A Study of Methods for Generating Aggregates in Computer Graphics

by

Kaisei Sakurai

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Kazunori Miyata

School of Knowledge Science
Japan Advanced Institute of Science and Technology

March 2013

Acknowledgements

Though only my name appears on the cover of this dissertation, a great many people have contributed to this work. Without helps from all of associates and friends, I would not be able to write this dissertation.

First and foremost, my deep gratitude goes to my supervisor, Professor Kazunori Miyata. Owing to his kindness, guidance, encouragement, and collaboration, I solved some problems. Discussions with him gave me stimulations for my studies, and his activity had me learn how to get successes.

Professor Yukari Nagai has supported me as the supervisor for minor research. Owing to her collaboration, I could attempt to find out a geometrical solver for a pair of simple ring puzzle. I am grateful to her for the long discussions that determine this theme.

I am grateful to Professor Takayuki Ito from Ochanomizu University, Professor Susumu Kunifuji, Professor Kazushi Nishimoto, and Professor Takashi Hashimoto from JAIST for serving on my dissertation committee. Thanks to good questions in the preliminary defense, I could consider contributions for knowledge science well.

I thank Mr. Shigeru Komatsubara, Mr. Goro Motai, Mr. Naoki Kawai, and Mr. Kazuo Matsufuji from Dai Nippon printing Co., Ltd. I could proceed to the doctoral program in JAIST owing to their encouragement.

I appreciate the support of my friends and members in Miyata laboratory. Without the support, I would not enjoy this student life.

Abstract

With the improvement in computer graphics (CG), the workload on creators is increasing to achieve the requirements necessary for the production of high-quality artwork, images, and movies. To reduce the time required for the creation of such media, it is necessary to develop methods that assist creators in their work. In CG, there are several phases. This dissertation focuses on the shape modeling phase, looking at the generation of aggregates as a field in need of assistance. Shape modeling is a creation phase in which objects are formed for the production of CG. In this phase, there are several operations that currently require intensive manual operations. One of these operations is the modeling of an aggregate composed of many components that are nonperiodically arranged. It is difficult to determine the positions and orientations of many components because of the complexity of the interrelationship of each component. To reduce this workload, this dissertation presents three methods for automatically determining the positions and orientations of the components: generating both 2D and 3D aggregates consisting of arbitrary-shaped components, and generating an aggregate composed of various-shaped components. This dissertation focuses on staple fibers as one of the various-shaped components.

The first method generates a 2D aggregate. To do so, a dart-throwing method using arbitrary exclusive regions is developed. This method randomly places a circular exclusive region without overlapping the placed exclusive regions for Poisson disk distribution. In the proposed method, the components comprising the aggregate are arranged nonperiodically without any overlaps. This method rapidly generates the aggregate by filling components in the gaps among already placed components. By using this method, aggregates of toys, forests, and leather textures can be generated.

The second method generates a 3D aggregate of piled components. The inside of a piled aggregate is generally filled. However, the second proposed method piles components only near the surface of the aggregate and does not fill the inside. This method does not use a physical simulation to form arbitrary-shaped aggregate because such simulations constrain the shape, and thus the results are limited. While a physical simulation makes a physically plausible shape, the shape has to be within the limits of physical existence. In forming various shapes, an aggregate is generated by a geometric operation without physical simulation. The first procedure of such a method distributes components only on the surface of the target aggregate shape. Then, a pile is constructed by refining the positions of the components. By this procedure, arbitrary-shaped aggregates, such as a rice ball, are generated.

The third method generates 3D aggregates composed of staple fibers, such as a dust ball. A staple fiber is used for making cloth and is generated daily from our bodies, and dust balls composed of staple fibers frequently appear; however, it is too difficult to express these aggregates. Thus, this method generates staple fibers procedurally using a chain of line segments as a fiber. Here, the curve and length of a fiber are parameterized and it is easy to change the fiber by changing the parameters. Thus, varied staple fibers can be generated without manual operation. Next, an aggregate is generated so that the generated fibers are converged into the target aggregate shape by adjusting the positions of the staple fibers. By this method, we can obtain arbitrary-shaped aggregates composed of staple fibers.

These proposed methods reduce the workload of manual manipulation for creating objects in CG. Thanks to these methods, creators can spend their time on more creative work.

Contents

Contents **iv**

List of Figures **vii**

List of Tables **x**

1 Introduction **1**
1.1 Overview 2
1.2 Structure of this dissertation 4

2 Related Work **5**
2.1 Point Set 6
2.1.1 Dart-Throwing Method 7
2.1.1.1 Sequential Method 8
2.1.1.2 Parallel Method 10
2.1.1.3 Other Methods 11
2.1.2 Relaxation Method 12
2.1.3 Analysis of Point Set 15
2.2 Procedural Modeling 15
2.3 Nonparametric Modeling Approach 18
2.3.1 Texture Synthesis 18
2.3.2 Placement 20
2.4 Physically-based Modeling 23
2.5 Discussion 23

3 2D Aggregate Generation Method **26**
3.1 Introduction 27
3.2 Preliminary Experiment 28
3.2.1 Exclusive Region 28

3.2.2 Procedure for Naive Dart-throwing Method 29
3.2.3 Results of Preliminary Experiment 31
3.3 Fill-Gap Method 35
3.3.1 Sparse Placement 36
3.3.2 Placement in Gaps 37
3.4 Result 38
3.4.1 Calculation Time 38
3.4.2 Changing Exclusive Region 39
3.4.3 Example 39
3.5 Evaluation 39
3.6 Discussion and Conclusion 40

4 3D Aggregate Generation Method 48
4.1 Introduction 48
4.2 Overview of Aggregate Modeling 50
4.3 Aggregate Modeling Procedure 52
4.3.1 Component Placement 52
4.3.1.1 Overview of Placement 52
4.3.1.2 Distribution Radius 53
4.3.1.3 Implementation of Dart-throwing Method 54
4.3.1.4 Orientation 55
4.3.1.5 Placement of Multiple Types of Components . . . 55
4.3.2 Refinement 56
4.3.2.1 Quantification of Overlap 56
4.3.2.2 Minimization 57
4.3.2.3 Removal of Components 58
4.3.2.4 Multiple Types of Components 58
4.4 Results 59
4.5 Evaluation 62
4.6 Limitation and Future Work 65

5 Aggregate of Staple Fibers 70
5.1 Introduction 71
5.2 Process Overview 71
5.2.1 Procedural Modeling of Staple Fiber 72
5.2.2 Forming Aggregate 72
5.2.2.1 Initial Placement 73

5.2.2.2 Adjustment 74
5.3 Result 76
5.4 Evaluation 81
5.5 Conclusion 85

6 Conclusion and Future Work 86
6.1 Summary and Contribution 86
6.2 Limitation and Future Work 87

References 90

List of Figures

1.1 Picture of part of my lunch. 2
1.2 Fabricated rice ball. 3

2.1 Poisson disk distribution. 6
2.2 Dart throwing. 7
2.3 Point set of multiple classes of the Wei method [118]. Total set and each class. 9
2.4 Available region of the Jones method [49]. 9
2.5 Available region of Dunbar and Humphreys' method [28]. 10
2.6 Point set using particle system [33]. 14
2.7 Centroidal Voronoi diagram for line segment [42]. 14
2.8 Noise-based texturing of Perlin [91]. 16
2.9 Cellular texturing of Worley [123]. 17
2.10 Stone aggregate of Peytavie et al. [92]. 17
2.11 Texture synthesis of Hurtut et al. [45]. 19
2.12 Texture synthesis of Ma et al. [71]. 20
2.13 3D collage of Gal et al. [34]. 22
2.14 Distribution of component of Lagae and Dutre [60]. 22
2.15 Small aggregates for generating large aggregates. 25

3.1 Comparison of results. 27
3.2 Specification of an exclusive region. 29
3.3 Components and exclusive regions. Images (a)–(c) are components and (d)–(f) are their exclusive regions. The black region denotes an inner region, and a magenta pixel indicates a vertex in (d)–(f). 30
3.4 Result of the naive dart-throwing method (210 components). . . . 30
3.5 Relationship between number of consecutive cancelations, number of components, and calculation time of Fig. 3.3. 31

3.6 Results of each of three types of components. 32
3.7 Relationship between number of consecutive cancelations, number of components, and calculation time of Fig. 3.6. 33
3.8 Overview of the proposed method. 34
3.9 Circular region O. The green circle indicates the circular region's boundary for each component. 35
3.10 Comparisons of yielded points Q (orange). 36
3.11 Comparisons of points P (orange) chosen from Fig. 3.10. 37
3.12 Exclusive regions placed by the proposed method. 40
3.13 Generated aggregates. 41
3.14 Variations of densities. 42
3.15 Variations of examples. 43
3.16 Generated height maps of leather texture # 1. 44
3.17 Generated height maps of leather texture # 2. 45
3.18 Generated height maps of leather textures # 3. 46

4.1 Generated aggregates. 49
4.2 Process overview. 51
4.3 Constructing a pile structure from overlapped components in the refinement step. **F** indicates a part of the aggregate shape. 52
4.4 Two spheres in contact with radii $2r$ (dotted line). 53
4.5 Largest inner sphere. 54
4.6 Two components in contact. e_2's center is located at a distance of $4r$ from e_1's point. 54
4.7 Placement of multiple types of components. Chestnuts and rice in a rice-ball shape. 56
4.8 Color mapping based on the function values. 58
4.9 Refinement of multiple types of components. 59
4.10 Comparison of generated aggregates. 60
4.11 Aggregates from multiple types of components. 61
4.12 Changing the range for collision. 62
4.13 Controlling component orientation. 63
4.14 Orientation in the global coordinate system. 63
4.15 Inside of the solid texture. 65
4.16 Variations in results of proposed method. 66
4.17 Fabrication example. 67
4.18 Fabrication example # 2. 68

4.19 Broken printed aggregate. 68

5.1 Generated staple fiber. 72
5.2 Polygonal chain model. 73
5.3 Process overview. 73
5.4 Initial placement of fiber. 74
5.5 Adjustment. 75
5.6 Variations in bunny shape. 77
5.7 Variations in dragon shape. 78
5.8 Changing fiber length. 80
5.9 Changing the number of fibers. 80
5.10 Scattered staple fibers. 81
5.11 Bird nest. 82
5.12 Sphere-shaped aggregate. 83
5.13 Gaussian-shaped aggregate. 84

List of Tables

2.1 Comparison of characteristics. $\surd$ indicates the available characteristics. agg., dens., and arb., are abbreviations for aggregate, density, and arbitrary components, respectively. 24

3.1 Comparison of calculation time between the proposed method and the naive dart-throwing method 39

4.1 Parameters, calculation time, and number of components 64

5.1 Staple fiber parameters 72
5.2 Parameters for Fig. 5.6 79
5.3 Parameters for Fig. 5.7 79
5.4 Parameter for Fig. 5.8 79
5.5 Parameter for Fig. 5.9 79
5.6 Parameters for Figs. 5.10 – 5.12 85

Chapter 1

Introduction

Shape modeling is an indispensable phase in computer graphics (CG), which produces objects in a computer.

A creator spends a great deal of time in modeling an object. Thus, a method of assisting creators in rapidly producing CG is desirable. With the aim of reducing the time required for modeling objects, researchers have proposed numerous shape modeling methods; however, these methods cannot generate each and every types of object.

With the aim of assisting shape modeling more effectively, I interviewed three creators to discover what tasks occupy most of their time. The creators are all professionals involved in creating commercial and industrial CG, and one is a winner of a CG contest. In the interview, a creator said "We often create daily scenes that include many objects. There are many nonperiodic aggregates in these daily scenes. In 2D aggregates, it is necessary to consider the balance of gaps and densities of the objects. In representing 3D aggregates, we need to pile components manually. In addition, we evaluate an aggregate by comparing variations. Both 2D and 3D aggregates need many manual operations, and we spend a lot of time on such tasks." For example, one nonperiodic 3D aggregate is rice, which appears in daily scenes, as shown in Fig. 1.1. There are many grains of rice in the picture, and it is exceedingly difficult to manually create such an aggregate using CG. It would be helpful to be able to automatically generate such aggregates. The creators commonly prefer optimized tools over a generalized tool. Therefore, this dissertation proposes two optimized methods for generating 2D and 3D aggregates.

Here, variations in components for assisting creators are also considered. A procedural modeling method normally produces many variations. This disserta-

Figure 1.1: Picture of part of my lunch.

tion will demonstrate a method that produces an aggregate from varied fibers generated by a procedural modeling method.

Three methods are proposed to automatically generate three types of aggregates. An overview of the methods and the structure of this dissertation are described in Sections 1.1 and 1.2, respectively. By using these three methods, it is easy to express various aggregates. The results of the proposed methods are geometric surfaces, which can be fabricated by a 3D printer, for example, rice balls shown in Fig. 1.2.

1.1 Overview

The targets of the proposed methods are aggregates composed of 2D and 3D components, which we frequently see in our daily life. The workload for creating aggregates can be reduced if automatic aggregate generation methods are developed. In proposing such useful methods, this dissertation considers controlling the arrangement of these components to obtain the desired aggregates. The proposed methods generate an aggregate by geometric transformations that are controlled by parameters used for controlling the arrangement. Nonperiodic

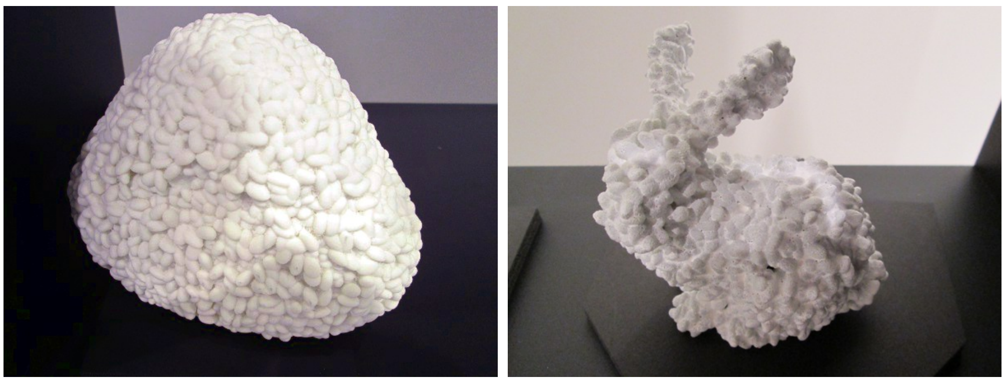


Figure 1.2: Fabricated rice ball.

arrangements are represented by the Monte Carlo method. Many methods use physical simulations for realistic representation [20]; however, the proposed methods do not use physical simulations because the simulations limit the shapes to physically correct ones.

First, a method for generating a 2D aggregate is presented. For this, an extension of the dart-throwing method [22] that can place arbitrary shapes is developed. The calculation time required for exclusive placement is very long, and so to reduce it, an effective method of placements was developed. The results produced by this method show that satisfactory nonperiodic arrangements of arbitrary-shaped components are obtained and that the densities of the arrangements can be controlled. The developed method improves interactivity in creating 2D aggregates.

Most 3D aggregates consist of components that have been piled up, and each component touches its neighbors. However, the method used for 2D aggregates cannot be applied to 3D aggregates because not all components touch their neighbors in the 2D aggregates. This dissertation presents a method that locally constructs piles around the arbitrary surface of a target aggregate to generate piled aggregates. As a result, the required appearances are obtained. The proposed method removes some of the components used to obtain a 3D aggregate, and they can be applied to a 2D aggregate; however, this removal procedure is time consuming for generating 2D aggregates because it increases the process length.

While these methods for 2D and 3D aggregates vary the arrangement of components, the shapes of the components are static. In the real world, all components have different shapes. Thus, the current methods for generating 2D and 3D aggregates are not sufficient to express variations. To express such variations, this dissertation attempts to produce aggregates from objects generated by a procedural modeling method. A staple fiber is chosen as a modeling target because there are numerous aggregates composed of staple fibers in our daily life, such as dust. In addition, fiber modeling is exceedingly difficult because the fiber is deformable and there are numerous variations in fiber shapes. Therefore, this dissertation presents a method for generating an aggregate composed of staple fibers as an example of generating an aggregate procedurally from a scratch. Existing methods cannot model an aggregate of staple fibers. The proposed method generates fiber shapes procedurally to produce an aggregate, and then correlates the generated fibers to form a user-specified shape.

Last, I interviewed two professional groups of creators regarding the usability and results of the three proposed methods to evaluate them.

1.2 Structure of this dissertation

The structure of this dissertation is organized as follows.

Chapter 2 discusses nonperiodic patterns as related works for generating aggregates. The chapter consists of five sections. The first section presents a nonperiodic point set, which is considered to be one of the main nonperiodic patterns. In addition, an aggregate is a type of nonperiodic pattern. A survey of the point set is conducted to understand the possible methods for generating an aggregate. The second and third sections respectively present procedural and nonparametric approaches for generating an aggregate, and the fourth section presents a physically-based approach for generating them. Finally, the limitations of these modeling methods are discussed.

Chapters 3, **4**, and **5** present the three proposed methods as described in Section 1.1. **Chapter 3** describes a proposed method for effectively generating a 2D aggregate after a preliminary experiment is presented for proposing an effective method. **Chapter 4** presents a procedure for modeling a 3D aggregate and shows objects fabricated from the generated aggregates by means of a 3D printer. **Chapter 5** proposes a method for generating an aggregate composed of staple fibers. This chapter shows variations in fibers generated by procedural modeling and aggregates composed of these fibers. The evaluation section in each chapter presents interviews with creators.

Chapter 6 summarizes what was achieved in this dissertation and discuss the limitations of the methods. Finally, I conclude this dissertation by describing areas for future research.

Chapter 2

Related Work

This chapter surveys two previous study fields; the nonperiodic point set and generation of objects.

Section 2.1 presents a study of the nonperiodic point set. In this study, methods to distribute and analyze a nonperiodic point set are explored. The oldest method [22] for the distribution of the nonperiodic point set produces enough quality results; however, the calculation cost is high. Thereafter, many researchers have focused on more effective distribution methods for the nonperiodic point set. Ordinarily, controllability of point distribution has been desired for practical use. This study aims to generate a nonperiodic aggregate, and there is common knowledge to arrange the components comprising an aggregate.

Sections 2.2, 2.3, and 2.4 present studies of the generation of objects. Controllability comes before computation cost in these studies, and it is more important to express desired objects than to propose effective calculations. In the generation of objects, there are three approaches for generating aggregates: procedural modeling, nonparametric modeling, and physically-based modeling. The procedural modeling approach, which is described in Section 2.2, automatically generates aggregates from specified parameters. The nonparametric modeling approach, which is described in Section 2.3, directly specifies characteristics for generating aggregates such as objects and structures. Physically-based modeling, which is described in Section 2.4, produces aggregates by means of a physical simulation.

The objective of several previous methods is similar to that of this study; however, they all have limitations, as discussed in Section 2.5.

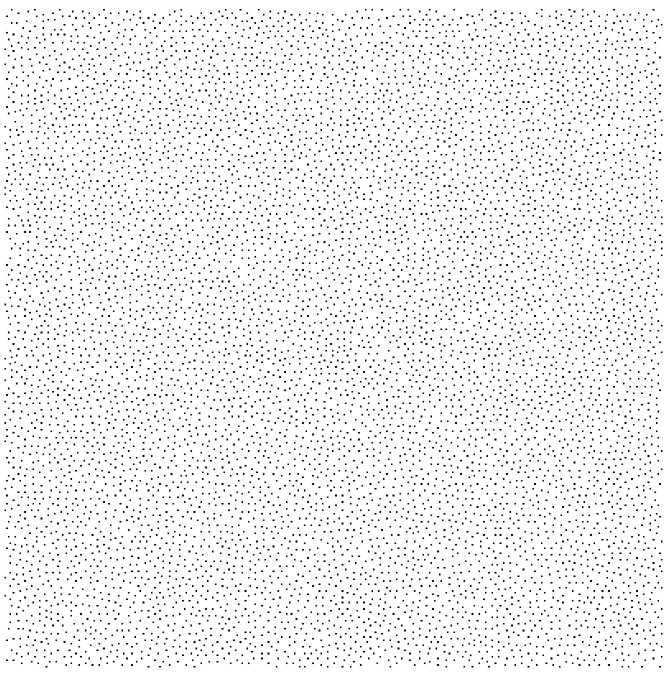


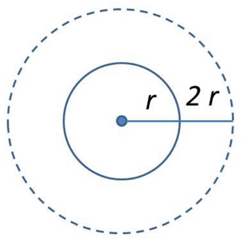
Figure 2.1: Poisson disk distribution.

2.1 Point Set

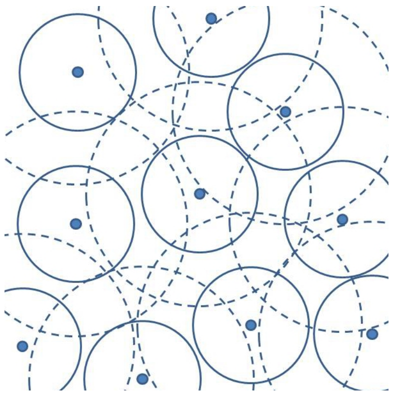
The simplest nonperiodic point set is given by random distribution, which is called Poisson distribution. The distributed points depend on only random function and the specified density of the points. Poisson distribution is simple, but has not been investigated in depth.

On the other hand, Poisson disk distribution has been explored in depth. Poisson disk distribution generates a point set in which a point keeps a specified distance from its neighboring points. The appearance of the generated point set is isotropic, as shown in Fig. 2.1. Each point has a circular region whose radius is r, the point is located at the center of the circular region, and there is no collision among any of the circular regions in the point set. The point set is widely used for the distribution of objects [60], rendering [96], procedural modeling [100], and so on.

To implement a distribution method, there are two major approaches: dart throwing and relaxation. Dart-throwing methods deterministically place points [22]. These methods place a circular region with a radius of $2r$ at a placed point, as shown in Fig. 2.2 (a). The solid-line circle and the broken-line circle in this figure indicate the outlines of circular regions, whose radii are r and $2r$,



(a) Point with Circular region.



(b) Dart throwing.

Figure 2.2: Dart throwing.

respectively. A random function specifies the position of the point. The point is placed if the specified position is located outside already placed circular regions. Otherwise, the point is not placed at that position. These methods iteratively place a point until a termination condition is satisfied. In the result, there is no collision among placed circular regions (solid-line circles), as shown in Fig. 2.2 (b). Details of dart-throwing methods are described in Section 2.1.1.

The relaxation methods refine the positions of placed points to obtain ideal distribution. To refine the positions appropriately, most of the methods apply minimized energy functions, which indicate the difference between the ideal condition and condition of the placed point set. Details of relaxation methods are presented in Section 2.1.2.

There are effective methods that use a Wang tile [115] for generating a Poisson disk distribution [58, 60, 61]. The Wang tile effectively generates a nonperiodic pattern by tiling square patches. A point set is rapidly generated by preparing patches that contain distributed points.

In Section 2.1.3, spectral analyses are presented to evaluate the properties of a point set, which indicate the conditions of related positions among placed points.

2.1.1 Dart-Throwing Method

The first implementation of the dart-throwing method was proposed by Cook [22]. In this method, random positions are initially prepared as a lookup table. A position in the lookup table is discarded if the position is closer than a specified distance from already placed positions. The positions remaining in the lookup table are a point set of the Poisson disk distribution. The Poisson disk distribution can be obtained in a straightforward manner; however, the method involves a high calculation cost because of the large size of the lookup table required for filling the region with placing points.

