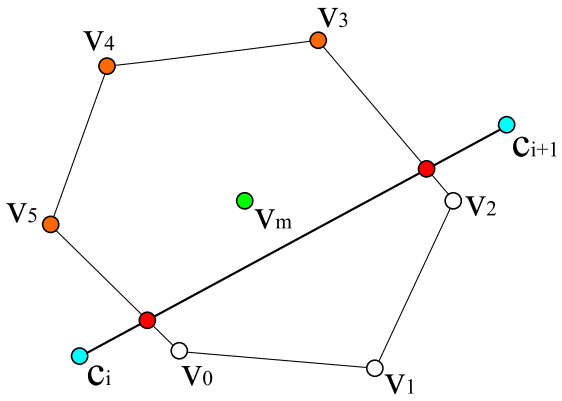


図 4.4: ボロノイセルと線分が 2 点で交差する場合

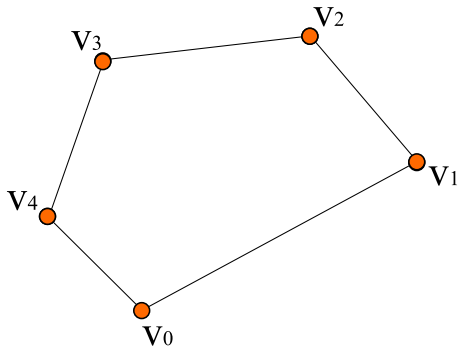


図 4.5: クリッピングしたボロノイセル

図 4.4 において，線分 $\overline{c_i c_{i+1}}$ とボロノイセルが 2 点で交わり，その交点を赤点で示す．この場合，ボロノイセルを構成する頂点 $\{v_0, v_1, ..., v_5\}$ のうち，緑点で示したセルの重心 v_m と線分 $\overline{c_i c_{i+1}}$ に関して反対側にある白点で示した点 $\{v_0, v_1, v_2\}$ は削除され，かわりに赤点で示した 2 つの交点と橙色で示した頂点 $\{v_3, v_4, v_5\}$ で新たにボロノイセルを構成する (図 4.5)．ただし，図 4.5 では新しいボロノイセルの頂点に対して添字を新たに振り直してある．

クリッピング例を次に示す (図 4.6)：

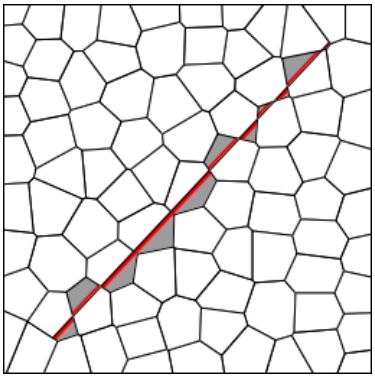
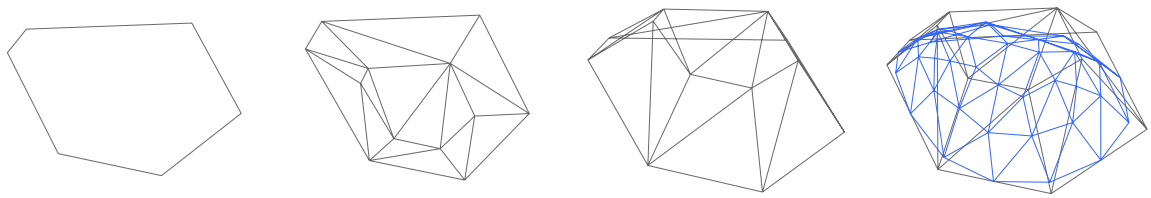


図 4.6: クリッピング例

図から，赤線で示した線分と 2 点で交わるボロノイセルの線分に対して，その重心と反対側の領域が削除され，背景色 (グレー) が見えているのが確認できる．図 4.2(b) の境界線を構成する線分のそれぞれに対して，図 4.2(c) のそれぞれのセルとの交差判定を行うことでセルのクリッピングを行う．そして，マスク画像の前景領域のセルも削除することで図 4.2(d) に示した背景を得る．

4.1.5　フラット形状の生成



(a)　(b)　(c)　(d)

図 4.7: ボロノイセルから石形状の生成
(a)：ボロノイセル．(b)：内部に点群を追加し，三角形分割した結果．(c)：内部の点群を持ち上げて石の基本ボリュームを生成．(d)：(c) の基本ボリューム (グレー) を再分割した結果 (青)．

ボロノイセルから石ボリュームを生成する場合も 3.4 節の手順を踏めばよい．すなわち，

各々のボロノイセルに対し (図 4.7(a))，その内部に点群を追加して三角形分割を施す (図 4.7(b))．そして内部の点を外側に移動させつつ，高さを付加する (図 4.7(d))．最後に再分割を行い，多少の丸みを持たせる (図 4.7(d))．以下に，ボロノイセルから生成した石畳モデル (図 4.8) と，それに再分割を施したもの (図 4.9) を示す．また，そのそれぞれに対してフラットシェーディングで真上から見たモデルも図 4.10 と図 4.11 に示す．