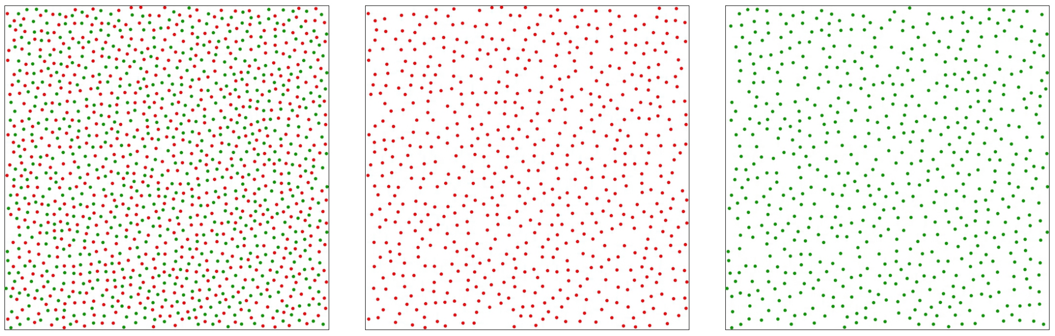
McCool and Fiume proposed a method called the relaxation-dart-throwing method [75]. In this method, points are placed within a circular region of large radius initially, and the radius is gradually reduced in order to place a large number of points. Points are preferentially filled into large gaps among placed points because a large circle is placed into only large gaps among placed circles. The results of the method show a certain number of points are placed, and this ensures stable sampling. In addition, the distribution of a generated point set has more isotropy than Cook's method. However, the procedure converges a point set to partially hexagonal periodic patterns [33, 75].

One drawback of all of above methods is that the calculation times are long. To reduce the calculation time, researchers have proposed adopting more effective methods such as sequential methods (Section 2.1.1.1), parallel methods (Section 2.1.1.2), and other methods (Section 2.1.1.3).

In addition, there is a method that focuses on the color of placed points. Most Poisson disk distribution methods place a point set without considering the colors of the placed points. In contrast, Wei proposed a dart-throwing method that considers the colors [118]. A set of points having the same color is called a class. Wei's method places a point set having multiple classes, as shown in Fig. 2.3. The distributed points in each class (Figs. 2.3 (b) and (c)) and the amount of both classes (Fig. 2.3 (a)) are isotropic. The method adjusts radius r of a circular region and the mixture ratios of the dart-throwing method for each class in order to place the point set.

2.1.1.1 Sequential Method

Mitchell proposed an effective dart-throwing whose calculation cost is $O(N^2)$ [76]. The termination condition of Cook's method [22] is that a specified region is filled with circular regions; however, the termination condition is unexpected because the position of a placing point is randomly chosen. In contrast, the method proposed by Mitchell has an obvious termination condition. Until the desired number of placed points reaches a specified number, points are placed by the following two-step iteration. In the first step, several random points are scattered. In the second step, the closest point having the furthest distance from



(a) Total set. (b) Class red. (c) Class green.

Figure 2.3: Point set of multiple classes of the Wei method [118]. Total set and each class.

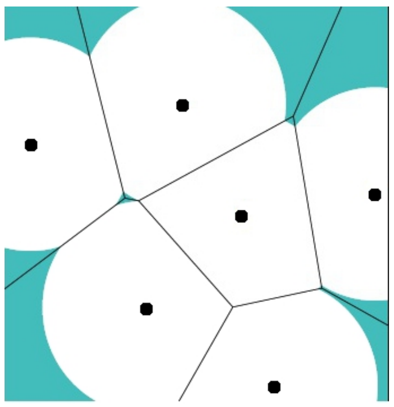
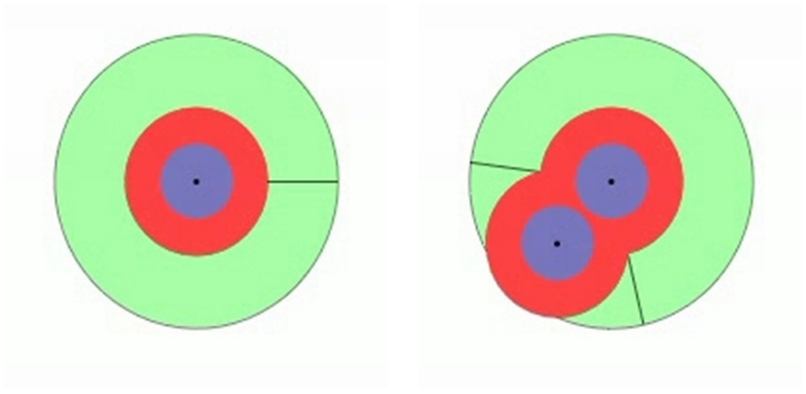


Figure 2.4: Available region of the Jones method [49].

already placed points is placed in the scattered points. Jones proposed a method having a lower calculation cost than the above-mentioned methods [49]. This method runs in $O(NlogN)$ time for N points by reducing the available regions in which points can be placed. In the iteration of placement, a point is randomly placed in the available regions. An available region is an intersection of the outside of its circular region and the inside of its Voronoi cell, as shown in Fig. 2.4. Here, black points indicate placed points, green regions indicate available regions, and white regions indicate the union of circular regions with radii of $2r$. Dunbar and Humphreys also proposed a low-calculation cost method [28]. In order to run in $O(NlogN)$, this method reduces available regions by specifying them using a scalloped sector, which is a sector bounded by circular arcs above and below, as shown in Fig 2.5 (a). The black point indicates a placed point,



(a) Single point. (b) With placed neighbor.

Figure 2.5: Available region of Dunbar and Humphreys' method [28].

and the violet, red, and green regions indicate circular regions with radii of r, $2r$, and $4r$, respectively. The green region is an available region. Figure 2.5 (b) shows that the available region is reduced after one neighboring point is placed. As a result, the calculation time is approximately $O(N)$. Bridson proposed an appropriate grid for distribution [17] in which the diagonal of each cell is r; therefore, the cell size is bounded by $r/\sqrt{n}$ in n-dimension. Using an appropriate grid, at most one point is placed in a cell. Therefore, there is one point or no point in a cell. A cell that contains a point and its neighboring cells are flagged to indicate that an entire cell or a part of a cell is inside a circular region. The method detects conflicts among circular regions by checking only flagged cells; hence, the calculation cost is low. An appropriate grid is used by other methods [14, 29, 117]. Gamito and Maddock proposed an effective method that places points outside of placed circular regions [35]. In addition, the calculation cost is $O(NlogN)$. Ebeida et al. used an appropriate grid optimized to run in a low calculation time, $O(N)$ [29]. This method requires a lower memory resource than the above-mentioned methods. In addition, it effectively produces a point set in arbitrary-shaped available regions.

The above sequential methods reduce the calculation costs by limiting available regions or the size of the lookup table.

2.1.1.2 Parallel Method

Wei proposed a parallel calculation method for Poisson disk distribution [117]. This method places point sets on appropriate grids [17] at multiple resolutions using a graphics processing unit (GPU). In addition, the method can adaptively distribute points in different spatial densities by adapting the radii of the circular regions in the dart-throwing method. This parallel method runs faster than sequential algorithms.

Bowers et al. extended Wei's method [117] for isotropic sampling on a three-dimensional (3D) surface [14]. Similar to Wei's method, the method of Bowers et al. divides a 3D surface into appropriate grids on geodesic distance. In general, the geodesic distance is calculated by a sequential algorithm [77, 108] consisting of two steps: (1) finding the shortest path on the surface of a 3D mesh, and (2) searching the geodesic distance between the vertices of the shortest path. However, parallelization is needed for estimating geodesic distances when applying Wei's method to a 3D surface. Bowers et al. proposed a method for approximating the geodesic distance d_g between two points $\mathbf{p}_1$ and $\mathbf{p}_2$ without using sequential methods. This approximation method can calculate d_g in random order; therefore, it can be parallelized. The method assumes that there is a smooth curve on the surface that passes through $\mathbf{p}_1$ and $\mathbf{p}_2$, and d_g is estimated as the length of the smooth curve. The smooth curve is defined by interpolating two cosine angles obtained from normals $\mathbf{n}_1$ and $\mathbf{n}_2$ at $\mathbf{p}_1$ and $\mathbf{p}_2$ on the surface. The method estimates d_g by calculating the direct integral of the smooth curve, as given by Eq. 2.1:

$$d_g = \int_0^1 \frac{d_e}{\sqrt{1 - \{(1-t)c_1 - tc_2\}^2}} dt, \tag{2.1}$$

where $d_e = ||\mathbf{p}_2 - \mathbf{p}_1||$, $\mathbf{v} = (\mathbf{p}_2 - \mathbf{p}_1)/d_e$, $c_1 = \mathbf{n}_1 \cdot \mathbf{v}$, and $c_2 = \mathbf{n}_2 \cdot \mathbf{v}$. This method also runs faster than sequential algorithms by parallelization.

Xiang et al. proposed another parallel method for generating a point set using a geodesic distance metric [125], which directly implements Cook's method [22]. The parallel method generates a lookup table using the point-distribution method on a surface [84]. A thread in a GPU chooses a point on the lookup table, and determines whether the chosen point can be placed.

2.1.1.3 Other Methods

Ostromoukhov produced effective calculations using tiles to reduce the available regions in which points can be placed. One of his papers presents a method for generating a point set given an importance density [86]. The plane on which points are placed is hierarchically subdivided into Penrose tiling to create sufficient sample points, which are numbered using the Fibonacci number system. These numbers are used to set the threshold value for the samples against the local value of the importance density. By using Penrose tiling, the distribution of a point set is isotropic and random. In addition, another method based on self-similar tiling of a plane or the surface of a sphere with polyominoes [85] has been proposed for hierarchical importance sampling. Sampling points are associated with polyominoes, one point per polyomino. Each polyomino is recursively subdivided until the desired local density of samples is reached. A numerical code generated during the subdivision process is used for determining a threshold to accept or reject the sample. The exact position of the sampling point within the polyomino is determined according to a structural index that indicates the polyomino local neighborhood. The variety of structural indices and associated sampling point positions are computed during the offline optimization process, and then tabulated. Consequently, the sampling is extremely fast because the calculation cost for constructing the graph is low.

2.1.2 Relaxation Method

In the method proposed by McCool and Fiume [75], first, Lloyd relaxation [70] using a Voronoi diagram is introduced after the relaxation-dart-throwing method. This relaxation has an iteration process consisting of three steps. In the first step, a Voronoi diagram is constructed from already placed points as sites, and these sites divide a plane into Voronoi cells. Any positions in a Voronoi cell are closer to its site than other sites. In the second step, the centroid of every Voronoi cell is calculated. In the final step, every placed point is moved to the centroid of its Voronoi cell. The points are iteratively moved until every point is located on the centroid of its Voronoi cell. The point set generated by relaxation has a strong isotropy because each placed point is constrained so as to maintain a specific relative distance from each of its neighbors. In applications such as generating mosaics [23, 55] and remeshing [1, 64, 68], the relaxation method is called centroidal Voronoi tessellation or centroidal Voronoi diagram. A drawback of the relaxation method is that the randomness of sampling decreases because regular hexagonal lattices are produced. Balzer et al. proposed the capacity-constrained point-distribution method in order to avoid this drawback [4]. This distribution method applies a capacity-constrained method to a centroidal Voronoi diagram. In constructing the Voronoi diagram, each cell is constrained so that its area is uniform. This method produces a point set by the center of each cell as a point.

Because the areas of the cells are uniform, the density of the placed points in the cells is also homogeneous.

As another approach, methods that move points as particles along formulated energies have been proposed. Fattal presented an approach for generating a point set using a statistical mechanics interacting particle model [33]. To control randomness of placed points, the formulation of the particle model unifies the randomness with the requirement for equal distance between points, as shown in Fig. 2.6. He assumed that each point is equally important and assigns an equal amount of matter, for example integrated density, to every kernel as a formulated function. The kernel converges the average point density on target density $\rho(\mathbf{x})$, where x is a point in $\Omega \subset \mathbb{R}^d$ and Ω is the domain of interest. The j-th kernel centered around the point $\mathbf{x}_j$ is defined by Eq. 2.2:

$$K(\mathbf{x}, \mathbf{x}_j) = \frac{1}{\sigma(\mathbf{x}_j)^d} \phi\left(\frac{||\mathbf{x} - \mathbf{x}_j||}{\sigma(\mathbf{x}_j)}\right), \tag{2.2}$$

where $\sigma(\mathbf{x}) = \sqrt[-d]{\rho(\mathbf{x})}$. Then, point density at $\mathbf{x}$ is approximated by Eq. 2.3:

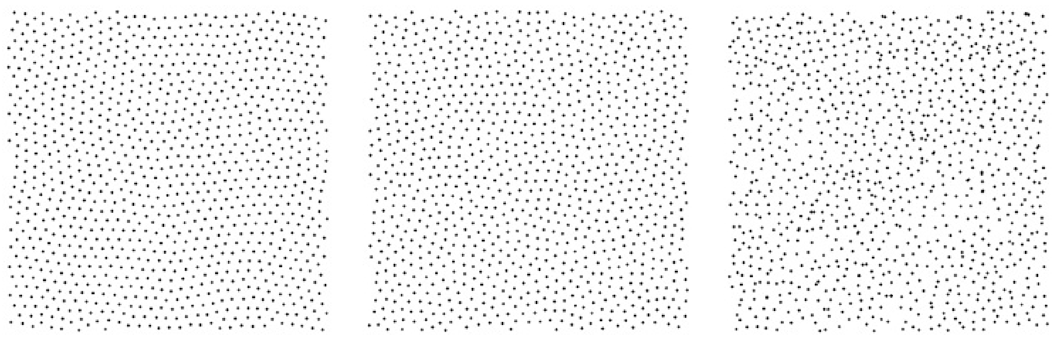
$$A(\mathbf{x}) = \sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j). \tag{2.3}$$

Error of point density E is defined by Eq. 2.4:

$$E = \int_{\Omega} |A(\mathbf{x}) - \rho(\mathbf{x})| dx. \tag{2.4}$$

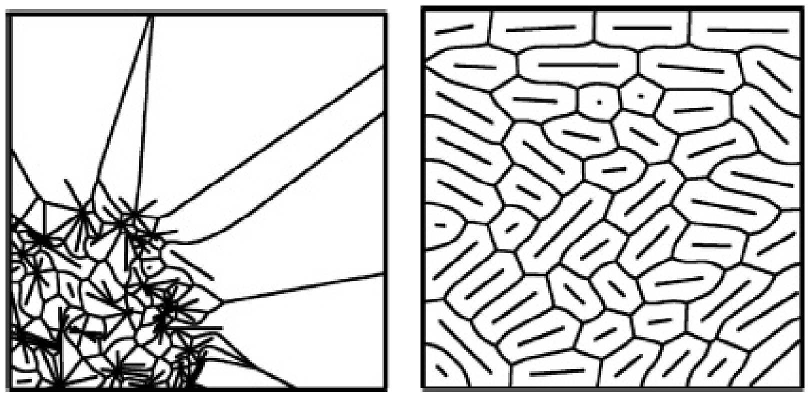
This process uses the Langevin method [38] to produce a new configuration by altering a single point so that E is minimized.

For distributing arbitrary-shaped components, Hiller et al. proposed an extension of the centroidal Voronoi diagram [42]. The method iteratively moves placed components. An ideal distribution lets components spread over their region. Because the boundary of a component maintains a distance from those of its neighbors, the appearance of distributed components is homogenous. In this method, in the first step, a Voronoi diagram is constructed form the components as sites. A Voronoi cell includes a position set of the nearest neighbors from the boundary of a component. In the second step, the component is rotated along its inertia axis so that the main inertia axis is oriented to the largest eigenvector (orientation of the longest diagonal) of the cell. By iterating these steps, an ideal distribution is obtained, as shown in Fig. 2.7.



(a) Low randomness. (b) Middle randomness. (c) High randomness.

Figure 2.6: Point set using particle system [33].



(a) Initial condition. (b) After 150 iteration.

Figure 2.7: Centroidal Voronoi diagram for line segment [42].

2.1.3 Analysis of Point Set

To date, distribution properties of point sets have been analyzed using several methods. Robinson evaluated regular lattices using a Fourier transform [98]; however, only periodic point sets were considered. Dippe and Wold [26] analyzed the distribution of stochastic sampling, which evaluates relative distances between points. Ulichnev [113] used spectra for expressing the properties of a point set. In general, the spectra of a point set **s**, which is composed of n points, is defined by Eq. 2.6:

$$F(\mathbf{f}) = \sum_{k=0}^{n-1} e^{-2\pi i(\mathbf{f}\cdot\mathbf{s})}, \tag{2.5}$$

where **f** is a frequency vector. The power spectrum $P(\mathbf{f})$ is derived from the spectra and periodogram [6] as given by Eq. 2.6:

$$P(\mathbf{f}) = |F(\mathbf{f})|^2 = P_r(\mathbf{f}) + P_i(\mathbf{f}), \tag{2.6}$$

$$P_r(\mathbf{f}) = (\sum_{k=0}^{n-1} \cos(2\pi\mathbf{f}\cdot\mathbf{s}))^2/n,$$

$$P_i(\mathbf{f}) = (\sum_{k=0}^{n-1} \sin(2\pi\mathbf{f}\cdot\mathbf{s}))^2/n.$$

The spectra is based on a periodogram that is the squared magnitude of the Fourier transform. This spectrum is still used, and most papers that evaluate the spectra of a point set use Ulichnev's analysis method. However, there is a drawback to this analysis method, i.e., it is not applicable to adaptive sampling. In order to evaluate adaptive sampling, Wei and Wang [121] developed an analysis method that uses a distance metric to the point density around each point.

The above analysis methods have been applied to halftoning [36, 37, 114] to ascertain the qualities of their results.

2.2 Procedural Modeling

Procedural modeling generates geometry and color from numerical parameters or contexts. Noise-based texturing functions are well-known techniques and are widely used to represent natural objects that have diverse colors and shapes such as the surfaces of bubbles and oceans [30, 91], as shown in Fig. 2.8. In addition, there are procedural modeling methods with context for modeling structural ob-

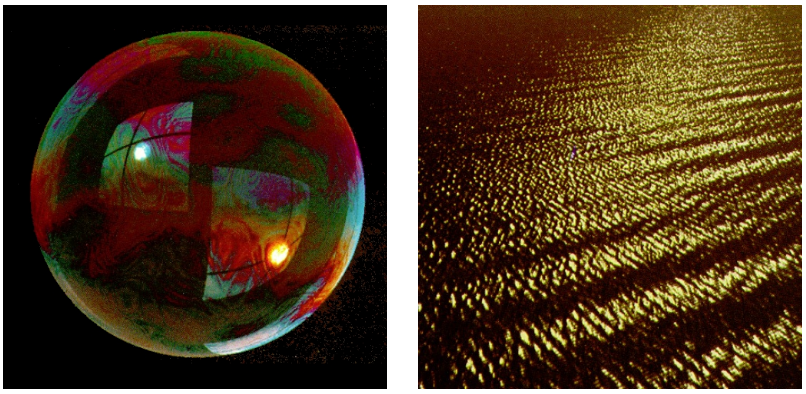

(a) Bubble. (b) Ocean.

Figure 2.8: Noise-based texturing of Perlin [91].

jects such as buildings [13, 81, 82, 122], cities [19, 90, 112], and trees [8]. However, these methods cannot generate an aggregate.

On the other hand, there are procedural modeling methods for generating an aggregate. Discrete components comprising the aggregate are frequently generated by constructing a controllable graph or curves. In other words, an algorithm consists of two phases: 1) controlling graphs or curves and 2) generating target components from them.

In general, a graph consists of nodes and links. In computer graphics, nodes have positions; therefore, links and cycles (also called circuits or polygons in graph theory) represent line segments and regions, respectively. Procedural modeling uses the links and cycles to generate discrete components. Miyata generated stone wall patterns [78]. His method generates cycles along lengths in specified ranges so that stones are piled up. Then, the method subdivides the links and provides noise in order to obtain jittered edges to the stones. Finally, 3D stone shapes are added into the graph. The result is composed of individual stones. Worley generated a graph with cells that have variations for general 2D and 3D textures. [123]. The graph is constructed of distance metric and random distributed points. Cell regions are defined by a specified distance metric from the distributed points, as shown in Fig. 2.9. Itoh et al. generated organic patterns considering anisotropic directions using an anisotropic graph [48]. The patterns are constructed by a pseudo-Voronoi diagram, which is constructed from anisotropic points. The anisotropic points are distributed by a particle system having anisotropy. Peytavie et al. stacked nonperiodic 3D stones using a

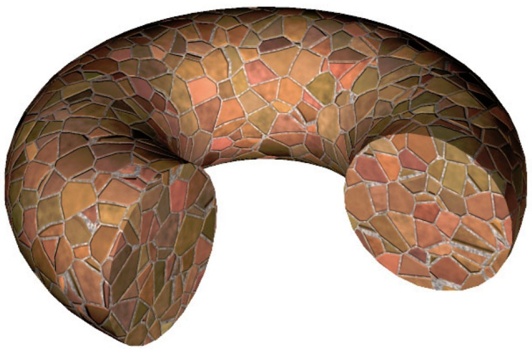
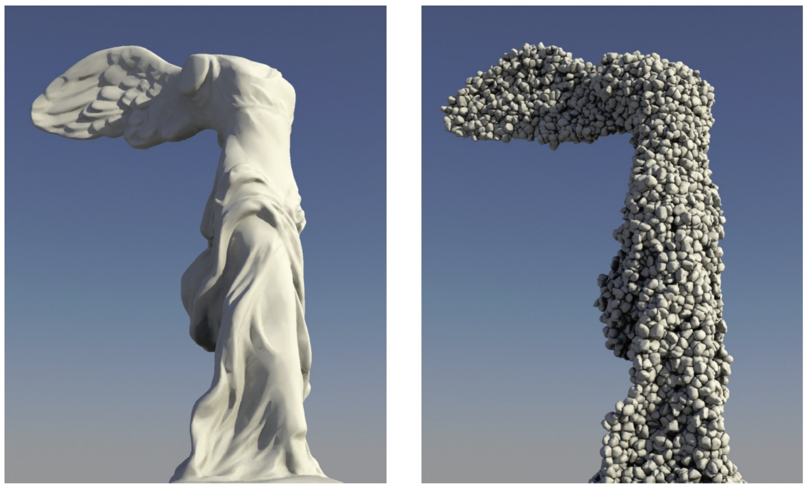


Figure 2.9: Cellular texturing of Worley [123].



(a) Specified aggregate shape. (b) Extracted stone.

Figure 2.10: Stone aggregate of Peytavie et al. [92].

Voronoi diagram [92]. To construct nonperiodic stone piles, this method uses Wang tiling of Voronoi cells. The stone shapes are simulated by erosion simulation considering the contact points between neighboring stones. The method extracts an arbitrary-shaped aggregate from generated stones, as shown in Fig. 2.10. Sakurai and Miyata represented piles of stones using a physical simulation [103]. The stones are generated by 3D Voronoi diagrams, and the sites are generated by a 3D Poisson disk distribution. The generated sites are isotropic points, and the distances between neighbors depend on the radius of the sphere for Poisson disk distribution. The size of the Voronoi cell can be controlled by changing the radius because the cell size depends on the distances between sites.

Kita and Miyata generated mosaic patterns so that placed stones are arranged along specified stream lines [57]. This method controls the orientations of stones using directions on a flow field.

Nowadays, procedural modeling methods are used for not only imagery or movies but also industrial products such as synthetic leather sheets [72, 73, 74] and wall-papers [101, 102] because procedural modeling methods rapidly generate variations of desired objects.

2.3 Nonparametric Modeling Approach

An inherent limitation of procedural modeling methods is that they generate only specific objects. To reduce this limitation, researchers have proposed nonparametric modeling in which users directly specify an intended shape. There are several approaches for nonparametric modeling such as sketch-based [109, 110], capturing shapes from real objects [7, 11, 88], editing shapes [16], mesh filters [21], and generating variations [52, 56, 111, 126, 127]. These methods do not consider aggregates composed of discrete components. On the other hand, texture synthesis methods, placement methods, and physically-based modeling methods can generate an aggregate.

2.3.1 Texture Synthesis

Example-based texture synthesis methods generate a large-scale texture from parts of small exemplars as input images. The generated large texture has a pattern similar to the exemplars. This approach lets users directly specify patterns using exemplars.

There are two approaches to texture synthesis: pixel-based and patch-based approaches [119]. The pixel-based approach considers the area around a pixel for synthesis. A texture is synthesized by finding and copying pixels that have the most similar local neighborhoods [3, 31, 32, 63, 120].

On the other hand, the patch-based approach considers appropriate regions for synthesis. To reduce the artifact of appearance, most methods using this approach extract boundaries of discrete components. Praun et al. constructed a texture from user-specified regions of texture [94]. Although this method does not automatically extract discrete components, it synthesizes the boundaries for representing discrete components. Dischler et al. automatically extracted the distribution by color quantization of input texture images to place user-specified

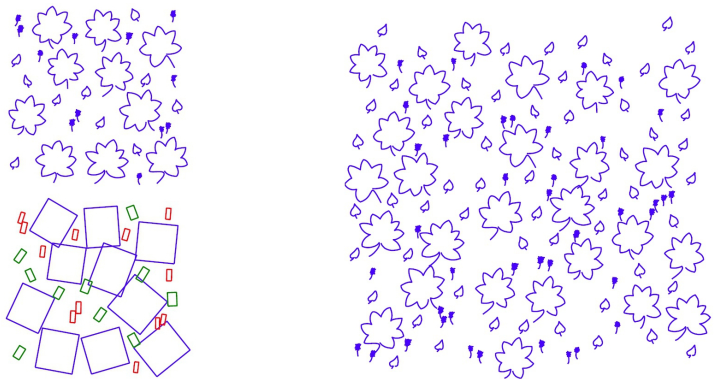
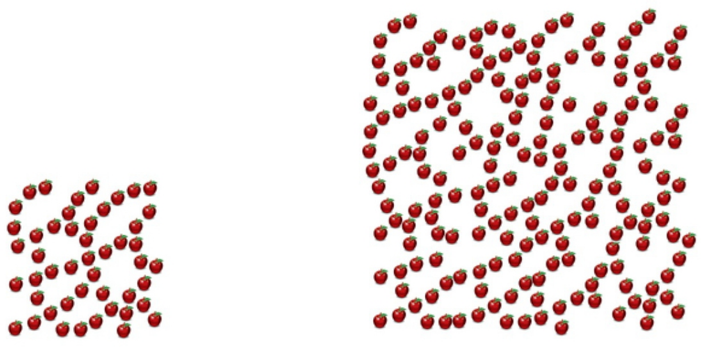

(a) Input and arrangement. (b) Synthesized texture.

Figure 2.11: Texture synthesis of Hurtut et al. [45].

discrete components [27]. This method considers the boundaries for representing discrete components. Kwatra et al. used a graph cut to automatically find boundaries from a specified texture [59]. The method considers only the boundaries of patches comprised of components, although it does not consider individual components. Wu and Yu maintained boundaries by detecting strong edges in an input texture [124]. The results of this method show smoother boundaries of components than those of a previous method [59]. Barla et al. extracted meaningful shapes of patterns to detect discrete components [5]. The target of this method is a sparse texture in which gaps exist among neighbor components. The components are classified by Hausdorff distances. To calculate the Hausdorff distance, a center and two eigenvectors of components are extracted. In generating a new texture, the components are arranged by a centroidal Voronoi diagram. Ijiri et al. arranged components from the referred component arrangements in an input texture [46]. In the arrangements, a placing component is determined from the relation of its neighboring components. By editing neighboring links, the results of the method are changed. Hurtut et al. arranged components drawn by strokes whose arrangement is similar to the specified texture [45]. The method consists of an analysis and a synthesis of 2D arrangements of the components. The analysis uses a method that classifies shapes using a histogram of the texton properties in the input texture [24]. The analysis method is called the contrario method in the computer vision field. The synthesis iteratively rearranges classified components along the probability density points of the input. The results of this method show similar textures to the inputs, as shown in Fig. 2.11. Liu et al. synthesized a



(a) Input. (b) Synthesized texture.

Figure 2.12: Texture synthesis of Ma et al. [71].

distribution and components from two different textures [69]. To extract the distribution and the components, this method also uses texton analysis. In addition, a texton is used for adding its component. A method that adds components with their textons [63] is applied to discrete components. However, these methods are not able to maintain the boundaries of individual discrete components. In order to maintain boundaries, Ma et al. generated new textures or aggregates from an input texture that included user-specified sample points [71]. The method evaluates the relative distances between sample points in the input texture. In the new texture, the relative distances of the distributed points are approximated to those of the input texture. The relative distances of the result are similar to those of input components, as shown in Fig. 2.12. However, preparing an aggregate is time consuming.

2.3.2 Placement

Placement methods that generate textures or aggregates by distributing components have been proposed. Unlike the texture synthesis methods, the aim of placement methods is to generate desired pictures or positioning and not a large texture. The placement methods use discrete components as input. This approach has frequently appeared in non-photorealistic rendering and modeling fields, particularly as representations of packing and mosaic.

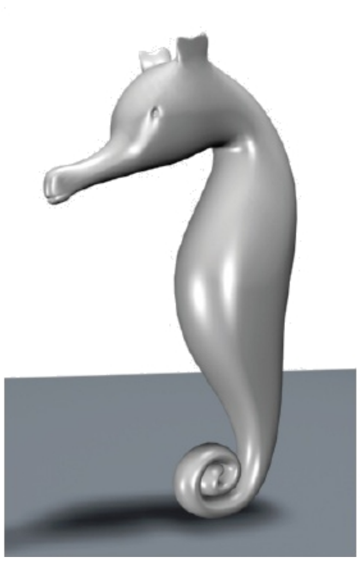
Miyata et al. proposed a method that packs a square cell into a specified region to generate a pavement [79]. The position and orientation of the square cell are determined by a bubble mesh method and a specified direction field in the specified region, respectively. As another solution, Hausner [41] used a centroidal Voronoi diagram for positioning square cells. To arrange the edges of square cells, the centroids of the cells are changed in iteration so that they avoid specified curves. Di Blasi and Gallo arranged rectangle components along offset curves as a directional guideline for determining the directions of the components [25]. In their method, to obtain a mosaic appearance, overlapped regions are removed from the components. Orchard and Kaplan [83] extended this method [25] to construct a mosaic of arbitrary-shaped components.

Kim and Pellacini proposed a method for generating a picture consisting of arbitrary-shaped components [55]. In their method, a centroidal Voronoi diagram determines the initial positions of components, and then the components are deformed to reduce gaps and overlaps among components. However, the centroidal Voronoi diagram is not suitable for generating dense aggregates composed of long components because large gaps occur in a generated aggregate composed of long components. In order to place the long components, Smith et al. proposed a method called animosaics [106] that uses an area-based Voronoi diagram [42] to determine the appropriate position and orientation of the components. Dalal et al. proposed a packing method [23] that extends a Jigsaw mosaic to reduce overlapping mosaic components [55]. This method rotates the components so that the summation of distances between component boundaries is minimized.

Gal et al. proposed a method called 3D collage for generating an aggregate composed of arbitrary-shaped components so that the components approximate the shape of the target aggregate [34]. The approximation mainly considers fitting the surface of the target aggregate with the surfaces of components, as shown in Fig. 2.13. Thus, this method arranges components and does not pile them on the surface of an aggregate.

Lagae and Dutre determined the positions of components by Poisson disk distribution [60]. By controlling the radius of the distribution, components of any size are distributed without overlapping, as shown in Fig. 2.14. However, the method is not suitable for long or concave shaped components because large gaps may occur among the placed components.

Kaplan and Salesin proposed a method for periodically tiling arbitrary-shaped components of the same type [53]. The tiling method deforms the components using the knowledge of periodic patterns [39]. According to this knowledge, any tiling component can be partitioned into precisely 93 combinatorial types. From the combinatorial types, the method parameterizes a component into 45 geometric parameterizations. Using parameterizations, the tiling method generates


(a) Specified aggregate shape.

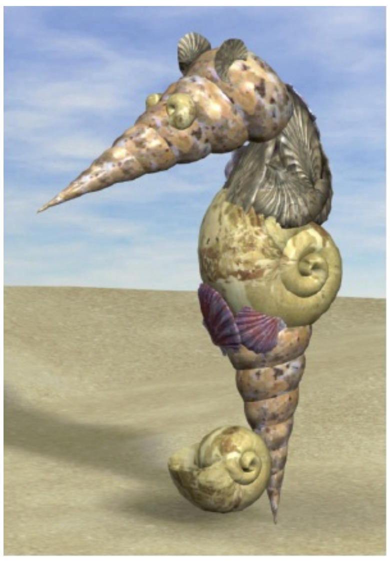

(b) Generated aggregate.

Figure 2.13: 3D collage of Gal et al. [34].

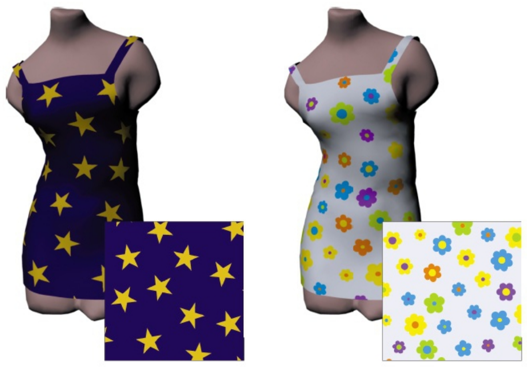


Figure 2.14: Distribution of component of Lagae and Dutre [60].

a patch of the periodic pattern. In the tiling method, components are packed without controlling their positions and orientations and the results are deterministically generated. In addition, they extended two types of components [54] in the same manner. The extended method adds parameterizations to tile two types of components.

2.4 Physically-based Modeling

Using physical simulation, some methods are capable of generating aggregates that consist of a large number of objects [66, 80]. Various methods have been proposed for easily generating the desired piles using a guide that specifies the shape of an aggregate [20, 44]. However, these methods might generate periodic arrangements using components of isotropic and near-isotropic shapes. In addition, the position of each component is unstable if conflicts among components occur, and it is difficult to determine the termination condition of the simulation. Although the simulation is appropriate in representing dynamics, it is not suitable for determining the layout of nonperiodic aggregates because it is difficult to create an intended aggregate by only specifying the physical parameters.

2.5 Discussion

In the above methods, the aims of the three methods proposed by Lagae and Dutre [60], Gal et al. [34] and Ma et al. [71] are similar those of this study. Their methods generate aggregate composed of arbitrary-shaped discrete components. However, there are differences in the targets and characteristics in these methods. Table 2.1 compares the characteristics of these methods and three methods proposed in this study. Each column indicates one method. The “2D agg.”, “3D agg.”, and “Pile” rows show the characteristics of generated aggregates. A method marked “2D agg.” and “3D agg.” rows can generate 2D and 3D aggregates, respectively. A method checked “Pile” row can generate an aggregate so that the components are piled up. A “Dens.” row presents controllability of density of aggregates. A method marked “Arb.” row lets the user to arbitrarily specify the shapes of components. In a method checked “Easy,” it is unnecessary to prepare an aggregate exemplar as an input.

The aim of the method of Lagae and Dutre [60] is to procedurally generate a 2D texture in which the components do not overlap each other. The size and

Table 2.1: Comparison of characteristics. $\surd$ indicates the available characteristics. agg., dens., and arb., are abbreviations for aggregate, density, and arbitrary components, respectively.

	2D agg.	3D agg.	Pile	Dens.	Arb.	Easy
Lagae et al. [60]	$\surd$				$\surd$	$\surd$
Gal et al. [34]		$\surd$		$\surd$	$\surd$	$\surd$
Ma et al. [71]	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	
My 2D	$\surd$			$\surd$	$\surd$	$\surd$
My 3D		$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
My Staple		$\surd$		$\surd$		$\surd$

orientation of the components can be easily manipulated. The method can be applied to real-time applications by effective Poisson disk distribution. In essence, the method is supported by a distribution method based on the Wang tile. Poisson disk distribution maintains a minimum specified distance between neighbors. Components are placed at a position so that the outline of the component is inside the circular region. Thus, a component does not touch neighboring components. In addition, the orientations of components can be changed to any direction inside their circular regions. The study by Lagae and Dutre [60] shows generated aggregates composed of discrete components; however, large gaps occur in the results because of placing long components as the circular region includes large voids. This dissertation will show these results in Chapter 3. In a centroidal Voronoi diagram, the same problem occurs because a Voronoi cell tends to shape an isotropic convex hull that resembles a circular region. To reduce the gaps, this dissertation considers generating aggregate composed of long and concave-shaped components. In addition, relaxation algorithms yield homogenous gaps [4, 42]. However, conflicts among placed components may occur.

Gal et al. proposed a 3D collage method [34] in which the surface of a user-specified target shape is approximated by those of the components. In the placement process, the position, orientation, and type of a component are determined without considering arrangement. In contrast, the proposed method generates nonperiodic arrangements and the appearance of piles without large gaps as described in Chapter 4.

Ma et al. generated a large aggregate from user-specified aggregates [71]. The problem with the method is the need to prepare a small aggregate, which requires tedious manual operation. The range of searching number for neighbors is 3^N to 7^N in N dimension. Thus, the user needs to prepare an aggregate that includes

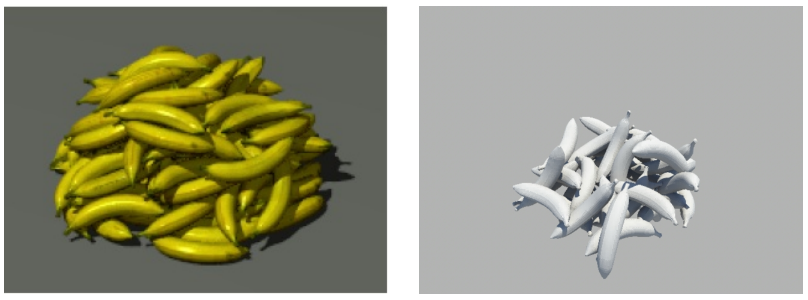

(a) Example of Ma et al. [71]. (b) Aggregate constructed by myself in 90 min.

Figure 2.15: Small aggregates for generating large aggregates.

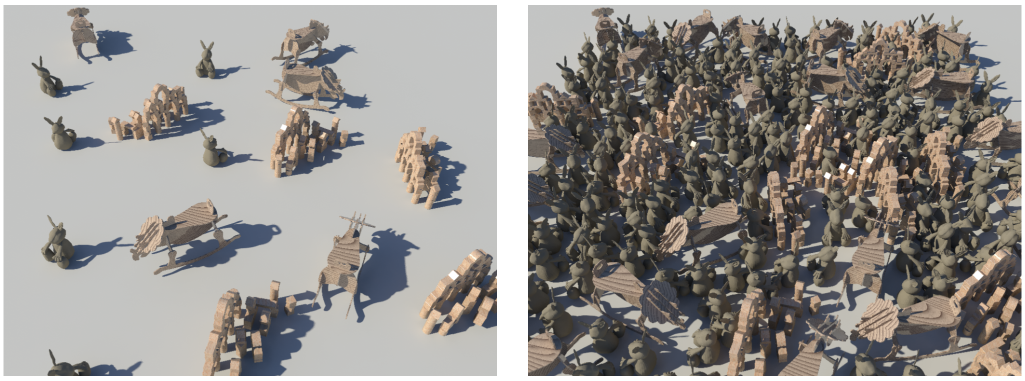
at least 3^N neighbors. However, the number of searching neighbors depends on the shapes of the components and their arrangement; therefore, it is not obvious. I attempted to create a similar exemplar in this paper, as shown in Fig. 2.15 (a). Fifty-three bananas appear in the screen space. Here assumes the aggregate as a cube, and the camera view is diagonally upward. One face has approximately 18 bananas, and one edge has approximately four bananas. Therefore, the aggregate as a whole has approximately 70 bananas.

It is too difficult to manually create an aggregate as an input for the method of Ma et al. [71]. I could not create an adequate aggregate within 90 minutes because it is too difficult to adjust the positions and orientations of components, as shown in Fig. 2.15 (b). To avoid these tedious operations, the proposed methods do not require constructing exemplars.

Chapter 3

2D Aggregate Generation Method

This chapter presents a method for modeling nonperiodic aggregates composed of arbitrary components in a two-dimensional (2D) plane. The method effectively generates a dense aggregate from the components to be arranged. In this method, exclusive regions on behalf of the components' regions are nonperiodically arranged without overlapping of the exclusive regions. The exclusive regions are inputs for this method, which nonperiodically distributes instances of the exclusive regions without overlapping. A transformation matrix is used to represent positions, orientations, and scaling of the instances. The method outputs these transformation matrices. An aggregate is generated by transforming components using these transformation matrices. The density of an aggregate depends on the shapes of the exclusive regions. To date, Poisson disk distribution has been used to generate nonperiodic aggregates; however, this approach creates large gaps among components. Dense aggregates of arbitrary components can be generated by using a dart-throwing method; however, this approach is time consuming. To improve the usability for creators, it is necessary to reduce the calculation time. This chapter presents a method that fills the gaps with components to reduce the calculation time. To effectively fill these gaps, the proposed method quantifies the gaps and finds additional positions for the components. This study showed that this method is more effective than the dart-throwing method. In addition, examples using 2D- and three-dimensional (3D) components are shown to confirm that the proposed method is generic.



(a) Previous method (16 components). (b) The proposed method (224 components).

Figure 3.1: Comparison of results.

3.1 Introduction

Nonperiodic aggregates of components are common in daily life and in natural phenomena such as toys on a table and cristae on a leather-textured surface. Aggregates are often represented by computer graphics, but creating and manually controlling the arrangement of the components are tedious. To reduce this time-consuming work, a method to generate a layout of the components is needed. Aggregates in daily life or natural phenomena have periodic (lattice) or nonperiodic arrangements. In this study, only the nonperiodic arrangement [104] is considered because it is easy to create periodic arrangements by placing components using placement rules and mathematical functions. The goal of this study is to effectively generate nonperiodic aggregates of arbitrary components without overlapping the components in a 2D plane. To achieve this goal, the proposed method utilizes to the dart-throwing method [22]that is used for Poisson disk distribution, which uniformly places points. Each point has a circular region with its center located at a point that avoids overlapping other circular regions. The dart-throwing method controls the layout of the distributed points by deforming the circular regions [65], and it is applied to randomly distribute components. To control the layout, the proposed method uses arbitrary-shaped regions called as "exclusive regions, " instead of circular regions. The positions of the components are not limited to the blue-noise properties, which are exhibited by uniformly and randomly distributed points, because the exclusive region can be specified as something other than a circle.

Several methods use Poisson disk distribution for placing components [60, 97], and they can generate various aggregates of components having isotropic shapes such as spheres or cylinders. However, these methods cannot generate dense aggregates of components that are concave or are of a variety of sizes because large gaps are produced among the components, as shown in Fig. 3.1(a). For generating a dense aggregate, two approaches are considered. In the first approach, the components are placed as it checks for component overlapping, similar to the naive dart-throwing method that uses arbitrary-shaped exclusive regions. Although this approach can generate a dense aggregate of many components, the calculation cost of the naive dart-throwing method is too expensive because of its repetitive placement trials. In the second approach, components are placed using collision and reaction in a physical simulation. This approach also generates dense aggregates, but it might also generate periodic arrangements when using isotropic shapes. To avoid periodic arrangements, the proposed method employs the first approach. The proposed method assumes that it is effective to iteratively fill the gaps among placed components to reduce redundant cancelations. In this experiment, the proposed method effectively generates an aggregate that includes only a few gaps, as shown in Fig. 3.1 (b). The experiment demonstrates that this approach is more effective than the naive dart-throwing method that iteratively places components while checking overlapping of exclusive regions.

3.2 Preliminary Experiment

As a preliminary experiment, this section attempts to determine the layout of components by using the naive dart-throwing method using arbitrary-shaped exclusive regions. This experiment confirmed the effects and drawbacks of the naive dart-throwing method.

3.2.1 Exclusive Region

The proposed method considers placing exclusive regions only in a 2D plane. The exclusive region is independent of the component shapes, and it determines the distance between a component and each of its neighbors. The position, orientation, and the size of the exclusive regions are specified by transformation matrices, and the components are attached to the exclusive regions. The exclusive regions are defined by 2D triangular meshes. With meshes, transformation calculations and overlap checking can be accelerated by the use of a graphics processing unit (GPU). A user specifies the vertices of the triangular meshes and the inner regions

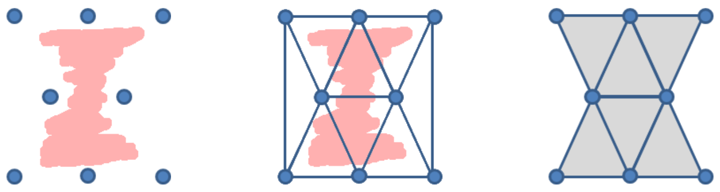


Figure 3.2: Specification of an exclusive region.

of the shapes, as shown in Fig. 3.2(a). Through a Delaunay triangulation, a triangular mesh is used to construct a convex hull from the vertices, as shown in Fig. 3.2(b). To construct the concave mesh, the triangular mesh is extracted from the inner regions, as shown in Fig. 3.2(c). Aggregate coordinates are simultaneously specified when constructing a mesh. The center of transformation is the center of the bounding box of an exclusive region. Transformation is determined by Eq. 3.1:

$$\mathbf{x} = (\mathbf{RS} + \mathbf{T})\mathbf{x}_0, \tag{3.1}$$

where $\mathbf{x}$ is the transformed position of a vertex, $\mathbf{R}$ is the rotation matrix, $\mathbf{S}$ is the scaling matrix, $\mathbf{T}$ is the translation matrix, and $\mathbf{x}_0$ is the initial position of the vertex. The transformation matrix is applied to placements of components.

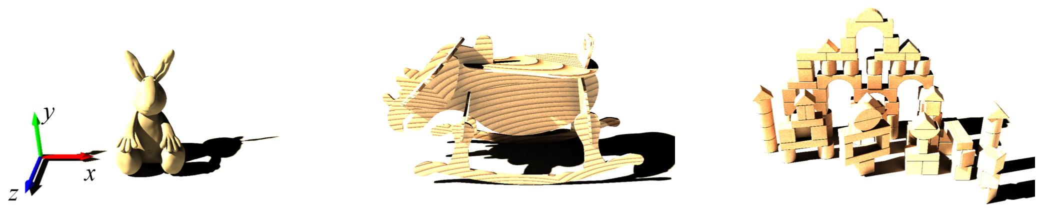
3.2.2 Procedure for Naive Dart-throwing Method

The naive dart-throwing method consists of the following steps:

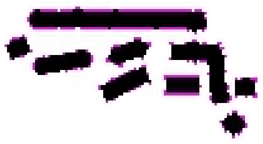
(1) Random placement of a rotated exclusive region.
(2) Collision detection of an exclusive region to be placed with already placed exclusive regions.
(3) Cancellation of placement if collisions occur.

Step (1) is repeated until the termination conditions are satisfied.

The naive dart-throwing method places a nonperiodically component. To ensure a consistent placement frequency, the exclusive region is cyclically selected from among the input exclusive regions when several exclusive regions are specified. The procedure is terminated when the number of consecutive cancelations

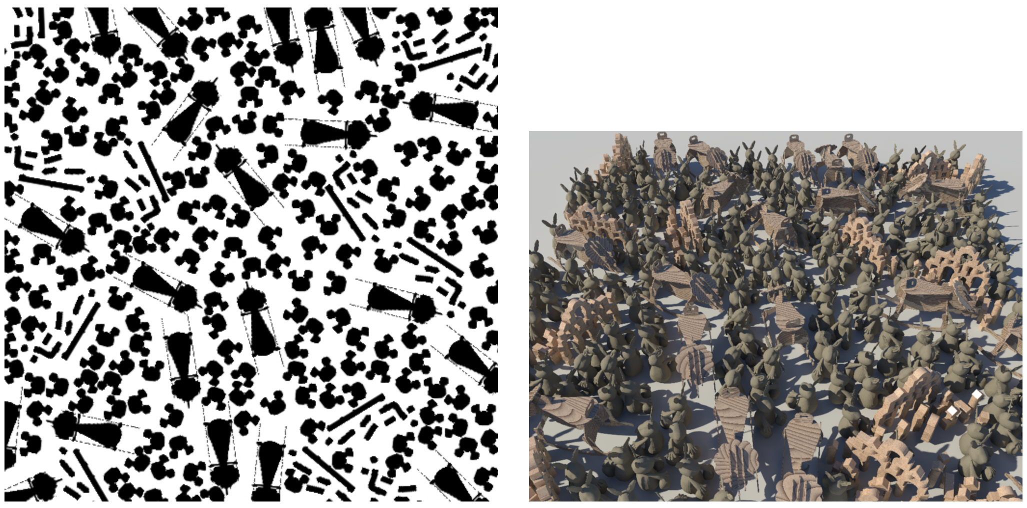


(a) E1. (b) E2. (c) E3.