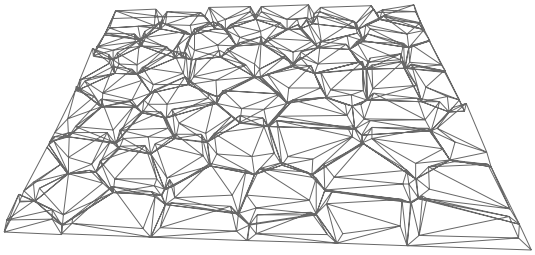


図 4.8: ボロノイ図から生成した石畳モデル

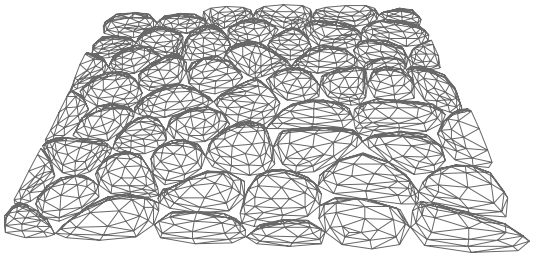


図 4.9: 再分割を施した石畳モデル

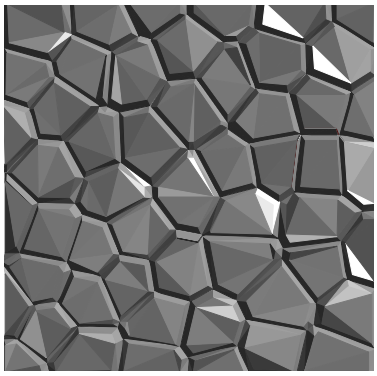


図 4.10: 真上から見た石畳モデル (フラットシェーディング)

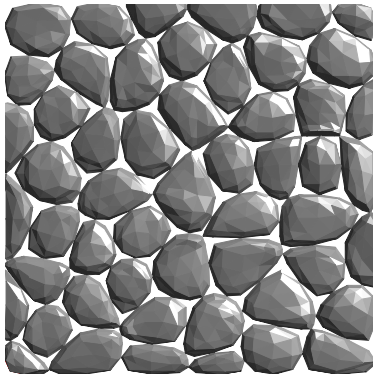


図 4.11: 真上から見た再分割を施した石畳モデル (フラットシェーディング)

4.2 小石モザイクの 3D 効果の演出方法

本手法では 3D モデルを入力とすることで図 4.1 のように高さを付与することが可能である．Noda らは 3D モデルを入力としたモザイク画の生成手法について提案している [Noda and Miyata 2009]．本手法では Noda 等の手法をもとに 3D モデルから特徴線を抽出する．

すなわち，3D モデルから法線マップと深度マップを生成し，各々に対してラプラシアンフィルタを用いて 3D 形状からエッジとリッジを抽出する．本手法ではエッジからの距離と 3D モデルから取得した深度マップの値の両方を考慮し適応的に小石の大きさを変化させる．まず，Poisson-Disk 分布に用いる Disk の形状に対し，

$$s = 1 + d \log \left(\frac{depth_{max}}{D_{depth}(\boldsymbol{p})} \right) \tag{4.6}$$

を乗じることで Disk 形状にスケーリングを施す．ここで $depth_{max} = 255$，$D_{depth}(\boldsymbol{p})$ は $\boldsymbol{p} = (x, y)$ における深度値である．また，石の基本ボリュームの生成に際して，

$$d \cdot s \sqrt{D_{distance}(\boldsymbol{p})} \tag{4.7}$$

を乗じることでエッジからの距離に応じたスケーリングを行う．ここで d は定数である．

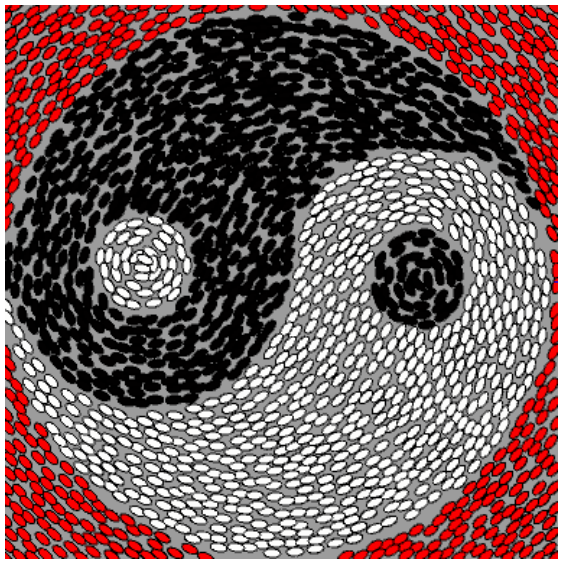
第 5 章
結 果 と 考 察

5.1 結果

以下に生成結果を示す．図 5.1，図 5.3 中の参照画像は実際の小石モザイクであり，図 5.5 中の参照 3D モデルは入力とした 3D モデルである．流れ場のデザインはユーザがマウスドラッグによって描いたストロークである．図 5.1(c)，図 5.3(c) では流れ場の寄与の影響が色領域内のみになるように領域制限を設けている．また，図 5.5(c) では，前景背景によって領域制限を設けている．