(d) E1(Region). (e) E2(Region). (f) E3(Region).

Figure 3.3: Components and exclusive regions. Images (a)–(c) are components and (d)–(f) are their exclusive regions. The black region denotes an inner region, and a magenta pixel indicates a vertex in (d)–(f).



(a) Placed exclusive regions. (b) Placed components.

Figure 3.4: Result of the naive dart-throwing method (210 components).

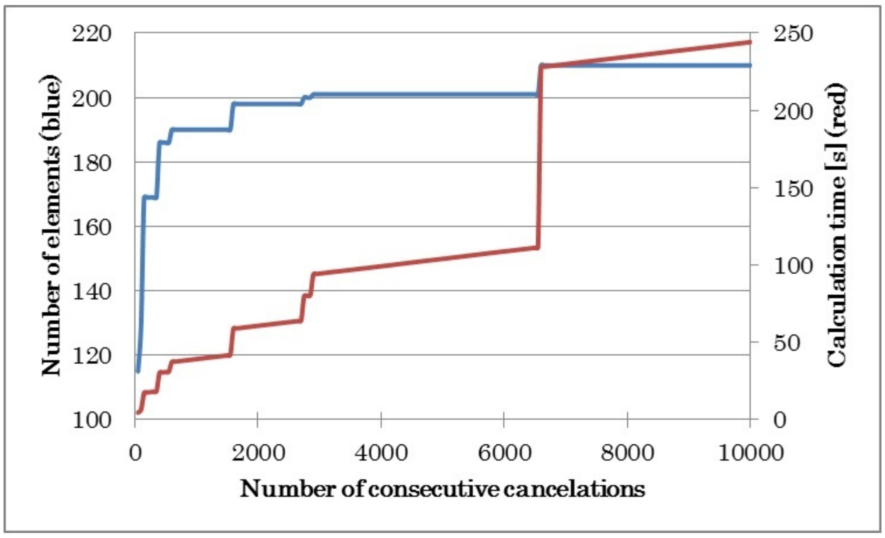
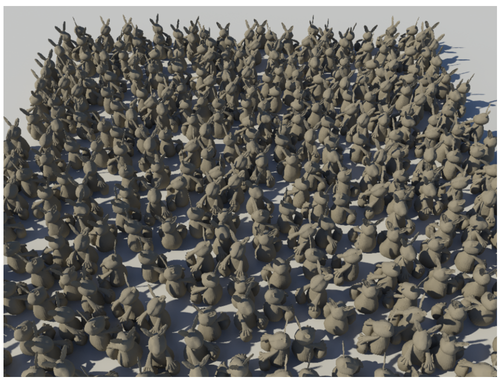


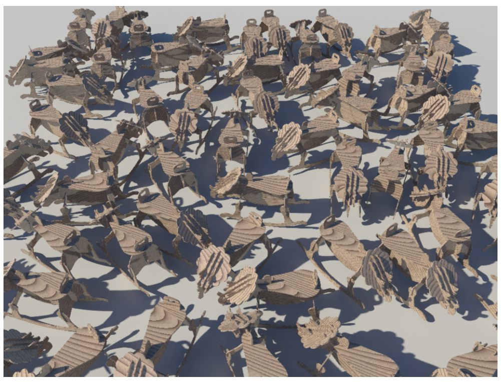
Figure 3.5: Relationship between number of consecutive cancelations, number of components, and calculation time of Fig. 3.3.

exceeds a user-specified number. In the dart-throwing method for Poisson disk distribution, the number of placed points can be estimated from the sizes of the placed circles [62]. Therefore, the termination condition is easily determined. In contrast, the number of placed exclusive regions is uncertain in the dart-throwing method using arbitrary-shaped exclusive regions. The number of placed exclusive regions depends not only on the size but also on the shape of the exclusive regions. However, the placement tends to be consecutively cancelled when a specific region in a 2D plane is filled with exclusive regions. The proposed method uses the number of consecutive cancelations as the termination condition.

3.2.3 Results of Preliminary Experiment

The naive dart-throwing method was implemented on a Windows PC with an Intel Core i7, 3.07 GHz CPU, and a 12.0 GB RAM. For this experiment, components E1, E2, and E3 are used, as shown in Figs. 3.3 (a)–(c). The exclusive regions are created by projection onto the XZ-plane in 3D as shown in Figs. 3.3 (d)–(e). Figure 3.4 shows the result of the naive dart-throwing method with termination condition specified as 10,000 consecutive cancelations. Here demonstrates the generation of an aggregate without scaling. Figure 3.4 (a) shows the placed exclusive regions, and their transformation matrixes are applied to the components as shown in Fig. 3.4 (b). The dart-throwing method generates


(a) E1.


(b) E2.

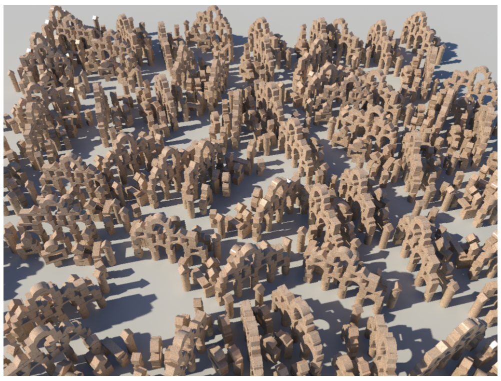
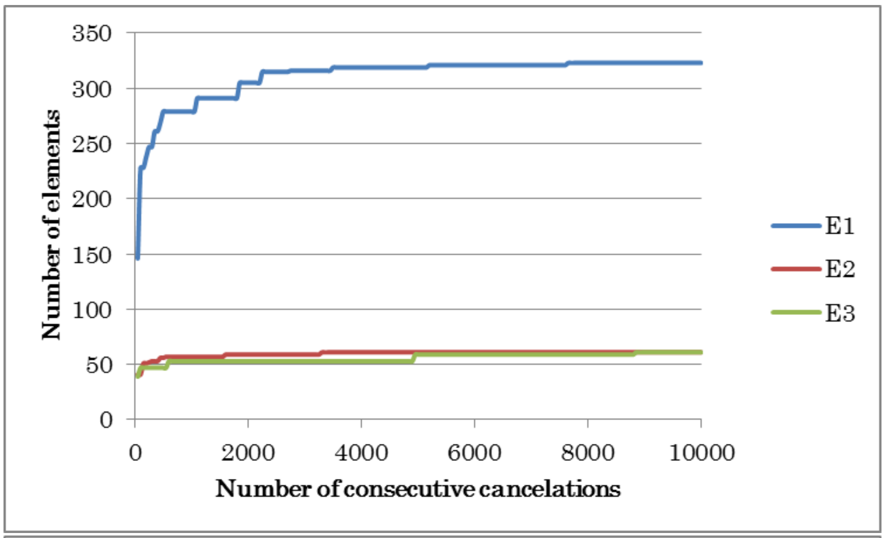

(c) E3.

Figure 3.6: Results of each of three types of components.



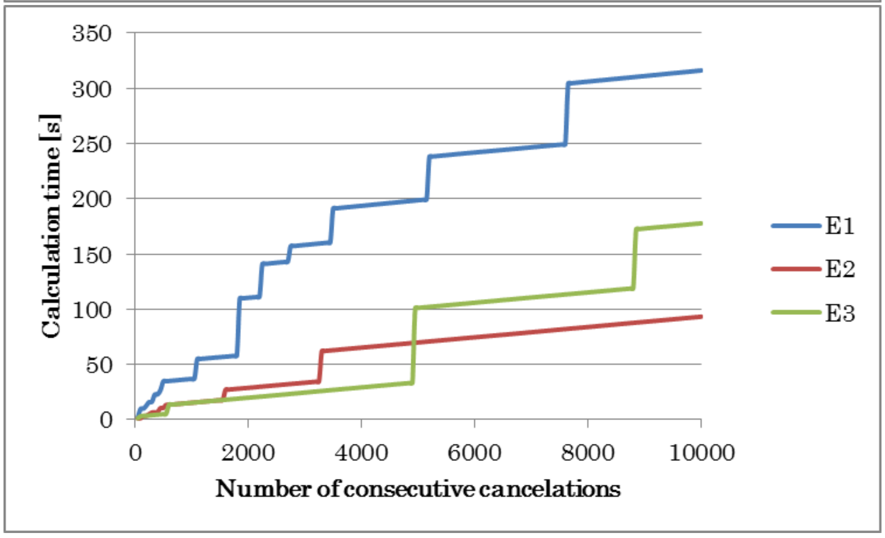
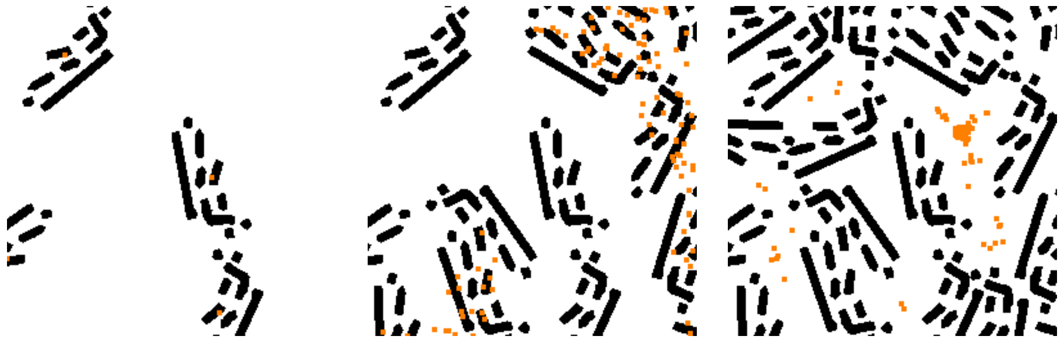


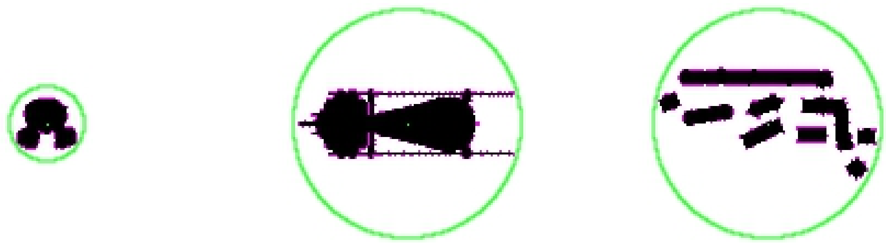
Figure 3.7: Relationship between number of consecutive cancelations, number of components, and calculation time of Fig. 3.6.



(a) Step 1. (b) Step 2 (one iteration). (c) Step 2 (termination).

Figure 3.8: Overview of the proposed method.

nonperiodic dense aggregates without overlapping of the components on the XZ-plane. Figure 3.5 shows the number of placed components and the calculation time of every termination condition for generating Fig. 3.4. The graphs are plotted for each of the 50 consecutive cancelations. Overall, a positive correlation appears in the graph. The calculation time (red line) is very short until 100 consecutive cancelations. The number of components (blue line) dramatically increases until approximately 2,000 consecutive cancelations, and then the increase becomes gentle. In addition, for approximately 7,000 consecutive cancelations, the calculation time dramatically increases while only nine components are added. The calculation time of the naive dart-throwing method depends on the position and orientation of the components, which are randomly determined. The method is not robust, and the calculation cost is high. As another example, Fig. 3.6 shows the results of distributing components E1 through E3. Figure 3.7 confirms that the calculation costs are in the same range as in Fig. 3.5. To reduce calculation time, the proposed method refers to the dart-throwing method to produce uniform points. There are two approaches to place points: parallelization [14, 117, 125] and specification of regions [28, 85]. Parallelization approaches are only used for points having blue-noise properties. These methods cannot be used because nonperiodic arrangements of the proposed method do not have blue-noise properties. The specification approaches, on the other hand, reduce the number of placement cancelations.



(a) E1. (b) E2. (c) E3.

Figure 3.9: Circular region O. The green circle indicates the circular region's boundary for each component.

3.3 Fill-Gap Method

To effectively generate an aggregate, the proposed method iteratively attempts to place several exclusive regions into gaps among already placed exclusive regions. This method assumes that having many placement cancelations increase the calculation cost. Thus, to reduce this cost, this method prepare a set of candidate points P where the exclusive region can be placed. In the preliminary experiment (Section 3.2.3), the calculation time was shortened by setting the termination condition to less than 100 consecutive cancelations. From this result, the proposed method specified the number of P as 100, thereby limiting the number of consecutive cancelations to 100. To avoid overlapping of exclusive regions as far as possible, instances of P are preferentially distributed in the large gaps among the placed exclusive regions. All processes use exclusive regions to determine the position, orientation, and scaling of components, as described in Section 3.2.1. The proposed method consists of two steps:

(1) Placement of exclusive regions sparsely
(2) Placement of additional regions in the gaps

Figure 3.8 shows the two steps: the black regions indicate exclusive regions, and the orange points indicate instances of P. Step 1 is an initial placement of exclusive regions, which ensures that there is no overlapping of exclusive regions by Poisson disk distribution. However, some gaps among placed exclusive regions are usually generated. Step 2 iteratively fills the gaps by using positions P. In the

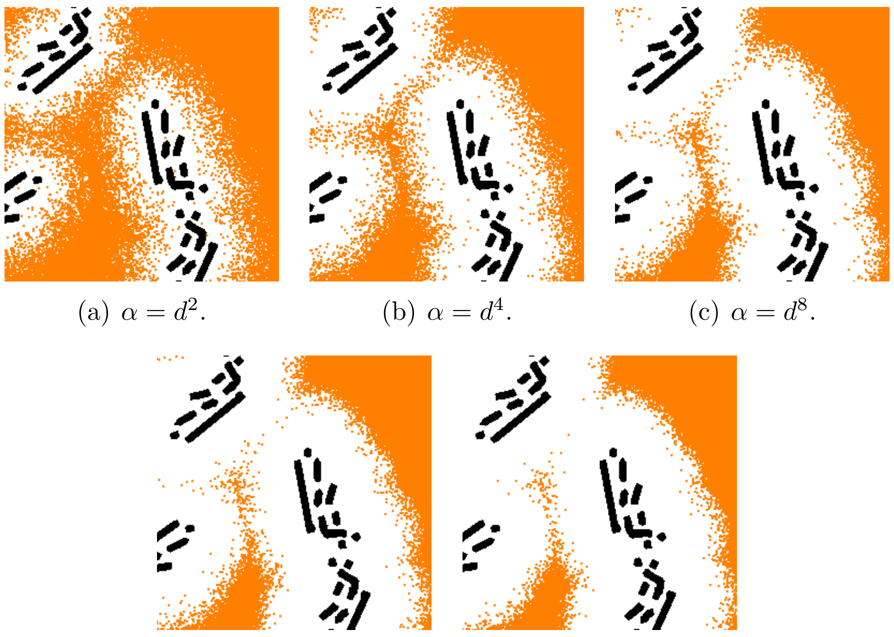
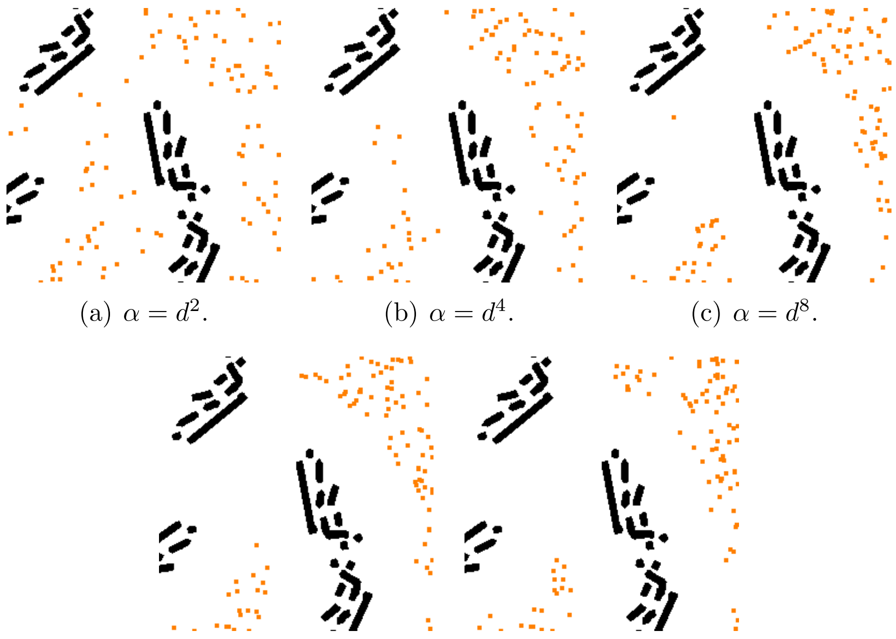

(a) $\alpha = d^2$. (b) $\alpha = d^4$. (c) $\alpha = d^8$. (d) $\alpha = d^{16}$. (e) $\alpha = d^{32}$.

Figure 3.10: Comparisons of yielded points Q (orange).

step, the cancelations of placements frequently occur when P and the boundaries of placed exclusive regions are too close. In this experiment, the proposed method specifies the termination condition to be 200 consecutive placement cancelations.

3.3.1 Sparse Placement

Step 1 sparsely places exclusive regions at the points distributed by the dart-throwing method for Poisson disk distribution. The Poisson disk distribution ensures that circular regions do not overlap. By specifying a circular region O involving an exclusive region, nonperiodic arrangements can be quickly calculated with no overlapping of exclusive regions. The value of $\mathbf{T}$ is determined by the position of a placed point, and the values of $\mathbf{R}$ and $\mathbf{S}$ refer to a random angle and scale in a specified range. The dart-throwing method uses an exclusive circular region O for point distribution, where the center of O is the placed point. Here the largest circle is specified for Poisson disk distribution; thus, the radius of O is determined by the distance between the center and the farthest point on the boundary as shown in Fig. 3.9.



(a) $\alpha = d^2$. (b) $\alpha = d^4$. (c) $\alpha = d^8$.

(d) $\alpha = d^{16}$. (e) $\alpha = d^{32}$.

Figure 3.11: Comparisons of points P (orange) chosen from Fig. 3.10.

3.3.2 Placement in Gaps

In the preliminary experiment (Section 3.2.3), the calculation time is shortened by setting the termination condition to less than 100 consecutive cancelations. By limiting the number of consecutive cancelations to 100, the proposed method assumes that the calculation time for placement will be shortened. The following two steps place instances of P: the first step places candidate points Q of P by means of GPU acceleration. The second step chooses 100 instances of P from Q to place exclusive regions for reducing the calculation time. To avoid concentrating Q at a few positions, Q are scattered across wide areas. The abundance of Q is in proportion to the distance from the placed exclusive regions. Here GPU acceleration is applied to quickly calculate distances by using an image plane as a distance field. The probability α of a pixel in the probability field is defined by d^n, where d is the value of a pixel in a normalized distance field. With a normalized distance field, the method can place exclusive regions of any size in the same manner. In addition, by using d^n, a higher probability is given to farther positions and a lower probability to nearer positions. Therefore, a larger or smaller number of points are placed at farther and nearer positions. The proposed method determines whether a point is placed by comparing the probability with a random number [0, 1]. When the random number is over the probability α for each pixel, a point q in Q is placed. Figure 3.10 shows the comparisons of points of different exponential variable n. The result shows that the 16th power yields acceptable results. The points Q placed between the 16th and 32nd powers are similarly distributed. The proposed method deduces that using the 16th power of normalized distances for the probability α is sufficient. The next step is to choose 100 points P from the placed points Q. Figure 3.11 shows comparisons of points P taken from points Q for each probability α. Points P chosen from $\alpha = d^{16}$ and $\alpha = d^{32}$ are similarly distributed in positions farther from the boundaries of placed exclusive regions. The results show that it is sufficient to use the 16th power for probability α.

3.4 Result

This section discusses the results of the proposed method. Sections 3.4.1 and 3.4.2 show the calculation times of the proposed method and examples of changing exclusive regions. Section 3.4.3 shows variations of the generated aggregates.

3.4.1 Calculation Time

The proposed method was implemented on a Windows PC with an Intel Core i7, 3.07 GHz CPU, and 12.0 GB RAM. Figure 3.1(b), which shows an aggregate having three types of components, was calculated in 18.3 s and includes 224 components. Figure 3.12 shows placed exclusive regions. The naive dart-throwing method needs approximately 250 s to generate an aggregate of 210 components as shown in Fig. 3.3. A comparison of the number of components placed by the proposed method and by the naive dart-throwing method reveals that the proposed method placed 14 more components, an increase of around 6.7%, which shows it can place a sufficient number of components. The other results exhibit the same trend as the above, as shown in Fig. 3.13. Table 3.1 shows a comparison of calculation times between the proposed method and the naive dart-throwing method. Figure 3.13 (d) consists of two types of trees with scaling ranging from 0.2 to 1.0.

Table 3.1: Comparison of calculation time between the proposed method and the naive dart-throwing method

As shown in Fig. 3.13 image	The proposed method		Naive method	
	Time (s)	No. of components	Time (s)	No. of components
(a)	17.2	321	316.2	323
(b)	11.5	64	93.3	61
(c)	13.2	58	177.8	61
(d)	31.5	244	488.0	258

3.4.2 Changing Exclusive Region

The proposed method can control the densities of generated aggregates by changing the size of the exclusive regions. Figure 3.14 shows controlled densities of aggregates accomplished by changing the exclusive regions. Smaller or larger exclusive regions create denser or sparser aggregates, respectively.

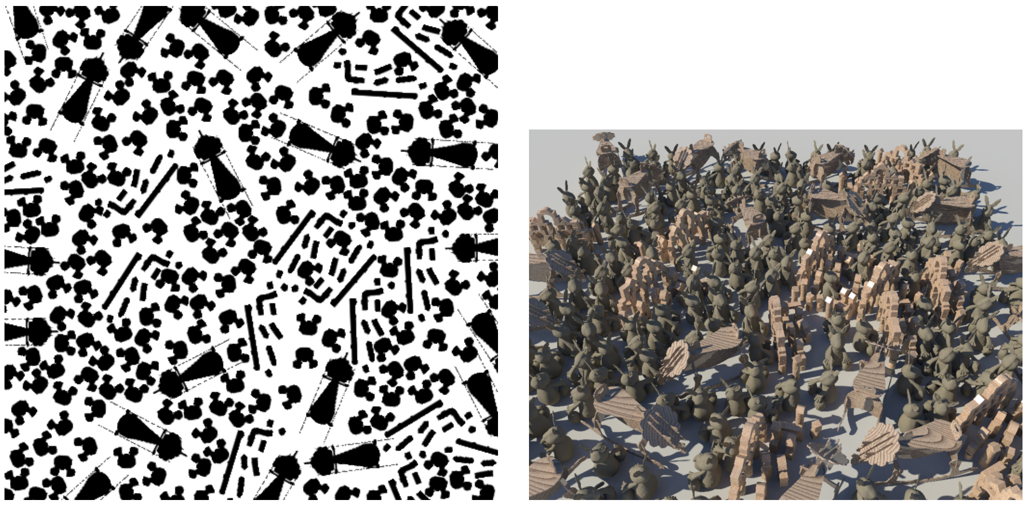
3.4.3 Example

The proposed method places arbitrary components not only in 2D but also in 3D. Figure 3.15 shows examples of placing 2D and 3D components. Traditional Japanese textures are expressed by placing 2D components. Examples of a 3D-component distribution express disarranged objects on a floor. The method can also be used to generate height maps. Figures 3.16–3.18 show height maps of a leather texture composed of cristae and sulci. The sulci are placed at the same positions as the cristae. This shows that the proposed method can be generically used for generating aggregates.