(a) 参照画像

(b) 流れ場のデザイン



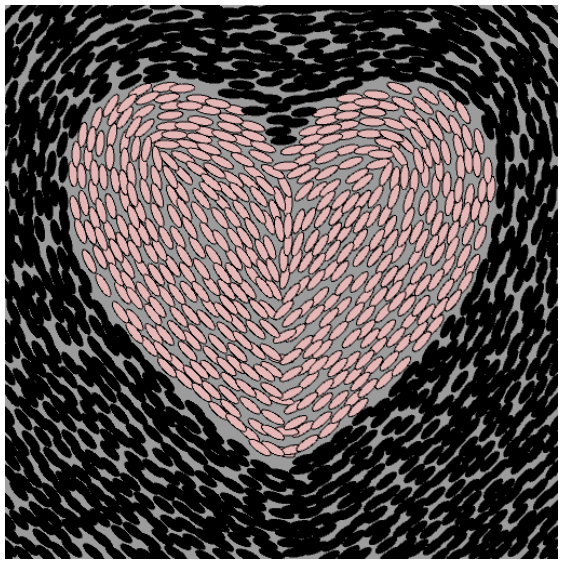
(c) 配置パターン生成結果

図 5.1: 生成結果 1：陰陽図の配置パターン

図 5.2: 生成結果 1：陰陽図のレンダリング結果

(a) 参照画像

(b) 流れ場のデザイン



(c) 配置パターン生成結果

図 5.3: 生成結果 2：ハートマークの配置パターン

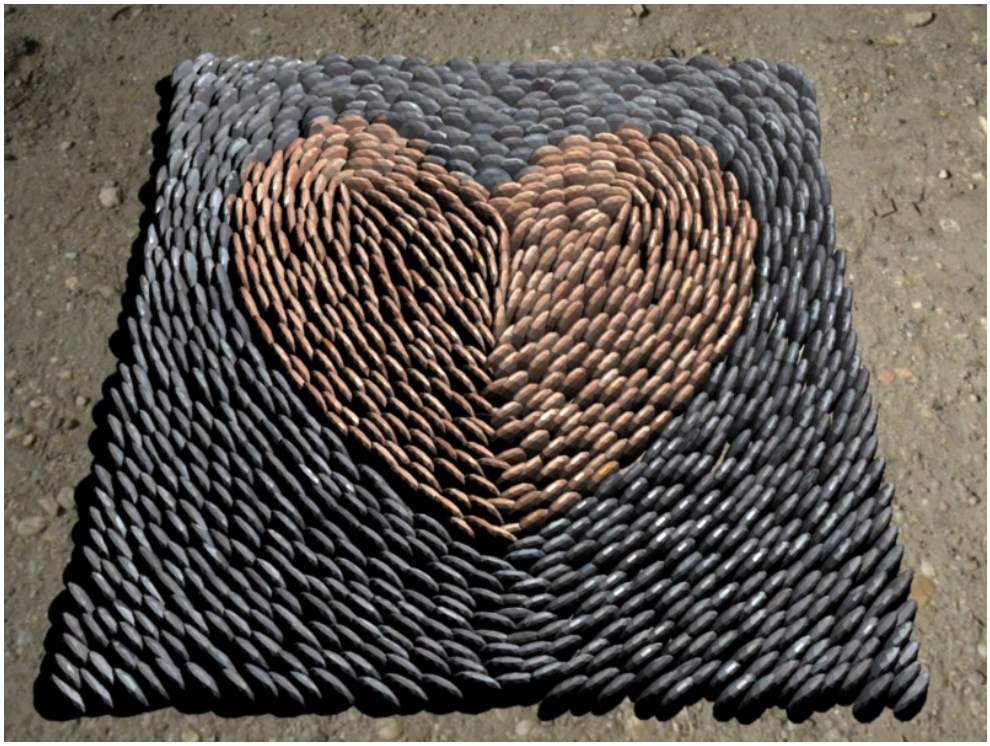
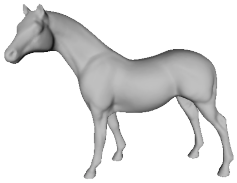
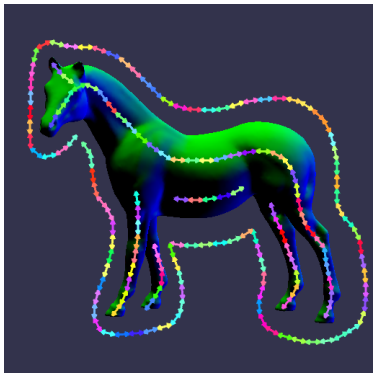


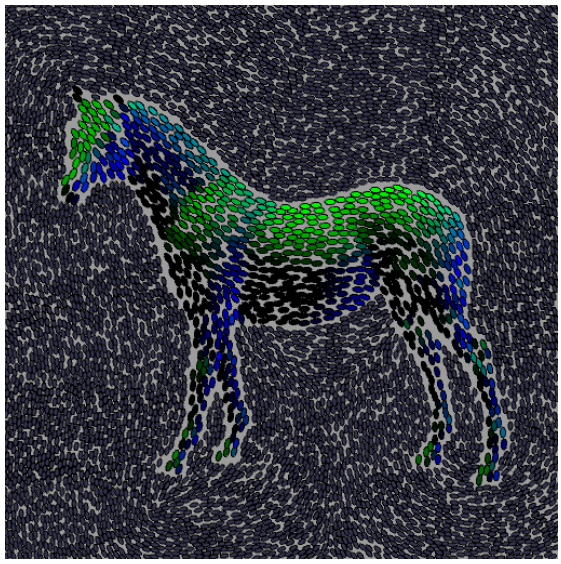
図 5.4: 生成結果 2：ハートマークのレンダリング結果



(a) 入力 3D モデル



(b) 流れ場のデザイン



(c) 配置パターン生成結果

図 5.5: 生成結果 3：馬の 3D モデルを入力とした際の配置パターン

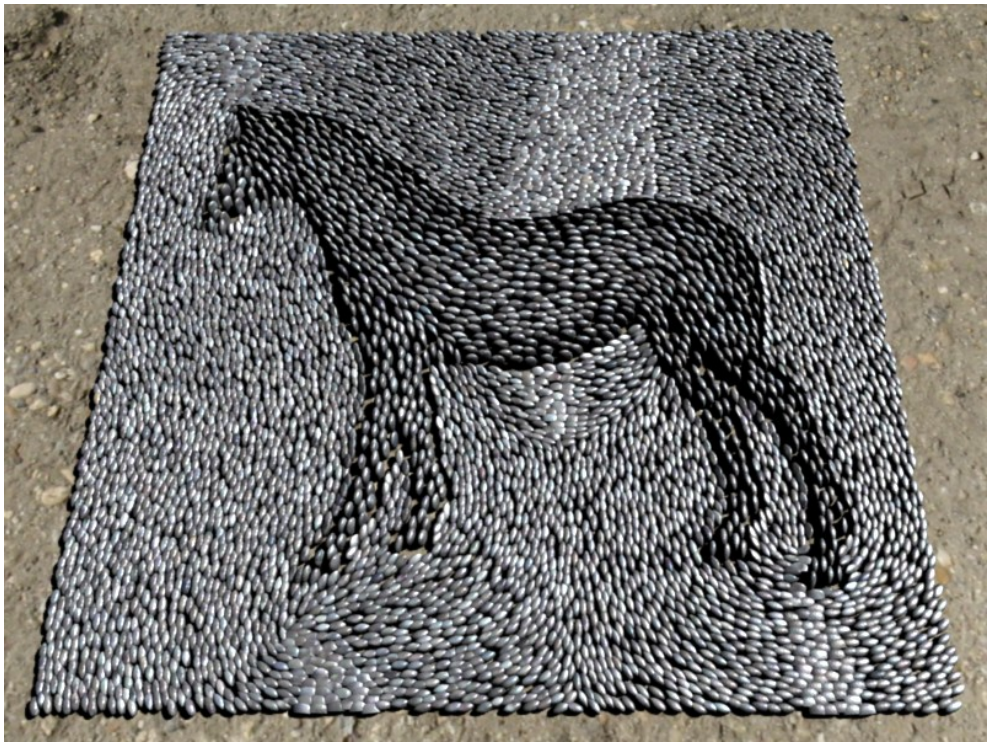


図 5.6: 生成結果 3：馬の 3D モデルを入力とした際のレンダリング結果

5.2 考察

生成結果 1(図 5.1, 図 5.2) は，陰陽図を入力として与えた際の配置パターンと生成された 3D モデルのレンダリング結果である．図 5.1(b) のように流れ場のデザインを指定した．これは，2 つの円に Radial パターンのテンソル場を指定し，他の領域はエッジに沿うかたちでストロークを置いたものである．その結果生成された流れ場と入力画像から抽出したエッジ情報をもとに Poisson-Disk 分布を行う．この際，楕円 Disk の長軸と短軸の比は 2 : 1 とした．図 5.1(c) がその配置パターンの生成結果であり，流れ場のデザインが反映されているのが確認できる．さらに，この配置パターンから 3D 形状を生成したものが図 5.2 になる．2D の配置パターンは石ボリュームの底面となるが，図 5.1(c) から石と石の間の隙間が目立つ印象を受ける．しかし，これは 3D 形状を生成する際に，3.4 節で述べたように，上方に持ち上げる頂点を底面よりも外側に移動しつつ持ち上げることでスムージング後も石と石の間の隙間を目立たせることなく，敷き詰まった生成結果を得ることができた．

生成結果 2(図 5.3, 図 5.4) は図 5.3(a) のような配置を生成するため，図 5.3(b) のようにハートマークに対して流れ場のデザインを行ったものである．この際，生成結果 1 よりも細長い形状の石となるよう楕円の長軸・短軸の比を 4 : 1 とした．これも，図 5.3 から石と石の間の隙間が目立つ印象を受けるが，3D 形状を生成した結果，図 5.4 からは敷き詰まっている様子が確認できる．また，本手法では点分布の際にダートスローイング法による Poisson-Disk 分布を用いており，一度打たれた点に関してその後点を移動させるような操作を行っていない．しかし，図 5.4 の生成結果からはハートのエッジに沿って一定の間隔 (オフセットラインあるいはレベルラインとも言われる) に沿って配置したかのような印象を受ける．ダートスローイング法による Poisson-Disk 分布を用いている限り，必ずしもこのような配置になる保証はない．これに対しての対策としては，点分布に対してリラクゼーション [Lloyd 1982] などの最適化を行うといった方法が考えられる．