3.5 Evaluation

I interviewed two professional groups of creators. Concerning aggregates of 3D components as shown in Figs. 3.12–3.13, they said: "These results look like natural aggregates. We often use a crowd of people in a still image. The method can be applied to represent a crowd of people." This response suggests that this method may be sufficient for generating 2D aggregates.

Commenting on the variations of densities, as shown in Fig. 3.14, they said,



(a) Placed exclusive regions. (b) Placed components.

Figure 3.12: Exclusive regions placed by the proposed method.

“There are many plant types and environments, for example soils and the amount of solar radiation. These affect the densities of plants, and various densities of plants exist in a scene.” To reflect these commons, the user interface should be able to interactively edit exclusive regions and areas to be placed for each component.

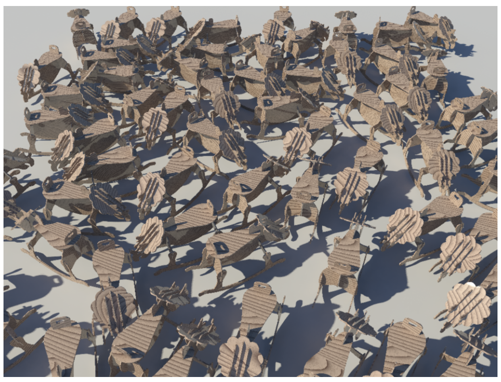
As for the height fields, as shown in Figs. 3.16–3.18, they said, “It is a highly effective method for texturing. It can generate the hide texture of any animal.” They clearly appreciated the applicability of this method as a texturing method.

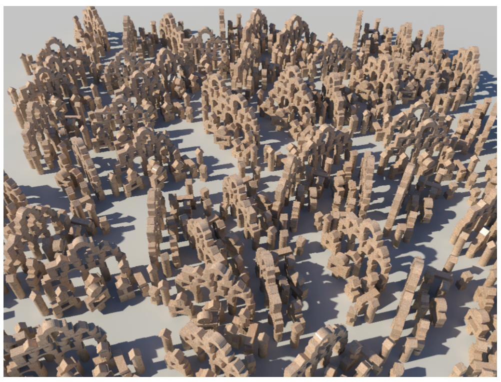
They concluded that the method makes it easy to control the densities of aggregates by directly editing exclusive regions, and the results are of a sufficient high quality for practical applications.

3.6 Discussion and Conclusion

This chapter proposed a procedure for determining the layout of the nonperiodic aggregates of arbitrary components. The proposed method was implemented on the basis of dart-throwing method using exclusive regions that prohibit each component from overlapping another. The method was used to fill the gaps among placed exclusive regions. The method is also dramatically faster than the naive dart-throwing method. The chapter has illustrated a few representative examples of nonperiodic textures in the figures. The proposed method enhances

(a) E1.


(b) E2.


(c) E3.

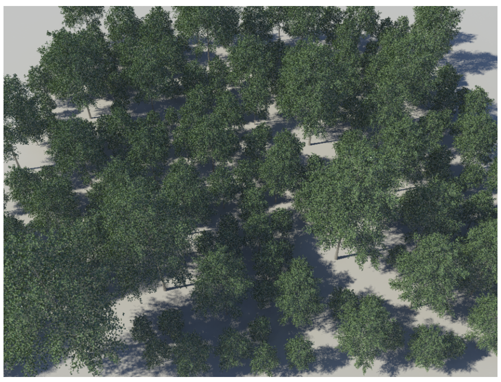
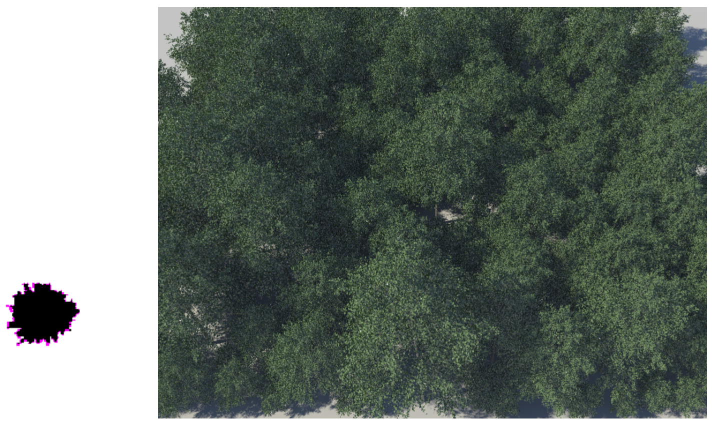
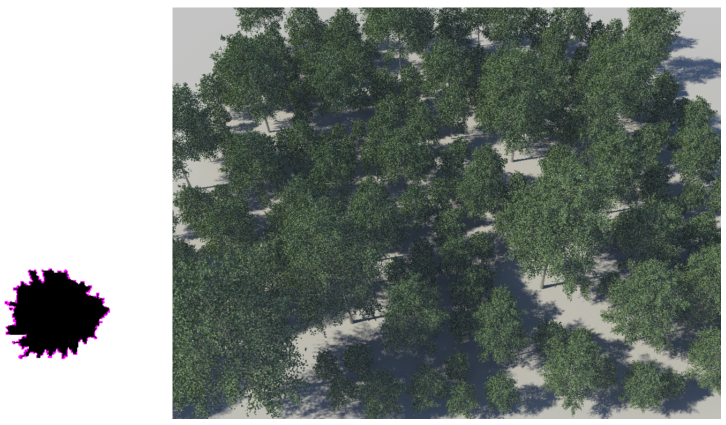

(d) Tree.

Figure 3.13: Generated aggregates.


(a) Dense.


(b) Middle.

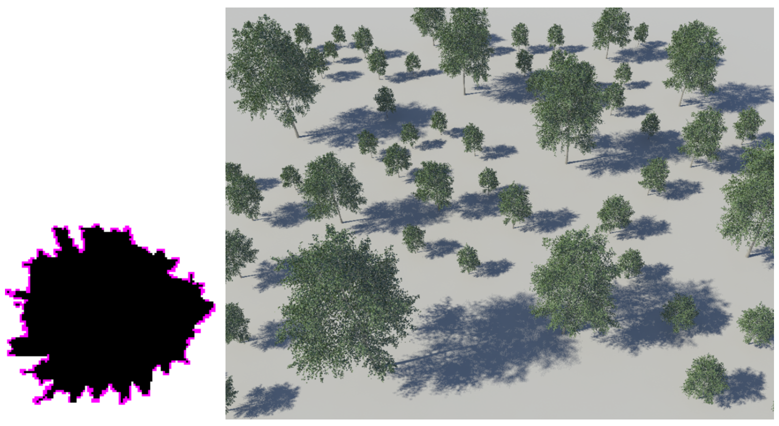

(c) Sparse.

Figure 3.14: Variations of densities.

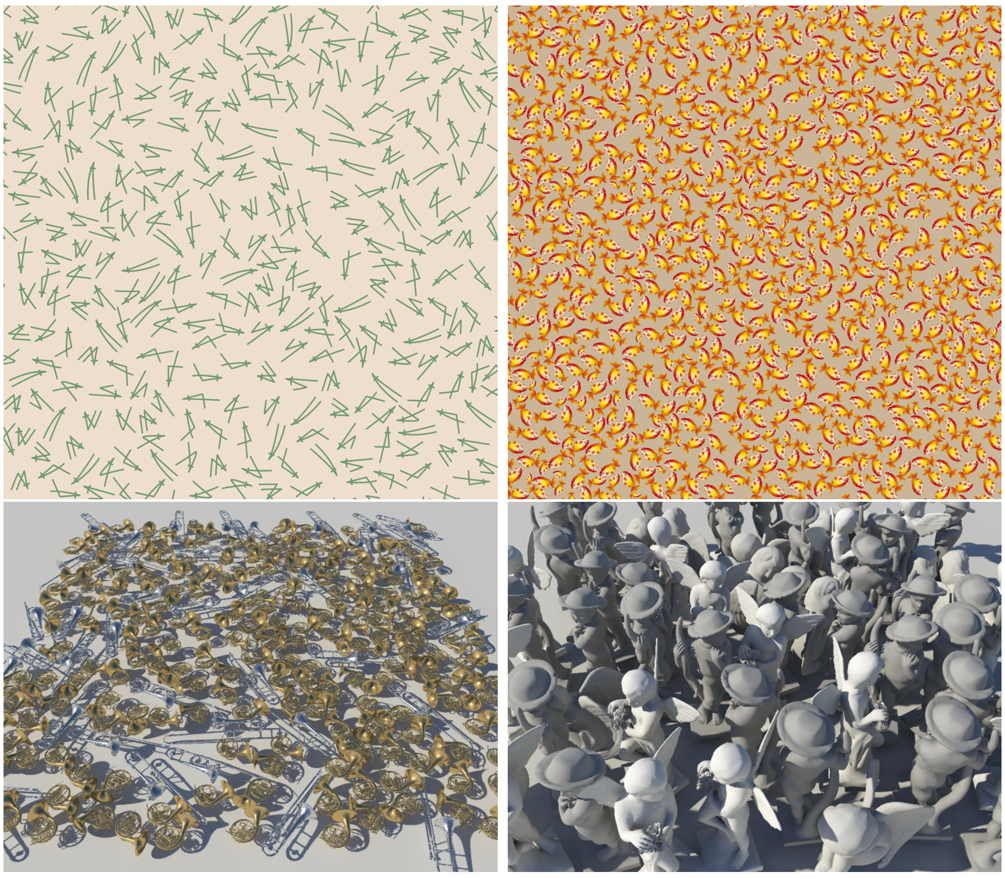
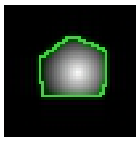
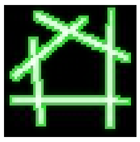
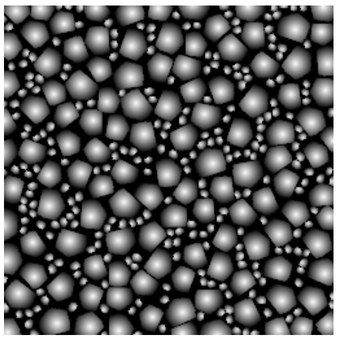
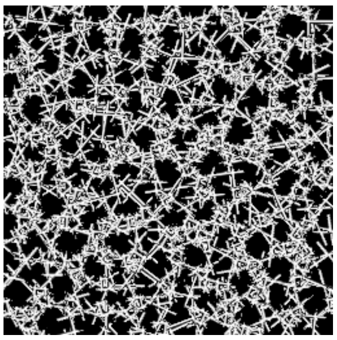


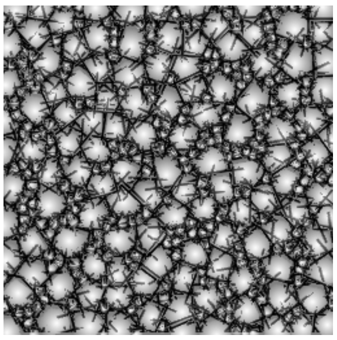
Figure 3.15: Variations of examples.


(a) Cristae comp.


(b) Sulci comp.


(c) Cristae.


(d) Sulci.


(e) Synthesis.

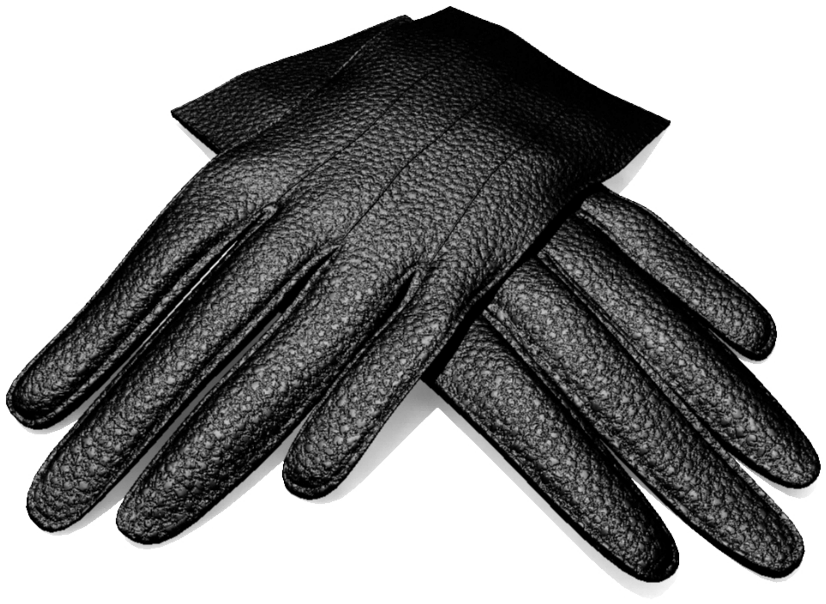
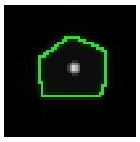
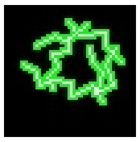
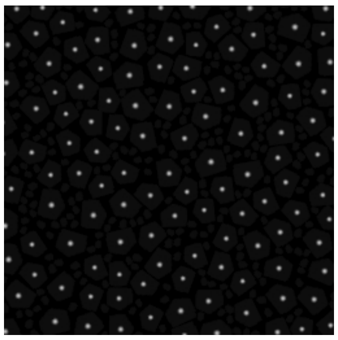
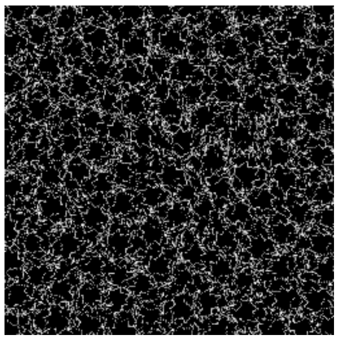

(f) Rendering image.

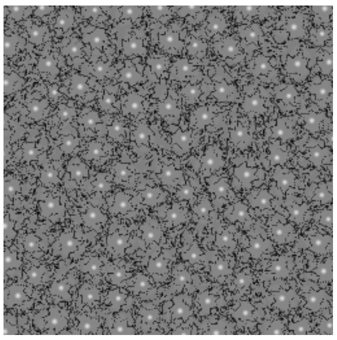
Figure 3.16: Generated height maps of leather texture # 1.


(a) Cristae comp.


(b) Sulci comp.


(c) Cristae.


(d) Sulci.


(e) Synthesis.

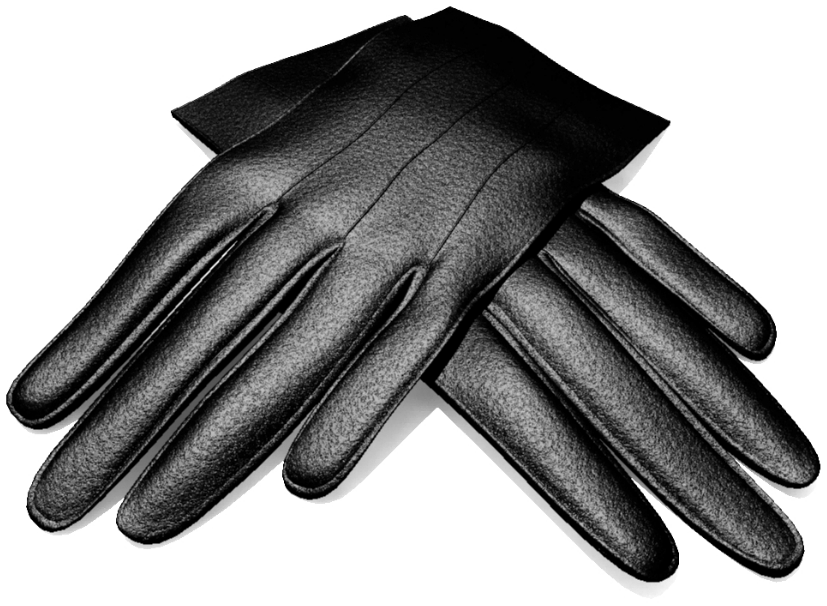
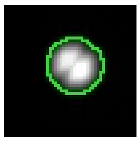
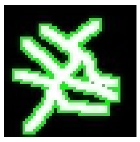
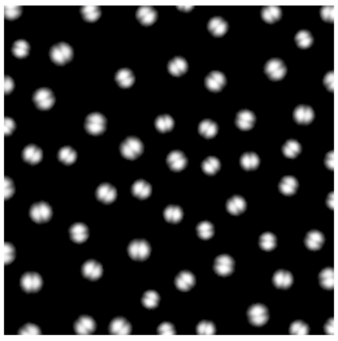
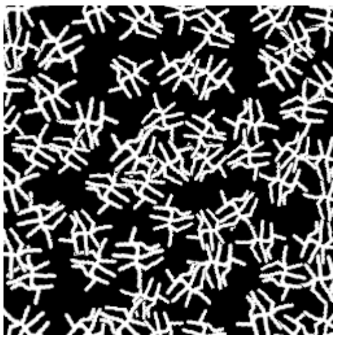

(f) Rendering image.

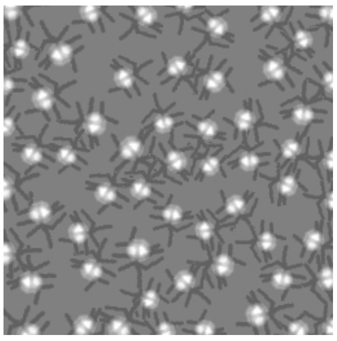
Figure 3.17: Generated height maps of leather texture # 2.


(a) Cristae comp.


(b) Sulci comp.


(c) Cristae.


(d) Sulci.


(e) Synthesis.

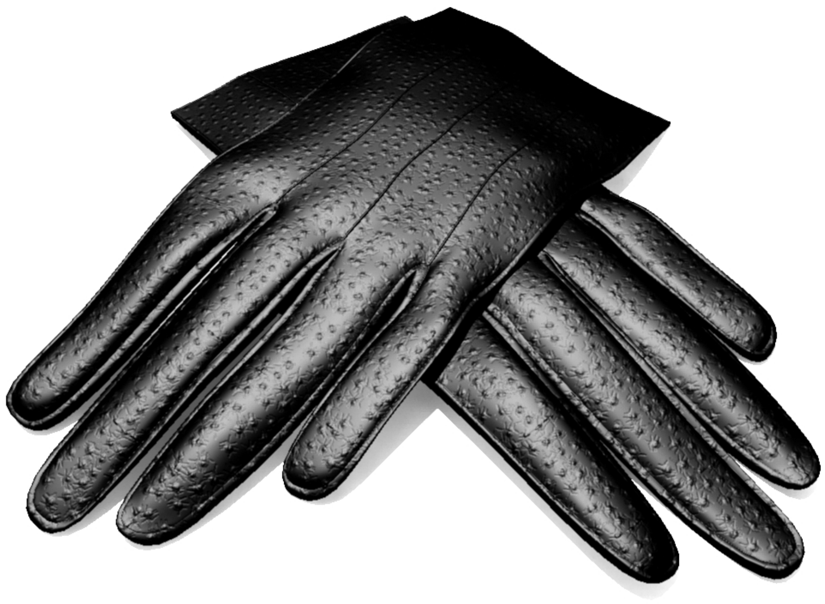

(f) Rendering image.

Figure 3.18: Generated height maps of leather textures # 3.

the interactivity for distributing components by accelerating the calculation.

The method has advantages and disadvantages compared with the related method [71] for generating aggregates. The related method synthesizes an aggregate from an exemplar. The distribution of the components of an aggregate is similar to that of the exemplar; however, the distribution can be periodic. An exemplar is not used in this method, and only nonperiodic aggregates are generated. This method is effective for a user who intends to obtain nonperiodic aggregates without specifying their distribution. However, it cannot generate periodic aggregates even if exclusive regions are given.

Spectrum analysis for distributed points is used to justify a point distribution method in general. However, these analysis methods are not applicable for the results of the proposed method. I will consider an adequate method for analyzing the distribution of arbitrary-shaped components in the future. The method currently does not apply to practical design tools. To achieve this, I would like to consider functions editing for exclusive regions and for areas in which the components will be placed.

Chapter 4

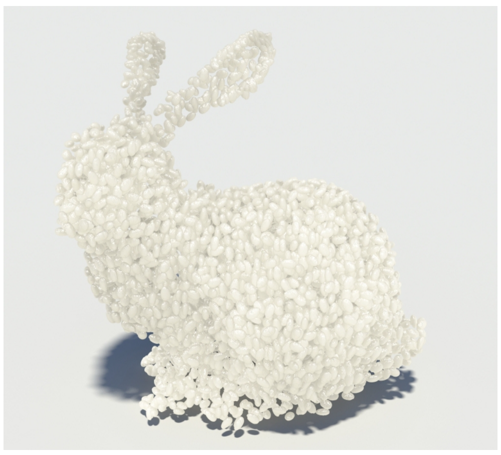
3D Aggregate Generation Method

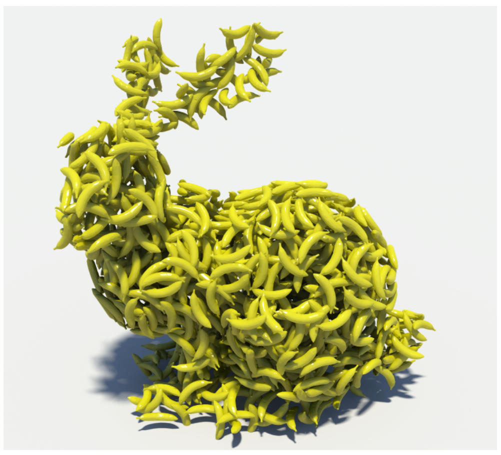
This chapter presents a procedure for modeling aggregates such as piles that consist of arbitrary components. The method generates an aggregate of components that need to be accumulated, and an aggregate shape represents the surface of the target aggregate. The number of components and their positions and orientations are controlled by five parameters. The components, the aggregate shape, and the parameters are the inputs for the method, which involves placement and refinement steps. In the placement step, the orientation and initial position of a component are determined by a nonperiodic placement such that each component overlaps its neighbors. In the refinement step, to construct a pile structure, the position of each component is adjusted by reducing the overlap. Finally, the method outputs a number of components and the positions and orientations of all the components to construct an aggregate.

4.1 Introduction

Representation of surface details enhances the quality of computer-generated imagery. Numerous methods have been proposed for generating surfaces, such as noise-based texturing functions, texture generation, and displacement mapping [30]. However, it is difficult to generate aggregates such as heaps of steamed rice and fruit by these methods. In addition, representation of piles generally requires many tedious and time-consuming manual operations. This dissertation proposes a procedural modeling method that generates aggregates and free

(a) Steamed rice.


(b) Bunny-shaped rice.


(c) Bunny-shaped bananas.

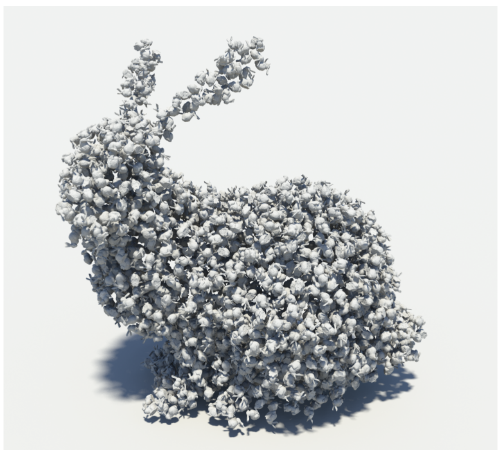

(d) Bunny-shaped bunnies.

Figure 4.1: Generated aggregates.

shapes that consist of arbitrary components, as shown in Fig. 4.1. To achieve a natural appearance, the components are placed nonperiodically in the generated aggregate.

Recently, Ma et al. proposed a method for synthesizing repetitive components from the input of small patch exemplars and large output domains [71]. The method preserves the properties of the individual components and their relative distances. However, it is exceedingly difficult to achieve a natural appearance by manual aggregate modeling even if the aggregate is a small patch because, in the abovementioned modeling, the position and orientation of each component need to be adjusted to reduce the gaps among the components. In contrast, an approach proposed in this chapter does not require the construction of exemplars to generate aggregates.