生成結果 3(図 5.5, 図 5.6) は，入力として 3D モデル (図 5.5(a)) を与えた際の生成結果である．3D モデルから深度マップと法線マップを生成し，それらに対してエッジ検出を行い，両者を合成したものをエッジ画像とする．点分布の際，Disk 形状を深度マップの値で重み付けすることで深度値に応じて Disk 形状をスケーリングすることができる．また，石ボリューム生成時にも深度値やエッジからの距離に応じたスケーリングを行うことで図 4.1 に見られるような 3D 効果を付加した．図 5.7 に馬の 3D モデルから得た深度マップを示す．また，図 5.8 に図 5.6 を低いアングルから見た拡大図も示す．深度マップから，馬の顔や左前足がよりカメラアングルに近いことが分かる．図 5.8 から，実際に小石モザイクを生成した結果でも，馬モデルは背景よりも全体的に石の大きさが大きい結果となっているのが確認できる．また馬の顔や左前足は前景のなかでも高い石形状が生成されている様

子が確認できる．また，図 5.9 にも別の 3D モデルを入力とした際の生成結果を示す．この結果から，前景と背景の境界エッジから前景中央部に向かうに従い，石の大きさが大きくなっている様子が確認できる．しかし現状では，図 4.1 のようなはっきりとしたスケーリングを実現できているわけではなく，重み付けの計算は再考する必要があると言える．

背景装飾の生成結果 (図 5.10) は 4 章で述べた背景装飾を施した結果をレンダリングしたものである．生成結果から，前景と背景で区別が明確になっているのが確認できる．しかし，前景と背景の境界領域の隙間も場所によって目立つ印象を受ける．これは現在，前景・背景 2 値のマスク画像に対して，輪郭線追跡した結果をセグメントに分割したものとボロノイセルが 2 点で交差する場合のみセルのクリッピングを行っているため，セグメントの大きさや，交差判定が原因となっていることが考えられる．セグメント化された線分は細かいほど実際の境界を近似できる一方で，線分が短すぎるとボロノイセルと 2 点で交差しない場合が多くなり，クリッピングが上手くいかない可能性も考えられる．そのため短い線分に対しても上手くクリッピングできるように 2 点の交差判定ではなく，クリッピング断面を線分そのものにするなどの方法が考えられる．また，図 5.11 と図 5.12 に背景装飾のみのレンダリング結果を示す．

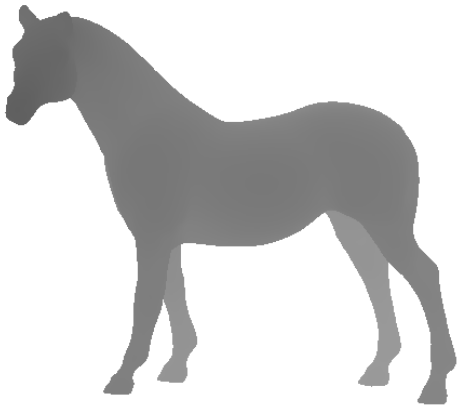


図 5.7: 馬の 3D モデルから得た深度マップ

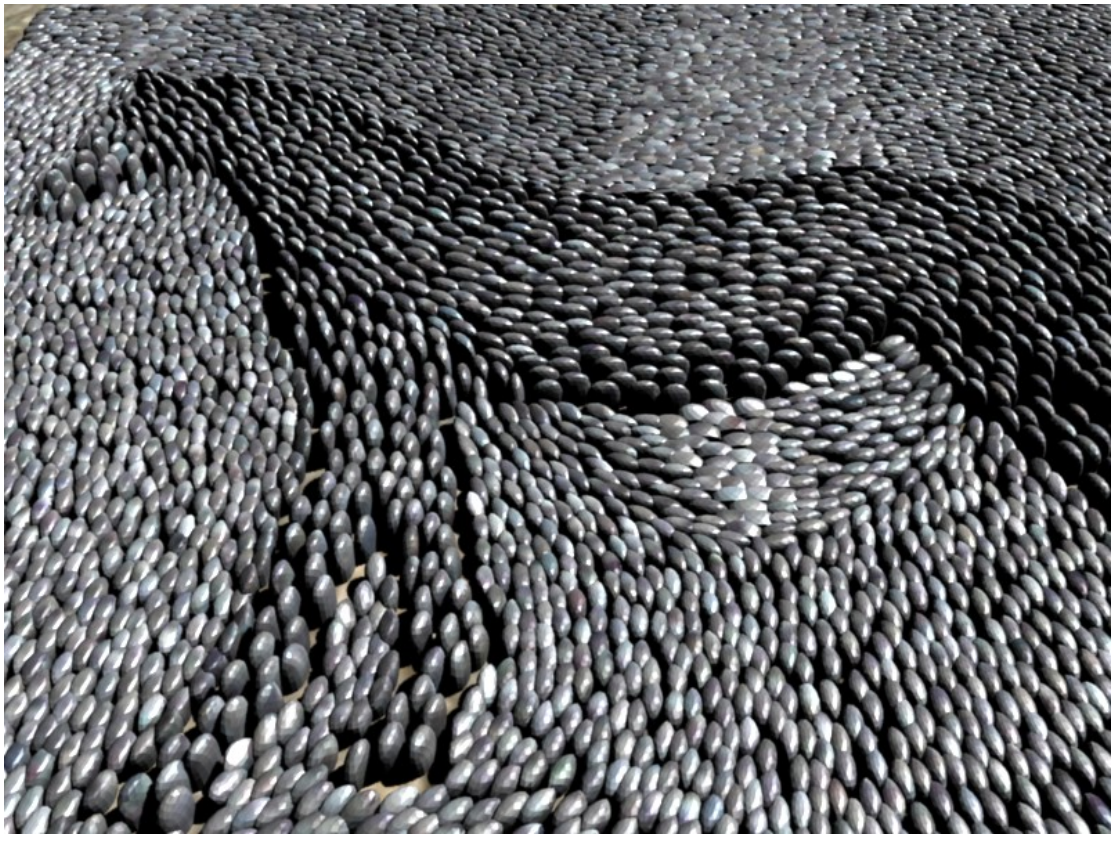


図 5.8: 図 5.6 を低いアングルから見た拡大図

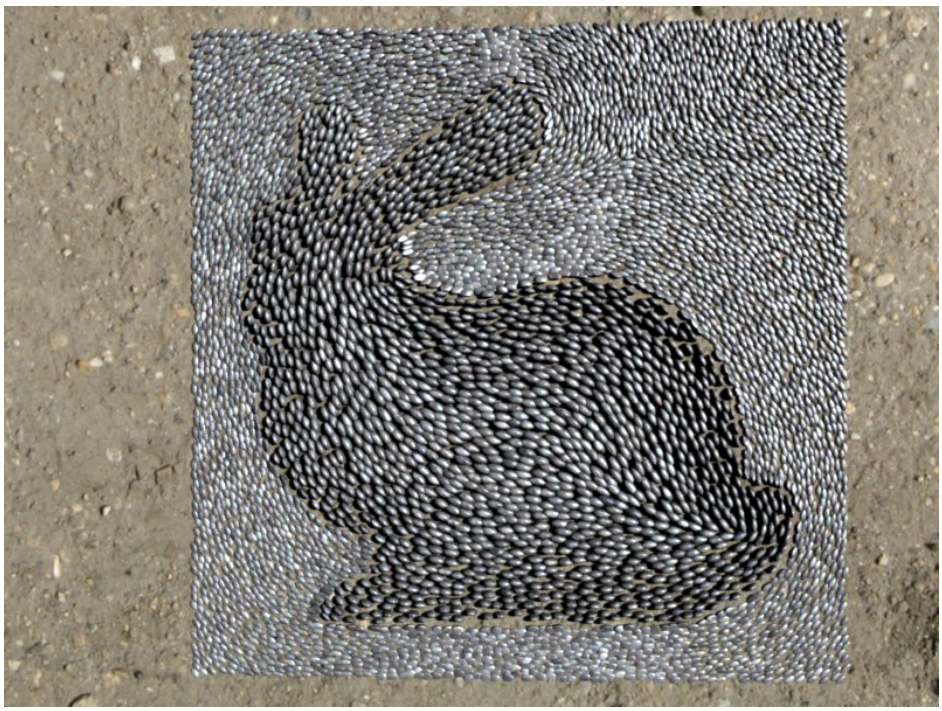


図 5.9: 3D 効果を施した小石モザイクのレンダリング結果 (Stanford Bunny)

図 5.10: 背景装飾を施した小石モザイクのレンダリング結果

図 5.11: 背景装飾のみのレンダリング結果 (Stanford Bunny)

図 5.12: 背景装飾のみのレンダリング結果 (ハートマーク)

第 6 章
まとめと今後の課題

6.1 まとめ

本研究では，3 次元形状としての小石モザイクの生成手法を提案した．ユーザは入力画像に対してストロークを描くことでモザイク片である小石の配置の流れを指定する．システムはストロークからテンソル場を計算し，流れ場を生成する．入力画像の特徴を反映させるため，エッジ情報を用いて流れ場から算出された流れの向きに沿うような楕円形 Disk を用いた Poisson-Disk 分布で母点を配置する．Disk 形状をもとに 2D パターンから 3D 形状の石ボリュームを生成する．生成されたボリュームはスムージングを施すことで丸みを持った丸石形状を生成する．

生成結果から，入力画像を反映した小石モザイクが生成できていることが確認できる．小石の配置パターンもユーザが指定した方向に沿っているのが確認できる．さらに，同様のアルゴリズムで丸石だけでなく，形状の不揃いな石も生成することができた．また，出力が 3D であることから，3D モデルを入力とすることで，現実の小石モザイクに見られるような 3D 効果を付与することもできた．このような多数のオブジェクトの集合として絵や模様を表現しようとすると手作業では負荷が高く，多大な時間と負荷を要することが考えられる．本手法ではプロシージャル技術とユーザインタラクションを効果的に組み合わせることで，容易にこれを達成する手段を提案できたと考える．

6.2 今後の課題

本手法は陽にエッジ検出を用いているため，複雑なエッジ検出結果からは意図した配置パターンを生成することは困難である．この解決策としては，ユーザが特徴線を手作業で入力することでエッジ検出を陽に行わない方法と，エッジ検出アルゴリズムを改良する方法などが考えられる．また，本手法では点分布に Poisson-Disk 分布を用いているが，Poisson-Disk 分布では排他的に順次試行を行うため，隙間が目立つ分布となる可能性がある．特に Disk 半径が大きい場合はその傾向が見受けられる．この解決策としてはリラクゼーションなどの点分布の最適化を行う必要性などが考えられる．