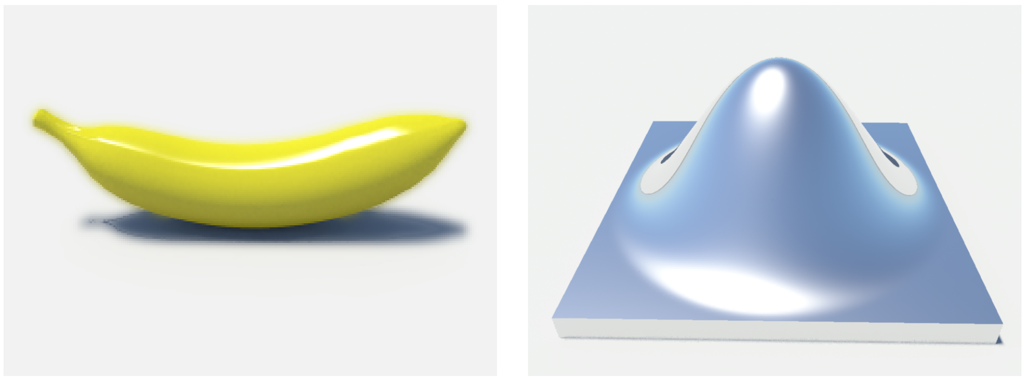
An aggregate comprises a large number of arbitrary components that include invisible components (inner components). To reduce the data volume of components, the proposed method assumes that it is sufficient to generate an aggregate only around the surface of the target shape. In addition, inner components are sometimes required to generate an aggregate, for example, a solid texture that represents inner components, such as a basket of fruits. By using conical meshes that have planar faces and possess offset meshes [67, 93], the proposed method can generate an aggregate that includes inner components.

4.2 Overview of Aggregate Modeling

This section is an overview of proposed procedure. The details are described in Section 4.3.

The proposed method generates aggregates containing pile structures that are constructed of arbitrary components. Here, a pile structure is defined as a state in which objects are placed one upon another. To construct an aggregate, it is necessary to determine the number of components, their orientation, and the position of each component. The proposed method includes a placement step and a refinement step, which are used to place components into an aggregate. The method does not use any iteration process for determining the orientations of the components to reduce the computational cost.

Figure 4.2 outlines this method. Two types of input data are first prepared: components and aggregate shapes, as shown in Figs. 4.2 (a) and (b), respectively. The placement step distributes the components on the aggregate shapes such that all components overlap their neighbors, as shown in Fig. 4.2 (c). The number of


(a) Component (banana). (b) Aggregate shape.

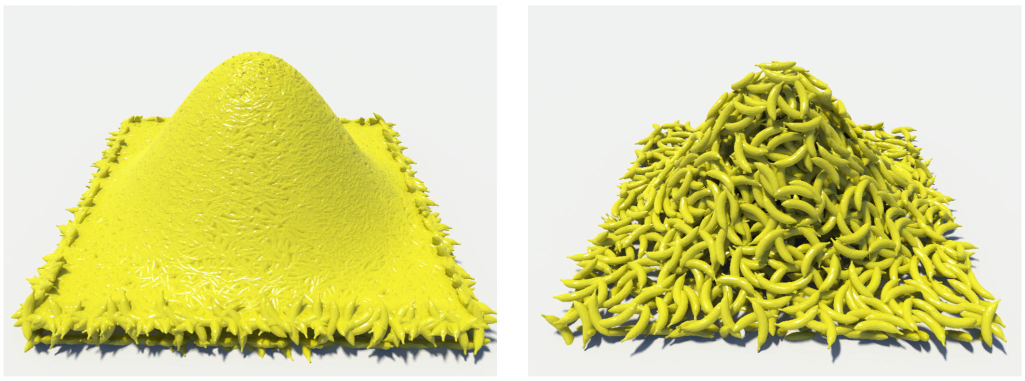

(c) After placement. (d) After refinement.

Figure 4.2: Process overview.

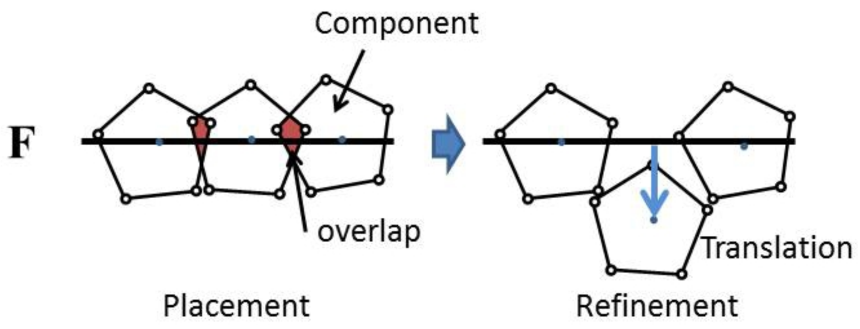


Figure 4.3: Constructing a pile structure from overlapped components in the refinement step. **F** indicates a part of the aggregate shape.

components and the orientation of each component are fixed in this step. Next, the refinement step sets the position of each component. This step reduces the overlap of each component by translation along the normal of the aggregate shape to construct a pile structure, as shown in Fig. 4.3. The pile structure is obtained by reducing the overlaps among the component, as shown in Fig.4.2 (d).

The proposed method does not consider any physical simulations, because it is difficult to create an intended aggregate solely by specifying the physical parameters. This method focuses on generating various aggregate models.

4.3 Aggregate Modeling Procedure

The proposed method specifies the aggregate shapes and components using a 3D polygon mesh. Here components are defined by closed meshes to quantify the overlapping domains among components.

4.3.1 Component Placement

This subsection describes a method for placement of single and multiple types of components.

4.3.1.1 Overview of Placement

This placement step has three requirements (R1), (R2), and (R3). In R1, a uniform arrangement of a component is required to avoid the holes that are included in the aggregates. In R2, the generated aggregates are constructed with nonperiodic arrangements to avoid an artificial appearance. The procedure uses

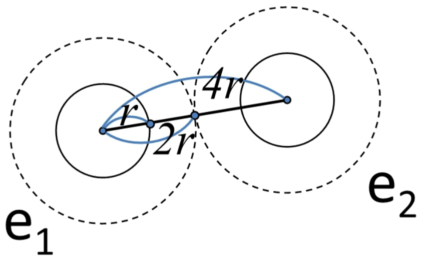


Figure 4.4: Two spheres in contact with radii $2r$ (dotted line).

a dart-throwing method to distribute a uniform, nonperiodic point set in the specified regions [22] to satisfy these two requirements. Here 3D components are arranged such that each component is located at a distributed point. To form a 3D shape, the dart-throwing method is used, in which the Euclidean distance is used as the measure of the distance between the points. In R3, to obtain a pile structure in the refinement step, it is necessary to place components such that they overlap their neighbors. Here exploits the distribution property of the dart-throwing method in which the distance between a point and its neighbors depends on the user-specified distance r, which is known as the distribution radius. At least one point should be located within the range of the distance between each point $[2r, 4r]$ [28]. This indicates that all distances between a point and its neighbors should be less than $4r$. Figure 4.4, in which two spheres with radii of $2r$ are in contact, shows the maximum distance between two points in the dart-throwing method. When the distance is less than $4r$, these spheres overlap. In the proposed method, a component is placed at the distributed point obtained by the dart-throwing method. In this example, the component must overlap its neighbors if a sphere with a radius of $2r$ is considered as the largest inner sphere of a component, as shown in Fig. 4.5.

4.3.1.2 Distribution Radius

The distribution radius, r, is defined as half of the radius of the largest inner sphere of each type of component to guarantee sufficient overlap of the components. A medial axis is used to construct the largest inner sphere. The medial axis consists of the centers of the medial balls, which are maximally empty balls of the closed surfaces [2]. This method chose the largest inner sphere from a set of maximally empty balls calculated using Amenta's method, which uses a Voronoi diagram [2].

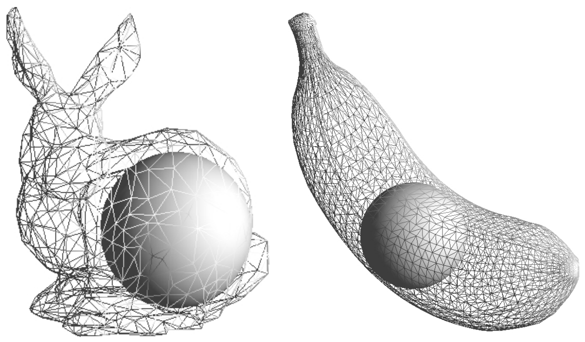


Figure 4.5: Largest inner sphere.

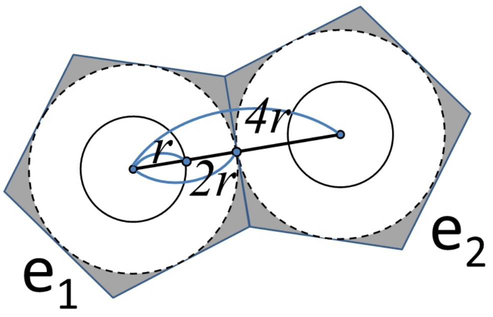


Figure 4.6: Two components in contact. e_2's center is located at a distance of $4r$ from e_1's point.

Figure 4.6 illustrates two pentagonal components, e_1 and e_2, of the same size, which have been put in place. In this case, these components share an edge. If the distance is less than $4r$, then these components overlap each other.

This method can control the density of a component by applying the parameter C_{radius}, which can change the distribution radius using the formula $r' = rC_{radius}$. The components are found to be tightly or loosely packed by using the condition C_{radius} <1 or C_{radius} >1, respectively.

4.3.1.3 Implementation of Dart-throwing Method

This method used a parallel dart-throwing method [125] accelerated by a graphics processing unit (GPU). This method selects points on a surface with geodesic distances; in contrast, the implementation of the proposed method uses the Euclidean distance to place the 3D points. The geodesic distance measures the length of the shortest path on the surfaces. The geodesic distance between two points on a curved surface is longer than the Euclidean distance. The number of points distributed on the surface by using the geodesic distance is higher than that using the Euclidean distance, especially on a highly curved surface. Therefore, to avoid the generation of excessive points, the method uses the Euclidean distance in the parallel dart-throwing method. The parallel dart-throwing method needs to input a high-density point set P by shape distribution [84]; however, the required density is unknown, and it needs to be established by maintaining the distance among the points at a value less than r. The proposed method calculates P by emplying a Catmull–Clark subdivision surface, which can be applied to any polygon [18]. The subdivision surface enables easy control of the density of vertices by adjusting the number of recursions. The Catmull–Clark subdivision surface is advantageous because it can be applied to all types of polygons, whereas the other subdivision methods cannot be applied to all types of polygons. The Catmull–Clark subdivision is iterated until the value of the edge length is less than r, and the obtained vertices are used as P.

4.3.1.4 Orientation

The orientation of a component is determined by rotation. The rotation center of a component is defined as the center of the largest inner sphere of the component to maintain the overlapping of components, as described in Sub–subsection 4.3.1.2. To determine the orientation of each component, the local uvw-coordinate system is used, the v-axis of which is oriented along the surface normal. To produce different orientations, such as randomly or neatly directed orientations, the direction of a component is perturbed from the surface normal within a specified angle range in the local coordinate system. Each component is randomly rotated within user-specified ranges in the local coordinate system. Parameters u_r, v_r, and w_r denote the range of rotation angles around the u-, v-, and w-axes, respectively.

4.3.1.5 Placement of Multiple Types of Components

To construct aggregates with multiple types of components, aggregates for each type of component are generated in the placement step and then merged to form an aggregate. This step discretely chooses each component to merge components, as shown in Fig. 4.7. Figures 4.7 (a) and (b) show the results of discretely selecting large and small components, respectively. The small component is removed, as shown in Fig. 4.7 (b), if the center of the small component is included within the distribution radius of a large component. The small components are placed in

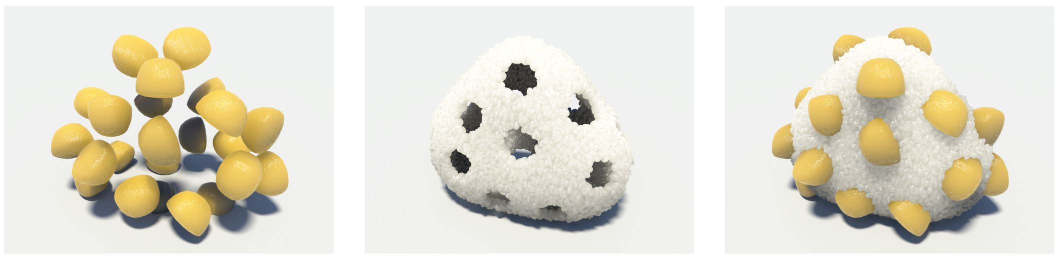

(a) Aggregation of larger components. (b) Aggregation of smaller components. (c) Final aggregation.

Figure 4.7: Placement of multiple types of components. Chestnuts and rice in a rice-ball shape.

the gaps among the large components, which are generated by loosely distributing the components, as shown in Fig. 4.7 (a). The loosely packed aggregation in Fig. 4.7 (a) is produced by setting a suitable value for the parameter C_{radius}. Figure 4.7 (c) shows the result of aggregating compound components of different sizes.

4.3.2 Refinement

At this point, components whose orientations were already determined are placed on an aggregate shape surface, and all the components overlap their neighbors. In the refinement step, each component is translated along the surface normal, $\mathbf{n}$, to construct a pile structure so that the number of overlapping components is reduced.

Wei's experiments in Poisson disk distribution [117] indicate that sampling of a scan-line order makes similar bias artifacts to the scan line, and sampling of random order tends to nonperiodic arrangement. Therefore, the refinement process is separately applied for each component in a random order to maintain a nonperiodic arrangement.

4.3.2.1 Quantification of Overlap

To reduce the overlap, the proposed method determines the minimum distance, t, of the translation along the normal within the range $[0, rC_{refine}]$, which is obtained by minimizing a cost function $c(t, T)$ and is given by Eq. 4.1. The refinement range is described in Sub–subsection 4.3.2.2.

$$c(t,T) = a(\max_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}+t\mathbf{n},T)) + t \tag{4.1}$$
$$a(h) = \begin{cases} h + rC_{refine} & \text{if } h > 0 \\ 0 & \text{otherwise (no overlap)} \end{cases}$$

Here $\mathbf{X}$ is the set of all vertices $\mathbf{x}$ of a component, and $f(\mathbf{x}, T)$ is an implicit function obtained by Eq. 4.2. The first term of the cost function calculates the maximum value of the overlaps, and the penalty value, rC_{refine}, is added to avoid overlaps.

The method quantifies the overlaps of the components by implicit modeling [12]. Implicit modeling is an effective method for calculating the depth of a 3D model. It represents the inside of a model by using the medial axis with sphere functions [2]. The method computes the implicit functions using the obtained medial axis. The implicit function $f(\mathbf{x}, T)$ is given by Eq. 4.2. It returns a positive value when $\mathbf{x}$ lies inside the model.

$$f(\mathbf{x},T) = \max_{g\in\mathbf{G}} g(\mathbf{x}) - T \tag{4.2}$$
$$g(\mathbf{x}) = (r_f - ||\mathbf{x}-\mathbf{p}_f||)s(\mathbf{x},\mathbf{p}_f,r_f)$$
$$s(\mathbf{x},\mathbf{p}_f,r_f) = \begin{cases} 1 & \text{if} ||\mathbf{x}-\mathbf{p}_f|| < r_f \\ 0 & \text{otherwise} \end{cases}$$

Here T is an offset value, and $\mathbf{G}$ is a set of $g(\mathbf{x})$ that indicates a function of a sphere whose center and radius are $\mathbf{p}_f$ and r_f, respectively. Figure 4.8 shows color mapping based on the function values. The colors blue, green, and red indicate low, middle, and high values, respectively. In this case, a deeper position from the surface has a higher value.

4.3.2.2 Minimization

The function $c(t, T)$ in Eq. 4.1 is minimized by a golden section search method [95]. In this example, 15 iterations are required to obtain a certain tolerance for a golden section search. The refinement range is specified by the coefficient, C_{refine}. The range does not intersect the surfaces of an aggregate shape to maintain the aggregate shape. The golden section search method calculates the distance

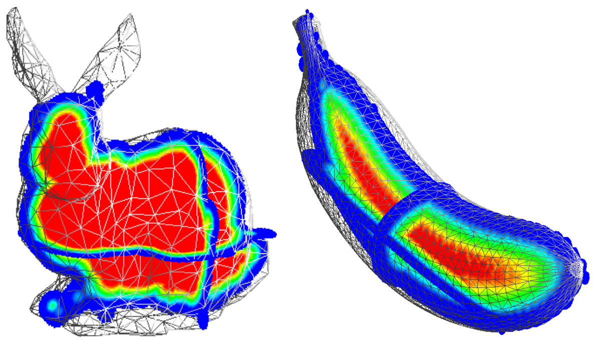


Figure 4.8: Color mapping based on the function values.

travelled by implicit functions in returning the minimal value within the range $[0, rC_{refine}]$. In this case, the minimum value is the local minimum value. In general, the global minimum value might be smaller than the local minimum value. However, it is too expensive to minimize the N-dimensional function by the downhill simplex method [95]. In the proposed method, dimension N represents the number of components, and it is not important to calculate the global minimum value because it results in the smallest summation of all overlaps. Therefore, local minimization is sufficient to meet the requirements.

4.3.2.3 Removal of Components

After the refinement process, a component may collide with other components on a surface with a high curvature even though the value of the cost function given by Eq. 4.1 is low. To avoid this, the method removes the component to maintain space if the condition given by Eq. 4.3 is satisfied.

$$c(t, T) < D \tag{4.3}$$

Here D is a threshold parameter given by the user. When D is set to zero, each component touches others on an isosurface, which is specified by an offset value, T.

4.3.2.4 Multiple Types of Components

A refinement range is specified for each type of component by changing the value, C_{refine}. Figure 4.9 shows a comparison of the results with and without specifying the strict refinement ranges. Figure 4.9 (a) shows the result when the value of C_{refine} for larger components is small. Large components are dominant among the

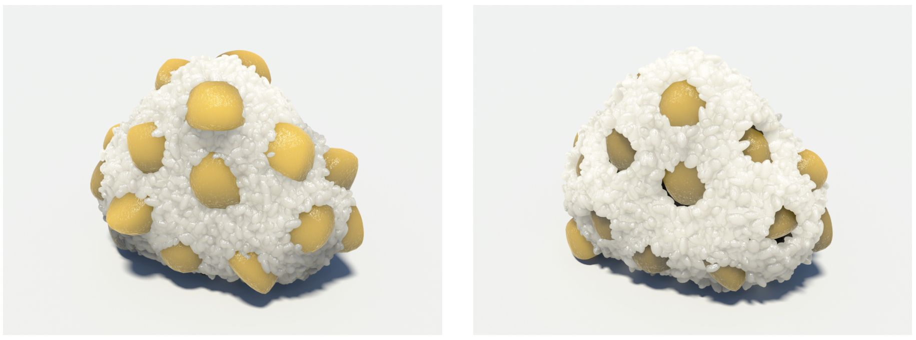

(a) When a strict range is specified. (b) When a strict range is not specified.

Figure 4.9: Refinement of multiple types of components.

number of overlaps and are embedded in the generated aggregate by specifying a strict range for C_{refine}. In contrast, large components stick out when a strict range is not specified, as shown in Fig. 4.9 (b).

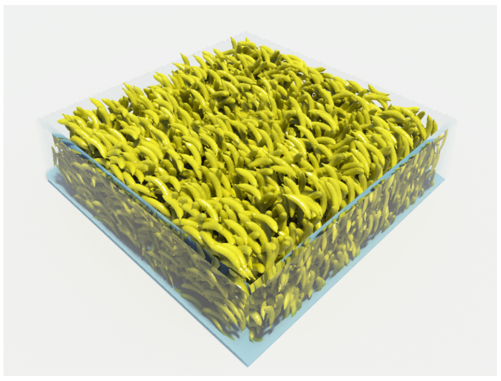
4.4 Results

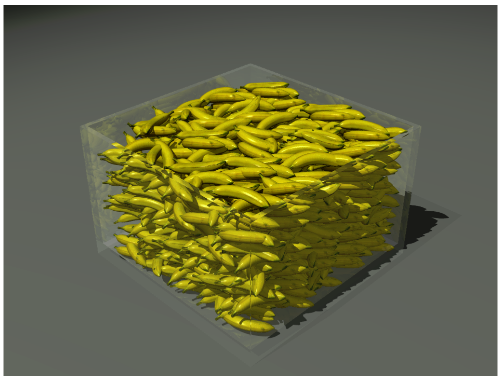
This section discusses the results of the proposed method, which was implemented on a personal computer running Windows 7, Intel Core i7 950(3.07 GHz) CPU, NVIDIA GeForce GTX 570, and 12.0 GB RAM using C++ with CGAL and OpenCL.

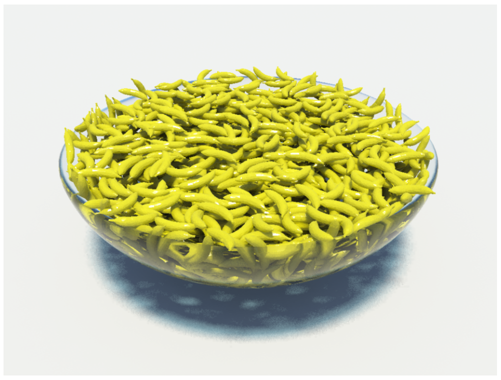
Figure 4.10 compares the proposed method with that in a previous study [71]. The proposed method generates an image only from one component model and one aggregate shape. In contrast, the previous method needs an exemplar-type structure. The proposed method can also control the orientation of components by using rotation parameters. The calculation times are approximately 3 and 4 min for obtaining the results during the comparison of the generated aggregates, as shown in Figs. 4.10 (a), (c), and (e).

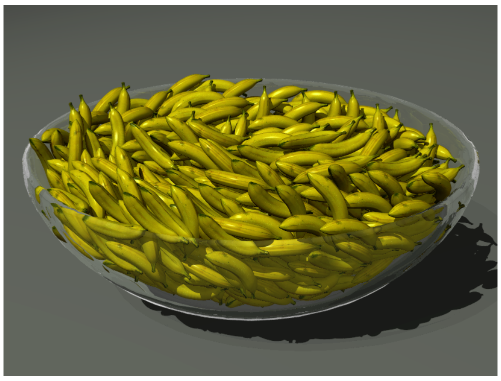
The proposed method can handle multiple components of various sizes, as shown in Fig. 4.11. These results are easily obtained because the proposed method does not need to construct exemplars. If these results were generated by texture synthesis approaches [71], exemplars composed of a large number of components would be necessary.

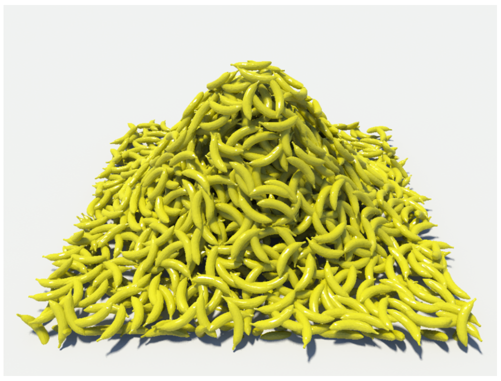
Figure 4.12 shows the difference by changing the offset parameter, T. The proposed method controls the collisions among objects by T. By setting any positive value to T, components collide with other components, as shown in Fig.


(a) The proposed method.


(b) Previous method [71].


(c) The proposed method.


(d) Previous method [71].


(e) The proposed method.