謝　辞

本研究を進めるにあたり，多くの方々に多大なご支援を頂きました．この場をお借りして感謝の気持を表したいと思います．指導教官である宮田一乗教授にはご多忙のなか貴重な時間を割いてご指導，御鞭撻と格別の御配慮を賜りました．宮田教授にはグループワーク活動だけでなく，個人の研究でも積極的に学会発表の機会を与えていただき，貴重な経験を得ることが出来ました．また，研究環境をはじめとして日頃の研究活動全般においても様々なご支援をして頂き，深く感謝いたします．審査員の先生方には研究にあたって有益なご指導と助言を賜りました．深く感謝いたします．

また，宮田研究室のメンバーには日頃から研究に対する助言や議論をいただきました．心より御礼申し上げます．同期である石橋賢氏，姜南氏，瀬木宏氏，寺田圭佑氏にはグループワーク活動をはじめとして多くの助言や議論をいただきました．また特別留学生として共にグループワーク活動を行った Arts et Métiers ParisTech(ENSAM) の Tony Da Luz 氏，Rémy Eynard 氏からは多くの刺激を受けました．ここに感謝します．同研究室の先輩である杜暁冬氏には研究の助言だけでなく，日頃の学生生活においてもお世話になりました．櫻井快勢氏と溝口敦士氏には研究面で多大な刺激をいただきました．感謝いたします．

研　究　業　績

【学会発表 (査読あり)】

- **Naoki Kita** and Kazunori Miyata,“A rule-based method for generating bookshelf models”, In ACM SIGGRAPH ASIA 2010 Posters, SA ’10, pages 36:1-36:2, New York, NY, USA, 2010. ACM.
- **北直樹**, 宮田一乗,“本棚シーンのルールベース生成”, Visual Computing / グラフィクスと CAD 合同シンポジウム 2010 予稿集#17,1–6,2010/6/26
- **北直樹**, 宮田一乗,“ルールベースの図書館シーンの自動生成”, 第 9 回 NICOGRAPH 春季大会 論文 & アート部門コンテスト予稿集,SII-2,1–6,2010/3/26
- Ken Ishibashi, Toni Da Luz, Remy Eynard, **Naoki Kita**, Nan Jiang, Hiroshi Segi, Keisuke Terada, Kyohei Fujita and Kazunori Miyata,“Spider Hero: A VR application using pulling force feedback system”, In Proceedings of the 8th International Conference on Virtual Reality Continuum and its Application in Industry, pp.197–202, Japan, 2009
- 石橋賢，Toni Da Luz, Remy Eynard, **北直樹**, 姜南, 瀬木宏, 寺田圭佑, 藤田恭平, 宮田一乗,“スパイダーヒーロー:張力提示システムを導入した VR アプリケーション”, インタラクション 2010 インタラクティブ発表 (プレミアム),2010

【学会発表 (査読なし)】

- 石橋賢，Toni Da Luz, Remy Eynard, **北直樹**, 姜南, 瀬木宏, 寺田圭佑, 藤田恭平, 宮田一乗,“スパイダーヒーロー”, インタラクティブ東京 2009,IVRC(国際学生対抗バーチャルリアリティコンテスト) 参加作品プレゼンテーション,2009
- 石橋賢，Toni Da Luz, Remy Eynard, **北直樹**, 姜南, 瀬木宏, 寺田圭佑, 藤田恭平, 宮田一乗,“スパイダーヒーロー:張力提示システムによる VR アプリケーション”, 第 7 回知識創造支援システムシンポジウム,2010

【展示】

- 石橋賢，Toni Da Luz, Remy Eynard, **北直樹**, 姜南, 瀬木宏, 寺田圭佑, 藤田恭平, 宮田一乗,“Spider Hero”, 第 17 回国際学生対抗バーチャルリアリティコンテスト (IVRC2009), 日本科学未来館,2009 年 10 月 22 日–25 日
- 石橋賢，Toni Da Luz, Remy Eynard, **北直樹**, 姜南, 瀬木宏, 寺田圭佑, 藤田恭平, 宮田一乗,“Spider Hero”, いしかわ夢未来博 2009, 石川県産業展示館,2009 年 10 月 30 日–11 月 1 日
- 石橋賢，Toni Da Luz, Remy Eynard, **北直樹**, 姜南, 瀬木宏, 寺田圭佑, 藤田恭平, 宮田一乗,“Spider Hero”, 第 15 回学生 CG コンテスト (ビデオ展示), 国立新美術館,2010 年 2 月 3 日–14 日
- 石橋賢，Toni Da Luz, Remy Eynard, **北直樹**, 姜南, 瀬木宏, 寺田圭佑, 藤田恭平, 宮田一乗,“Spider Hero”,Laval Virtual 2010, Place de Herce(France),2010 年 4 月 7 日–11 日

参 考 文 献

T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008. ISBN 987-1-56881-424-7.

S. Battiato and G. Puglisi. 3d ancient mosaics. In *Proceedings of the international conference on Multimedia*, MM '10, pages 1571–1574, New York, NY, USA, 2010. ACM.

S. Battiato, G. D. Blasi, G. Farinella, and G. Gallo. A novel technique for opus vermiculatum mosaic rendering. In *14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'06)*, pages 133–140, 2006.

S. Battiato, G. di Blasi, G. Gallo, G. C. Guarnera, and G. Puglisi. A novel artificial mosaic generation technique driven by local gradient analysis. In *ICCS (2)'08*, pages 76–85, 2008.

G. D. Blasi, G. Gallo, and P. Maria. Puzzle image mosaic. In *In proceedings of IASTED/VIIP2005*, 2005.

G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang. Interactive procedural street modeling. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.

R. L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5:51–72, January 1986.

M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, April 2008. ISBN 3540779736.

G. di Blasi and G. Gallo. Artificial mosaics. *The Visual Computer*, 21(6):373–383, 2005.

Y. Dobashi, T. Haga, J. Henry, and T. Nishita. Method for creating mosaic images using voronoi diagrams. In *Proceedings of Eurographics 2002 Short Presentations*, Eurographics 2002, pages 341–348, Saarbrucken, Germany, 2002.

V. A. Dos Passos and M. Walter. 3d mosaics with variable-sized tiles. *Vis. Comput.*, 24: 617–623, July 2008.

V. A. Dos Passos and M. Walter. 3d virtual mosaics: Opus palladium and mixed styles. *Vis. Comput.*, 25:939–946, September 2009.

D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2002. ISBN 1558608486.

G. Elber and G. Wolberg. Rendering traditional mosaics. *The Visual Computer*, 19: 67–78, 2003.

G. Faustino and L. de Figueiredo. Simple adaptive mosaic effects. In *Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on*, pages 315 – 322, 2005.

J. Gain, P. Marais, and W. Straß er. Terrain sketching. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 31–38, New York, NY, USA, 2009. ACM.

A. Hausner. Simulating decorative mosaics. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 573–580, New York, NY, USA, 2001. ACM.

H. Hnaidi, E. Guèrin, S. Akkouche, A. Peytavie, and E. Galin. Feature based terrain generation using diffusion equation. *Computer Graphics Forum (Proceedings of Pacific Graphics)*, 29(7):2179–2186, 2010.

M. Howarth. *The Complete Pebble Mosaic Handbook*. Firefly Books, 2009. ISBN 978-1554074181.

T. Ijiri, R. Mêch, T. Igarashi, and G. Miller. An Example-based Procedural System for Element Arrangement. *Computer Graphics Forum*, 27(2):429–436, April 2008.

T. Itoh, K. Miyata, and K. Shimada. Generating organic textures with controlled anisotropy and directionality. *IEEE Comput. Graph. Appl.*, 23:38–45, May 2003.

N. Kita and K. Miyata. A rule-based method for generating bookshelf models. In *ACM SIGGRAPH ASIA 2010 Posters*, SA '10, pages 36:1–36:2, New York, NY, USA, 2010. ACM.

Y.-K. Lai, S.-M. Hu, and R. R. Martin. Surface mosaics. *Vis. Comput.*, 22:604–611, September 2006.

E. Langetepe. *Geometric Data Structures for Computer Graphics*. A K Peters Ltd, 2006. ISBN 978-1568812359.

M. Lipp, P. Wonka, and M. Wimmer. Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics*, 27(3):102:1–10, Aug. 2008. Article No. 102.

Y. Liu, O. Veksler, and O. Juan. Simulating classic mosaics with graph cuts. In *Proceedings of the 6th international conference on Energy minimization methods in computer vision and pattern recognition*, EMMCVPR'07, pages 55–70, Berlin, Heidelberg, 2007. Springer-Verlag.

Y. Liu, O. Veksler, and O. Juan. Generating classic mosaics with graph cuts. volume 29, pages 2387–2399. Blackwell Publishing Ltd, 2010.

S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

K. Miyata. A method of generating stone wall patterns. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '90, pages 387–394, New York, NY, USA, 1990. ACM.

K. Miyata, T. Itoh, and K. Shimada. A method for generating pavement textures using the square packing technique. *The Visual Computer*, 17(8):475–490, 2001.

T. Noda and K. Miyata. Mosaic image generation using 3d models. ASIAGRAPH 2009 PROCEEDINGS, pages 110–115, 2009.

K. Nomura and K. Miyata. Automatic generation method for processing plants. In *Proc. of the IIEEH Image Electronics and Visual Computing Workshop 2010, 3B-6*, pages 1–6, 2010.

Y. I. H. Parish and P. Muller. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, pages 301–308, New York, NY, USA, 2001. ACM Press.

A. Peytavie, E. Galin, J. Grosjean, and S. Mèrillou. Procedural generation of rock piles using aperiodic tiling. *Comput. Graph. Forum*, 28(7):1801–1809, 2009.

Procedural Inc. CityEngine. http://www.procedural.com/, 2010.

E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. *ACM Trans. Graph.*, 25:1294–1326, October 2006.

E. Zhang, J. Hays, and G. Turk. Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13:94–107, January 2007.

H. Zhou, J. Sun, G. Turk, and J. M. Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, July/August 2007.

昌達. **詳解 画像処理プログラミング**. ソフトバンククリエイティブ, 2008. ISBN 978-4797344370.

櫻井 and 宮田. 地表に無造作に配置された岩石の生成手法. NICOGRAPH 秋季大会, pages 1–7, 2010.

石原. **テンソル—科学技術のために**. 裳華房, 1991. ISBN 978-4785310684.