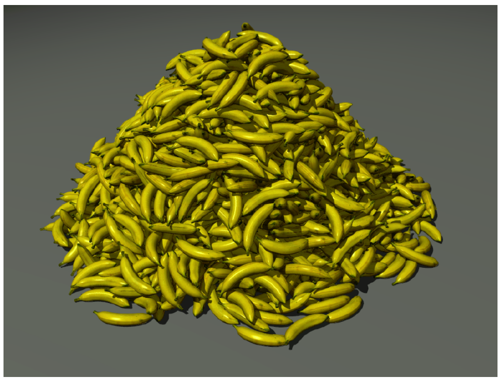
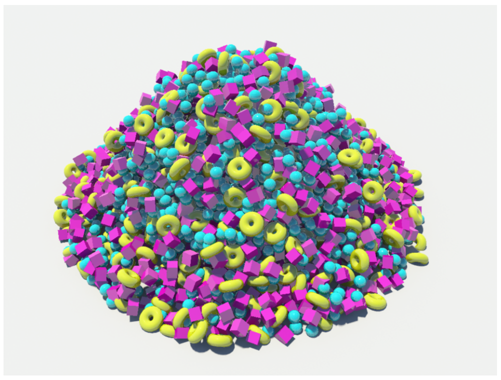

(f) Previous method [71].

Figure 4.10: Comparison of generated aggregates.


(a) Aggregate of three types of components: tori, balls, and boxes.

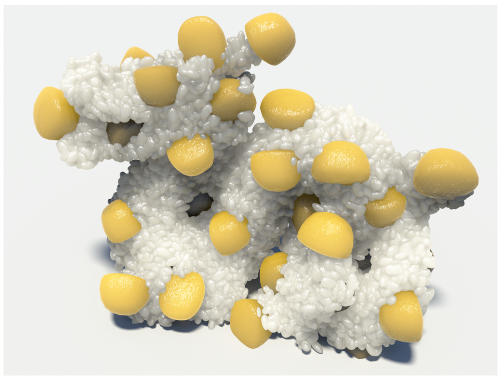

(b) Dragon-shaped chestnut-rice ball.

Figure 4.11: Aggregates from multiple types of components.

4.12 (b).

The proposed method determines the orientation of a component. Figure 4.13 shows the results generated by changing the ranging parameters u_r, v_r, and w_r. Figure 4.13 (a) shows an input component. The proposed method orients a component such that the v-axis lies along the surface normal. Without rotation, all components are oriented toward the normal of the aggregate shape, as shown in Fig. 4.13 (b).

The orientations of components can be controlled in the global coordinate system. In general, an aggregate composed of nonadhesive components is piled up so that the number of gaps in an aggregate is small. Ordinary, anisotropic gaps tend to be yielded in an aggregate of anisotropic-shaped components. Therefore, nonadhesive anisotropic-shaped components tend to be piled so that their orientations are toward similar directions in the piled aggregate. Figure 4.14 shows that components are oriented toward similar directions by specifying small ranges of rotations in the global coordinate system. The appearance of the figure looks like components that have been piled up.

An aggregate is generated when the components are placed on the surface of an aggregate shape. Using conical meshes of an aggregate shape, the proposed method generates a solid texture that is composed of discrete components, as shown in Fig. 4.15. The conical meshes are manually created using an offset curve function in a 3D modeling tool.

Figure 4.16 shows variations in the results. The proposed method can specify properties, including size and shape, of components and the shape of the aggre-

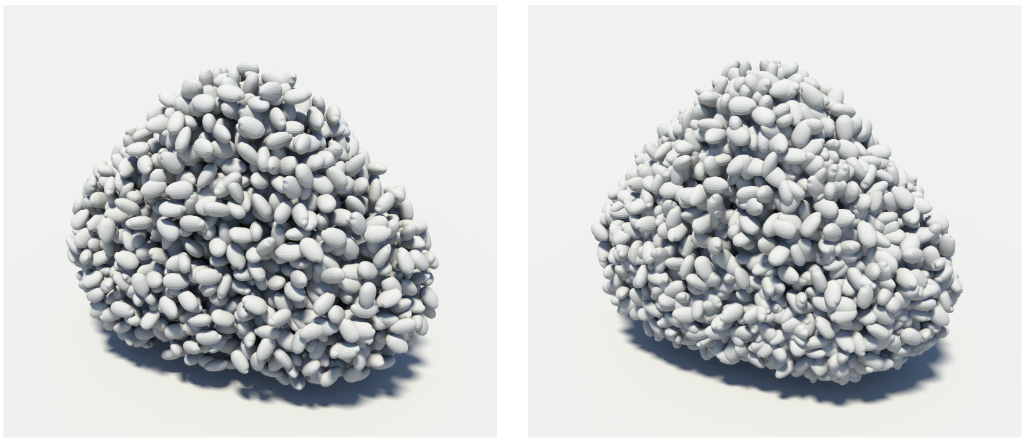

(a) Without collision ($T = 0.0$). (b) With collision ($T = 1.0$).

Figure 4.12: Changing the range for collision.

gate. The components do not disperse and fall because the proposed method generates aggregates without a physical simulation.

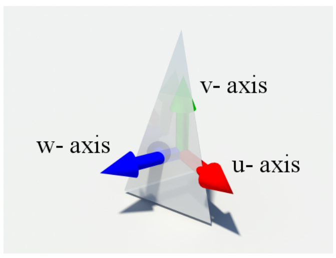
Table 4.1 shows the parameters and calculation times for the results. In the experiments, offset parameter T is set to 0.2 for dense aggregates, except for Fig. 4.12. By setting T to 0, collision is avoided; however, the aggregates are sparse, and the number of gaps among components is remarkable.

Figure 4.17 shows objects fabricated by a 3D printer (ZPrinter 650). Because the fabricated objects do not have adhesive properties, discrete components might be disconnected. Overlapping of each component is maintained by adjusting the range of C_{refine} or collision T. In this experiment, T is set to 0.3 or more for fabrication.

Figure 4.18 shows two fabricated objects generated from banana components. In Fig. 4.18 (a), bananas are rotated along the surface normals of the bunny model. In Fig. 4.18 (b), bananas are randomly rotated. By setting an appropriate range for C_{refine} and collision T, both models were fabricated. Otherwise, the fabricated objects would be broken as shown in Fig. 4.19.

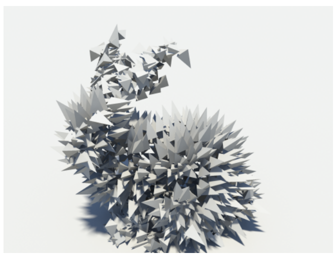
4.5 Evaluation

I interviewed two professional groups of creators. They appreciated the appearances of all the piled up aggregates. The arrangements of the components were evaluated as having natural placements. However, they mentioned that there is

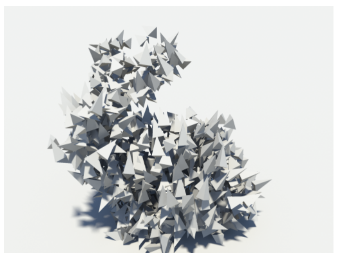


e

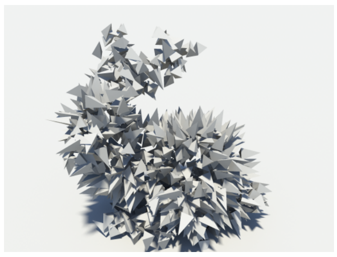
(a) component.



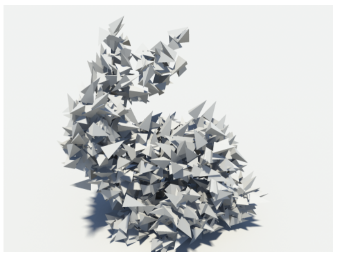
(b) $u_r = 0$, $v_r = 0$, $w_r = 0$



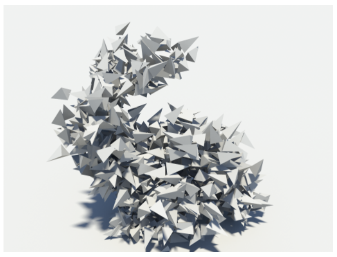
(c) $u_r = 2\pi$, $v_r = 0$, $w_r = 0$



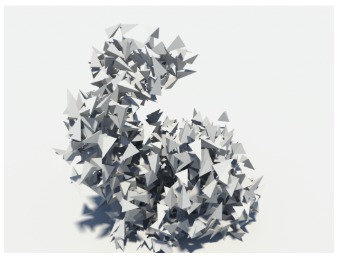
(d) $u_r = 0$, $v_r = 2\pi$, $w_r = 0$



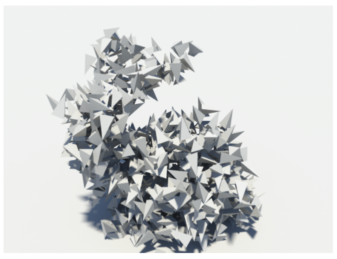
(e) $u_r = 0$, $v_r = 0$, $w_r = 2\pi$



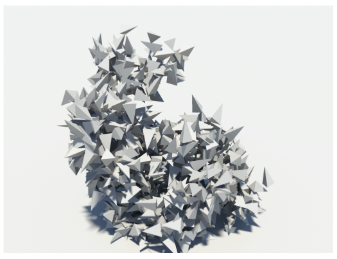
(f) $u_r = 0$, $v_r = 2\pi$, $w_r = 2\pi$



(g) $u_r = 2\pi$, $v_r = 0$, $w_r = 2\pi$

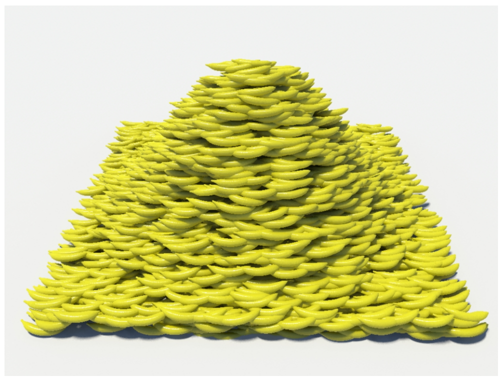


(h) $u_r = 2\pi$, $v_r = 2\pi$, $w_r = 0$

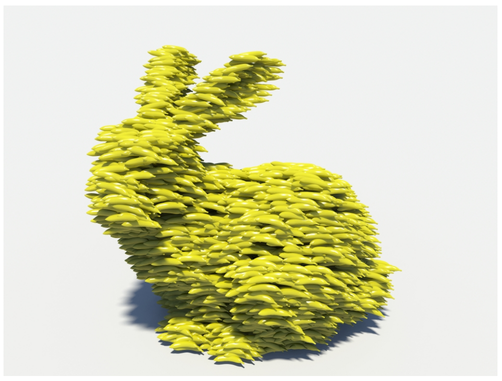


(i) $u_r = 2\pi$, $v_r = 2\pi$, $w_r = 2\pi$

Figure 4.13: Controlling component orientation.



(a) Gauss.

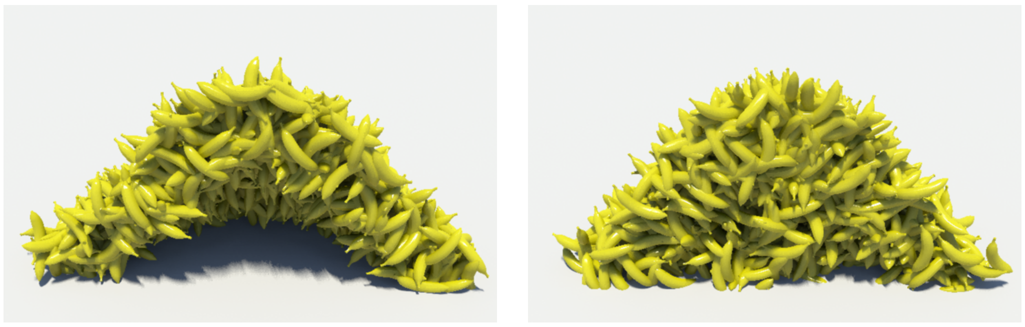


(b) Bunny.

Figure 4.14: Orientation in the global coordinate system.

Table 4.1: Parameters, calculation time, and number of components

	C_{radius}	u_r, v_r, w_r	C_{refine}	Time [s]	Number
Fig. 4.10 (a)	1.0	$0, 2\pi, 0$	2.0	324.2	4524
Fig. 4.10 (c)	1.0	$0, 2\pi, 0$	2.0	246.5	1234
Fig. 4.10 (e)	1.0	$0, 2\pi, 0$	2.0	182.2	2089
Fig. 4.11 (a)	Torus:4.0 Ball:2.0 Box:1.0	$2\pi, 2\pi, 2\pi$	Torus:1.0 Ball:2.0 Box:3.0	51.1	2523
Fig. 4.11 (b)	Nut:3.0 Rice:1.0	$2\pi, 2\pi, 2\pi$	Nut:0.1 Rice:1.5	161.99	7400
Fig. 4.12 (a)	1.0	$2\pi, 2\pi, 2\pi$	3.0	145.1	2123
Fig. 4.12 (b)	1.0	$2\pi, 2\pi, 2\pi$	3.0	150.1	4770
Fig. 4.13 (b)	1.0	0, 0, 0	1.0	58.0	882
Fig. 4.13 (c)	1.0	$2\pi, 0, 0$	1.0	59.9	882
Fig. 4.13 (d)	1.0	$0, 2\pi, 0$	1.0	57.5	882
Fig. 4.13 (e)	1.0	$0, 0, 2\pi$	1.0	61.5	882
Fig. 4.13 (f)	1.0	$0, 2\pi, 2\pi$	1.0	60.1	882
Fig. 4.13 (g)	1.0	$2\pi, 0, 2\pi$	1.0	55.2	882
Fig. 4.13 (h)	1.0	$2\pi, 2\pi, 0$	1.0	54.7	882
Fig. 4.13 (i)	1.0	$2\pi, 2\pi, 2\pi$	1.0	56.5	882
Fig. 4.14 (a)	2.0	$0.1\pi, 0.1\pi, 0.1\pi$	3.0	339.0	2856
Fig. 4.14 (b)	2.0	$0.1\pi, 0.1\pi, 0.1\pi$	3.0	270.6	2557
Fig. 4.15 (a)	1.0	$2\pi, 2\pi, 2\pi$	2.0	203.9	2986
Fig. 4.15 (b)	1.0	$2\pi, 2\pi, 2\pi$	2.0	244.0	3686
Fig. 4.16 (a)	7.5	$0, 2\pi, 0$	6.0	19.5	490
Fig. 4.16 (b)	Nut:7.0 bolt:2.0	$2\pi, 2\pi, 2\pi$	Nut:3.0 Bolt:3.0	45.4	1148
Fig. 4.16 (c)	1.0	$0, 2\pi, 0$	1.0	101.7	897
Fig. 4.16 (d)	Nut:4.0 bolt:2.0	$2\pi, 2\pi, 2\pi$	Nut:6.0 Bolt:6.0	679.0	5939
Fig. 4.16 (e)	7.5	$0, 2\pi, 0$	6.0	30.5	676
Fig. 4.16 (f)	Nut:7.0 bolt:1.5	$2\pi, 2\pi, 2\pi$	Nut:5.5 Bolt:5.0	62.4	2342



(a) Single mesh.　　(b) Conical mesh.

Figure 4.15: Inside of the solid texture.

no variation of the organic components in a generated aggregate, although real organic components have fluctuations. This method does not consider deformation of the components. To express natural appearances, it is important in future to consider the deformation for representing organic components.

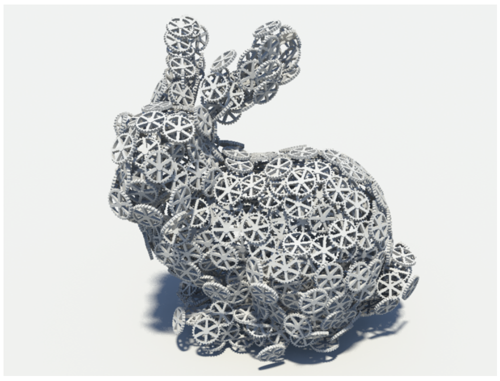
The creators expressed interest in using the proposed method in their work. They appreciated the qualities of the generated aggregates and the controllability of the aggregates by using the parameters in this method.

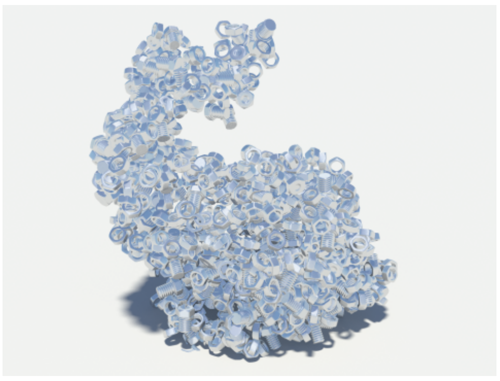
4.6 Limitation and Future Work

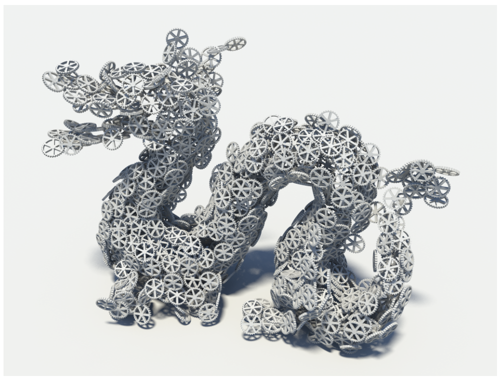
The proposed method can generate variations in aggregates by changing the number of components, aggregate shapes, and parameters. It is easy to implement this method that embodies the dart-throwing method, implicit surface, and a golden section search.

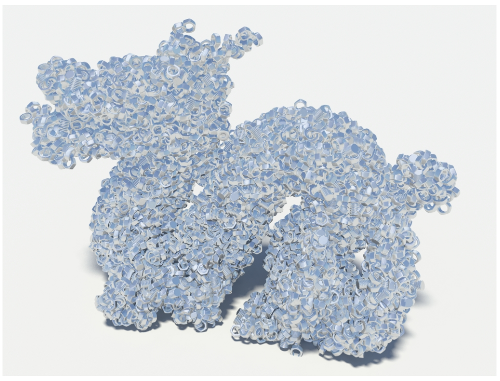
The method considers that an aggregate consists of components that have adhesion. The method supposes that the aggregate is not fragile. The shape of the generated aggregate is directly specified with an aggregate shape. Physical stability of the aggregate strongly depends on the aggregate shape. The method does not consider the physical stability of the aggregate because the method entrusts the creators with the aggregate shape.

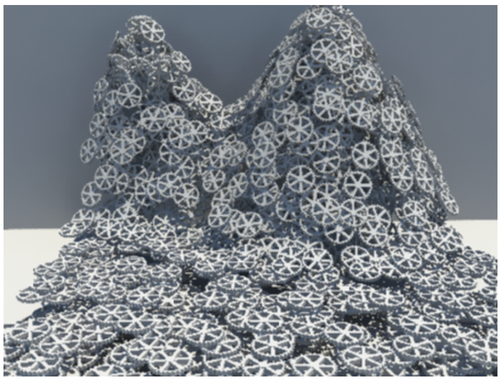
The implementation uses parallel processing in a GPU. One calculation step searches all neighbors by parallel processing. The position of each component is determined by the relative distance among its neighbors. Therefore, the calculation cost is $O(N)$ for N placed components. However, the implementation is not real time because the data transfer time between the CPU and the GPU is long.


(a) Component: gear, Shape: bunny.


(b) Component: bolt and nut, Shape: bunny.


(c) Component: gear, Shape: dragon.


(d) Component: bolt and nut, Shape: dragon.


(e) Component: gear, Shape: wave.

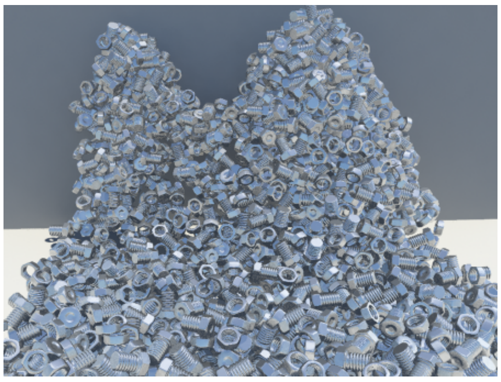
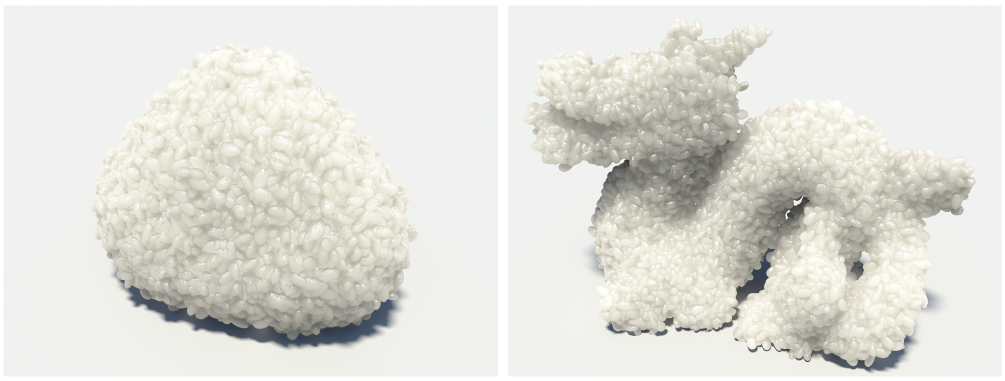
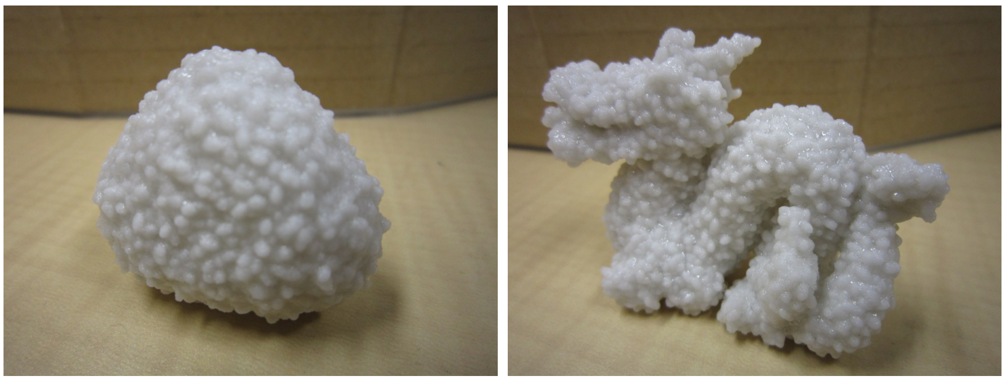

(f) Component: bolt and nut, Shape: wave.

Figure 4.16: Variations in results of proposed method.

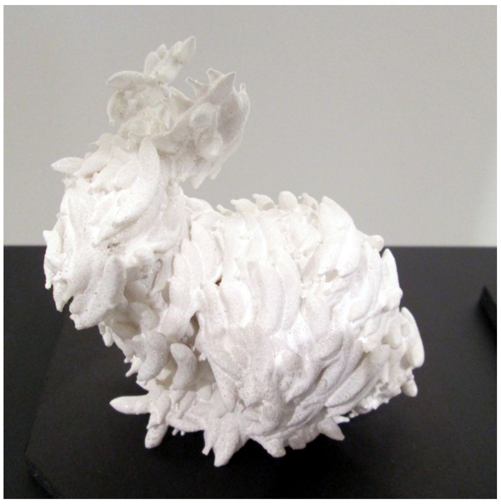


(a) Rendered aggregate data.



(b) Fabricated object.

Figure 4.17: Fabrication example.


(a) Fabricated banana bunny 1.

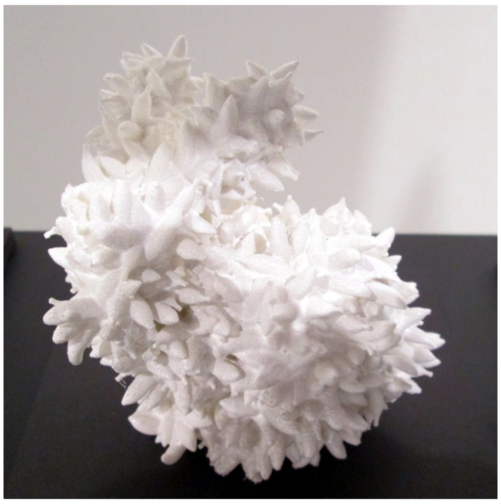

(b) Fabricated banana bunny 2.

Figure 4.18: Fabrication example # 2.

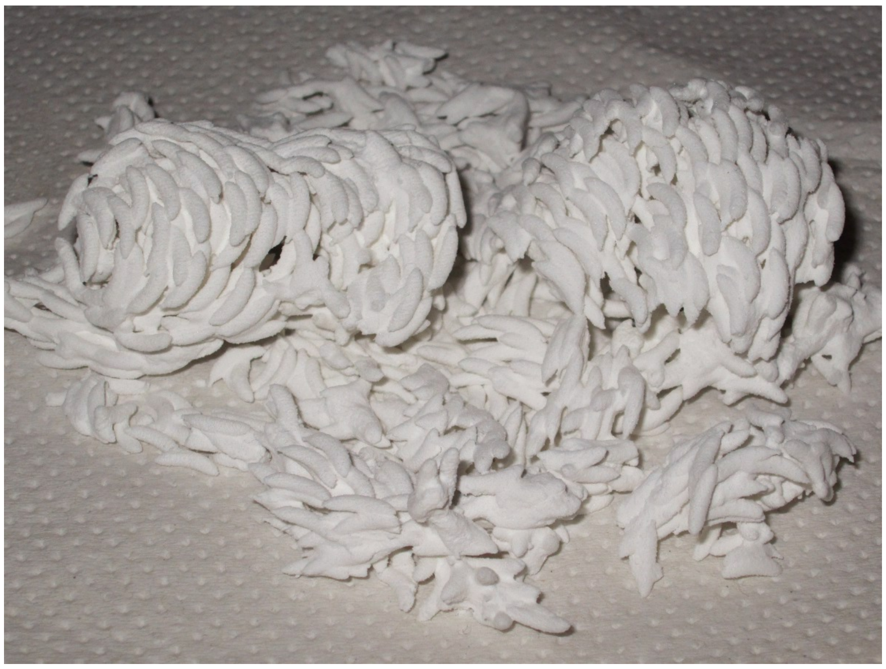


Figure 4.19: Broken printed aggregate.

The approach is designed for uniform and random distribution, but the above-mentioned method is not suitable for comprehensive aggregates, for example, aggregates with roughness and fineness as well as those containing flows such as curls or divergences. It is necessary to use a distribution method that includes white-noise spectral properties if a creator wants to represent holes and gaps in an aggregate, such as an aggregate that contain floating objects such as bubbles in water. In addition, by generating vector fields on the aggregate shape [129], the flow of component orientations can be intuitively controlled.

Many components are placed in the placement step if the placed components have many holes or strong anisotropy because their largest inner spheres tend to be small. The calculation time of the refinement step is in approximate proportion to the number of components because the method moves each component. Therefore, an aggregate composes of many components results in high calculation costs. It is necessary to optimize the distribution of the dart-throwing method so that the generated aggregate includes fewer components for maintaining a natural appearance.

The method applied the single component procedure to each type of component and resolved the collisions later to generate an aggregate composed of multiple types of components. This is not efficient when considering many types of components; however, it is difficult to find a solution for the distribution of multiple types of components. Although Wei proposed multiclass sampling [118], this sampling is used only for points and not for components having volumes. To effectively distribute multiple types of components, a new technique for sampling or distribution needs to be developed.

The method did not consider aggregate compounds of open meshes such as thin objects like sheets and papers. To handle thin objects, it is necessary to quantify the overlaps among open meshes.

Chapter 5

Generation Method for Aggregate of Staple Fibers

Chapters 3 and 4 presented two methods for generating arbitrary components. This chapter focuses on fibers that are long and deformable, such as cotton and wool, as one of various components.

Dust balls composed of staple fibers frequently appear in our daily lives. Several methods have been proposed to render a collection of long fibers such as hair; however, they cannot generate sparse aggregates composed of staple fibers. Because the shape of each fiber is noticeable in a sparse aggregate, it is necessary to represent various staple fiber shapes in order to express a visually plausible aggregate. This chapter presents a procedural modeling method for generating staple fibers because such a method can generate a wide range of objects. The fiber model is defined by a polygonal chain to represent various fibers having crimps in the procedural modeling method. The crimping of fibers is controlled by changing the angles between neighboring line segments in a polygonal chain. The method generates an aggregate from a target shape and the parameters for the crimping of fibers. The parameters are inputs, and the method outputs vertices of the polygonal chains in the aggregate. Collisions among staple fibers are not considered. This chapter presents examples of generated aggregates to verify the effectiveness of the proposed method.

In addition, this chapter shows that the method can be applied to other long components, for example a nest.

5.1 Introduction

Numerous methods have been proposed to represent objects composed of large number of fibers, such as cloth, hair, and fur. However, it is difficult to generate sparsely aggregated staple fibers. Cloth animation and material representation methods for fabrics have been previously investigated [15, 47, 51, 87, 99]. These methods do not generate individual fibers; therefore, the noticeable fibers in a sparse aggregate are not represented. There is a method for generating woven cloth consisting of individual staple fibers [100]; however, it does not form arbitrary-shaped aggregates. In addition, the representation of aggregates generally requires many tedious, time-consuming manual operations, as do other aggregates.

Many research papers that deal with the representation of a set of long fibers such as hair and fur have been published [9, 10, 40, 50, 89, 105, 116, 128]. These methods contribute to improve the reality of computer-generated images; however, most of them focus on the representation of the macroscopic characteristics of fibers. The objective of the proposed method is to represent a meso-structure of aggregated fibers.

To do so, this chapter attempts to generate the aggregates without collision detection because it is impossible to calculate the collision detection of fibers without their volumes. In addition, it is very expensive to calculate collision detections for numerous deformable fibers. The fibers in a dust ball are not regular; thus, it is difficult to anticipate when and where collisions will occur. Thus, a physical simulation that includes collision and response is not used for generating the aggregate, and only geometric operations are used to constrain aggregate formations.

5.2 Process Overview

The proposed method consists of two procedures: (1) procedural modeling of staple fibers and (2) aggregate formations comprised of generated fibers. It is necessary to fill staple fibers on the inside because the inside of the aggregate can be seen. Thus, the method presented in Chapter 4 cannot be applied to an aggregate. Staple fibers are placed in the input target shape in the procedure (2) to fill the inside. However, parts of the staple fibers are outside the target shape. The fibers are moved into the target shape to fit the aggregate to the target shape.

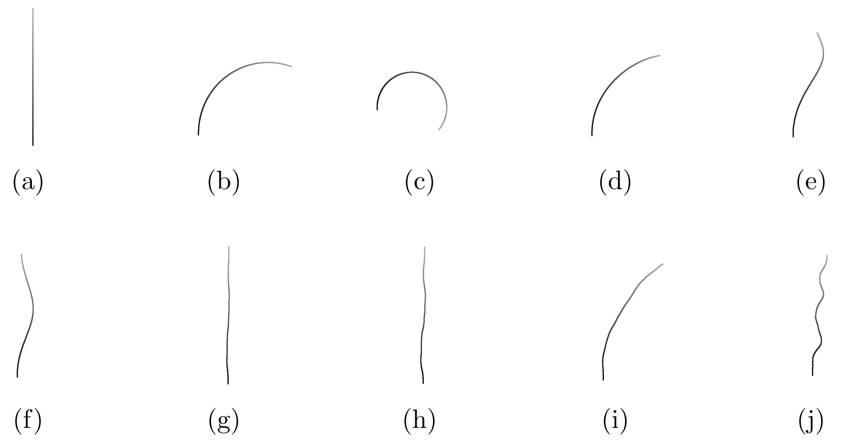


Figure 5.1: Generated staple fiber.

Table 5.1: Staple fiber parameters

	a	b	c	d	e	f	g	h	i	j
$\theta_{min}(^\circ)$	0.0	3.6	7.2	3.6	3.6	3.6	-3.6	-7.2	-3.6	-3.6
$\theta_{max}(^\circ)$	0.0	3.6	7.2	3.6	3.6	3.6	3.6	7.2	7.2	18.0
$\phi_{min}(^\circ)$	0.0	0.0	0.0	3.6	7.2	10.8	0.0	0.0	0.0	21.0
$\phi_{max}(^\circ)$	0.0	0.0	0.0	3.6	7.2	10.8	0.0	0.0	0.0	39.6

5.2.1 Procedural Modeling of Staple Fiber

The proposed method represents a staple fiber as a polygonal chain model, as illustrated in Fig. 5.1. The shape is generated by twisting each segment sequentially, as shown in Fig. 5.2. Here, the y-axis is oriented in the direction of the i-th segment $v_i - v_{i+1}$; then, the $(i+1)$-th segment $v_{i+1} - v_{i+2}$ is twisted by θ on the z-axis and ϕ on the y-axis in a local coordinate. θ and ϕ are selected randomly within the range $[\theta_{min}, \theta_{max}]$ and $[\phi_{min}, \phi_{max}]$, respectively. In this implementation, the number of segments for each fiber is fixed at 32. Table 5.1 shows the parameters for the results in Fig. 5.1.

5.2.2 Forming Aggregate

An aggregate of fibers is formed by gathering fiber elements loosely and gently; thus, the elements are not overly deformed to fit the target shape of an aggregate. As shown in Fig. 5.3, the proposed method consists of two procedures to form an aggregate comprised of staple fibers: (1) initially placing fibers inside the target

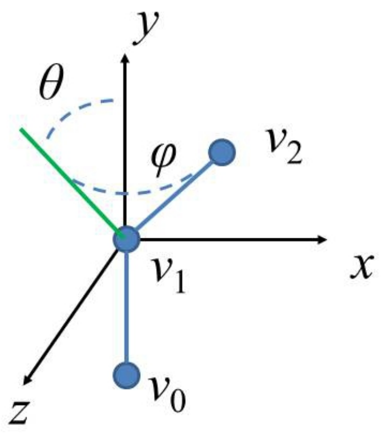
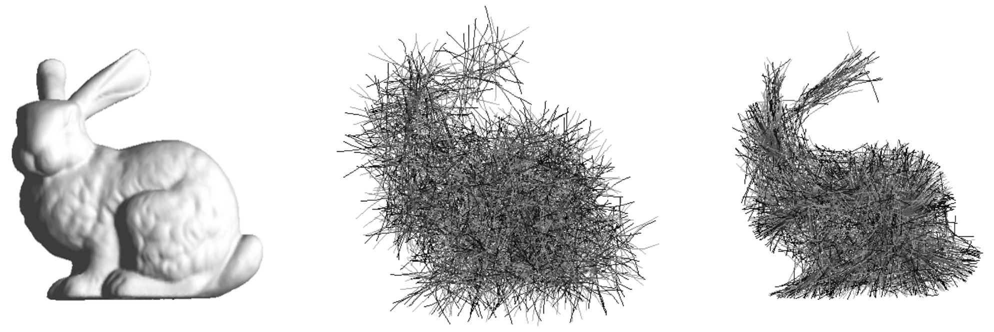


Figure 5.2: Polygonal chain model.



(a) Target shape. (b) Initial aggregate. (c) After adjustment.

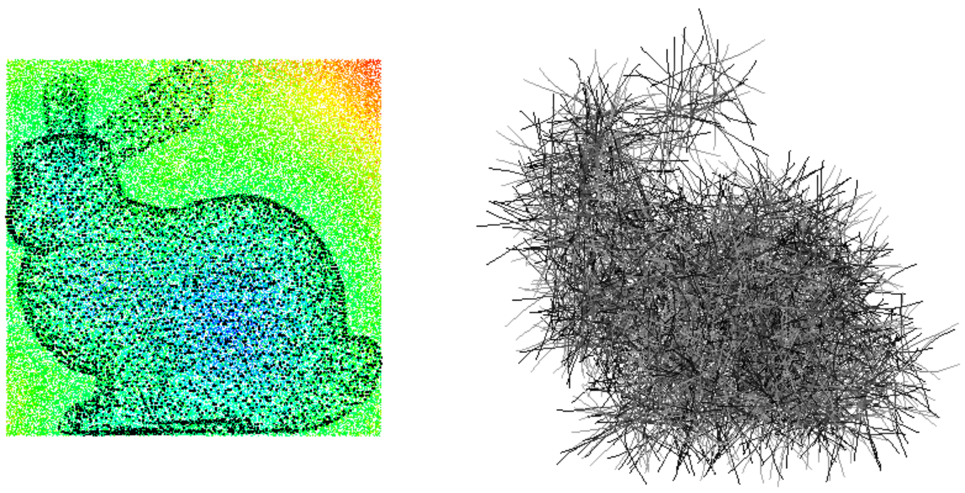
Figure 5.3: Process overview.

shape (Fig. 5.3 (a)) as shown in Fig. 5.3 (b), and (2) adjusting the fibers to fit the target shape, as shown in Fig. 5.3 (c). The aggregate shape is specified using a 3D polygon mesh whose vertices have normal vectors.

5.2.2.1 Initial Placement

When placing a generated staple fiber to form a target shape, it is necessary to determine if the fiber is located within the target shape. The proposed method uses the signed geometric distance [43] for this determination.

The signed distance d from an arbitrary point $\mathbf{x}$ to point $\mathbf{v}$ on the surface of



(a) Signed geometric distance. (b) Initial placement.

Figure 5.4: Initial placement of fiber.

a target shape is defined by Eq. 5.1:

$$d(\mathbf{x}) = (\mathbf{x} - \mathbf{v}) \cdot \mathbf{n}, \tag{5.1}$$

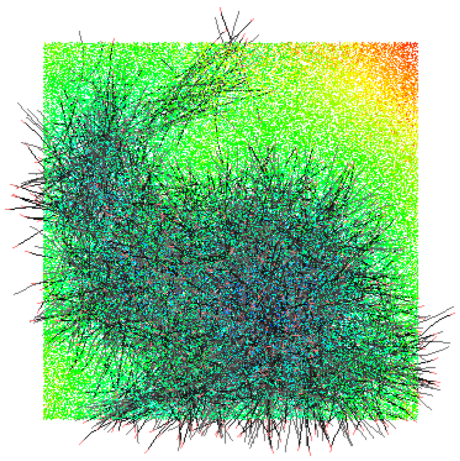
where $\mathbf{v}$ is the nearest point from $\mathbf{x}$ and $\mathbf{n}$ is a normal vector at $\mathbf{v}$. For each point, $\mathbf{n}$ are precalculated.

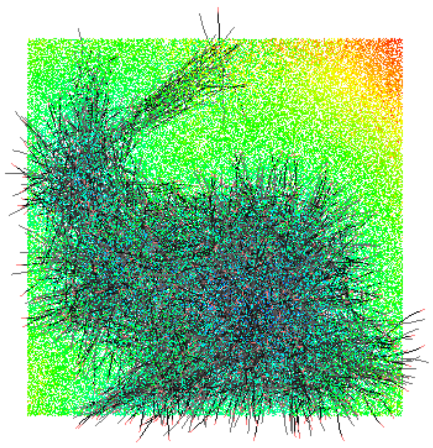
Figure 5.4 (a) shows color mapping based on distance values at a randomly distributed point in a 3D space; blue, green, and red indicate high, middle, and low values, respectively, and the black point is a vertex.

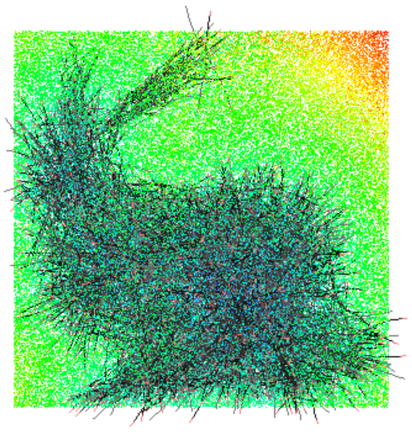
A staple fiber is placed such that its center is located inside the target shape of an aggregate by referring to the calculated distance values. Figure 5.4 (b) shows the result of placing 2048 fibers. As shown in this example, several fiber segments stick out from the target shape. The next process adjusts these fibers to fit the target.

5.2.2.2 Adjustment

The adjustment process relocates the fibers iteratively to fit the target shape by optimizing the cost function. The proposed method does not apply this process to a segment that is entirely involved within the target shape. The cost function


(a) 5 iterations


(b) 10 iterations.


(c) 15 iterations.

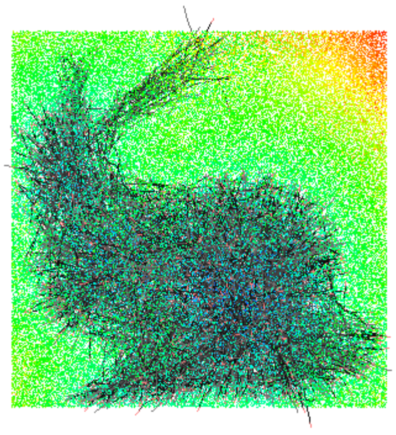

(d) 20 iterations.

Figure 5.5: Adjustment.

$C(\mathbf{X})$ is given by Eq. 5.2:

$$C(\mathbf{X}) = \sum_{\mathbf{x}_i \in \mathbf{X}} d(\mathbf{x}_i), \tag{5.2}$$

where $\mathbf{X}$ is a set of vertices of all segments in a fiber. The more a fiber is within the target shape, the higher the value returned from the cost function. Optimization is resolved by searching the maximum value of $C(\mathbf{X})$ while shifting and rotating a fiber object within a specified range. This process does not consider collisions among elements.

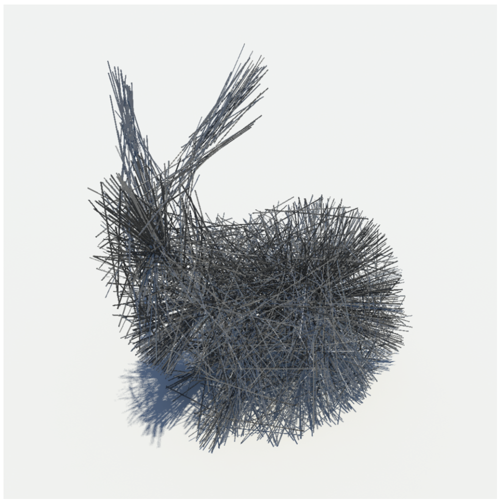
Figure 5.5 shows the sequence of the adjustment process in five-iteration increments. As shown in these examples, the fibers are gradually fitted into the target shape. This adjustment is repeated until it becomes a steady state.

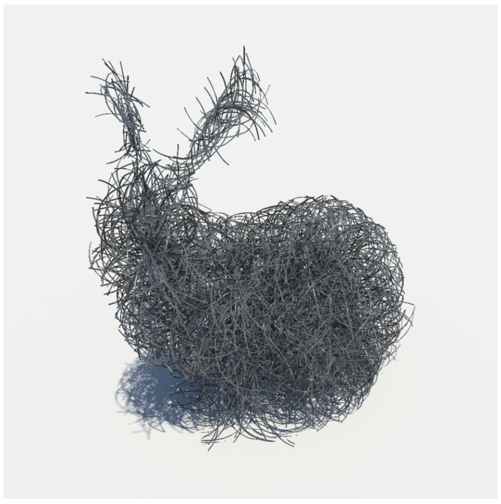
5.3 Result

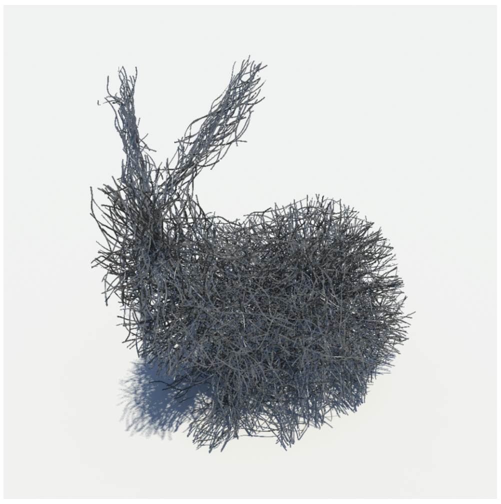
The proposed method is implemented on Windows 7 with an Intel Core i7, 3.07 GHz CPU, 12.0 GB RAM, and NVIDIA GeFORCE GTX 570 using C++ with CGAL and OpenCL.

Each fiber is converted into a set of fine cylinders prior to the rendering process. Figure 5.6 shows the results that are generated by changing the parameters of the fibers. Table 5.2 shows the parameters, calculation cost, and the number of iterations in the adjustment process for Fig. 5.6. Here, $t1$ is the cost for the initial placement of fibers, $t2$ is the average cost for one adjustment process, and I is the number of iterations for the adjustment process. The fiber length and the number of fibers in Figs. 5.6 and 5.7 are 10.0 and 2048, respectively. The parameters are set to generate straight fibers (a), curved fibers (b), varied fibers (c), and spiral fibers (d). Figure 5.6 (b) shows that the curved fibers are well aligned along the gentle curved surfaces, while not many straight fibers are placed there, as shown in Fig. 5.6 (a). Regarding the computation cost, there is no obvious difference among the costs for the initial placement of fibers.

Figure 5.7 shows the results for a dragon model used as the target shape. The parameters of the fibers are set as for Fig. 5.6. Table 5.3 show the parameters, calculation costs, and number of iterations in the adjustment process shown in Fig. 5.7. Comparing Tables 5.2 and 5.3, the cost for the adjustment process depends on the number of vertices in a model for the following reason: The adjustment process is parallelized on a GPU in this experiment; however, the


(a)


(b)


(c)

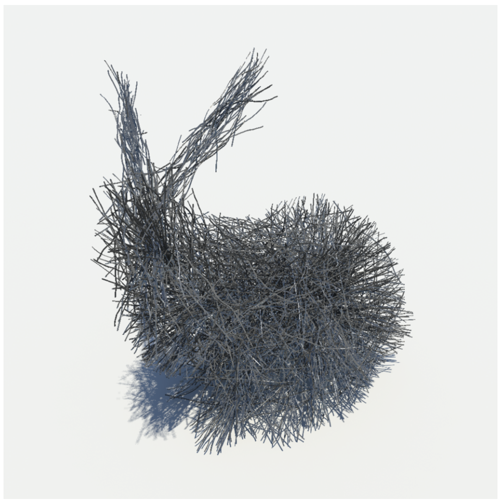
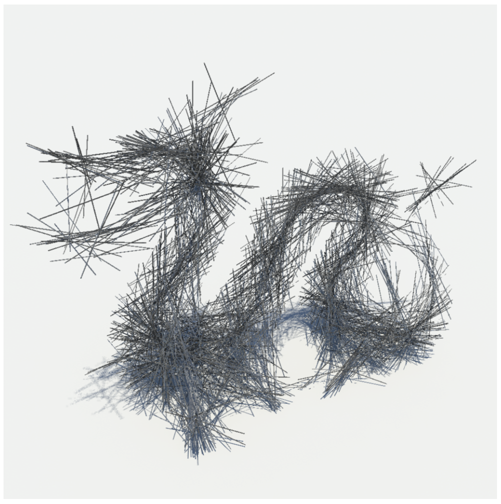
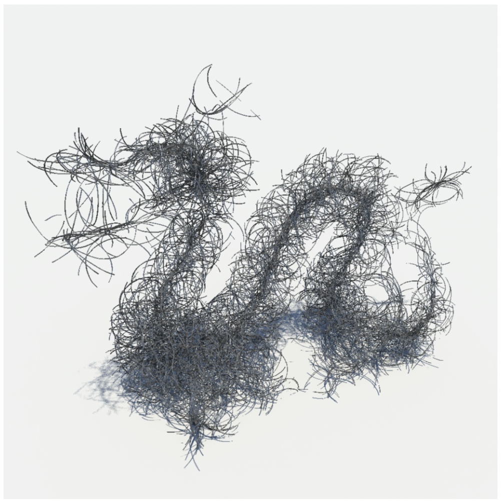
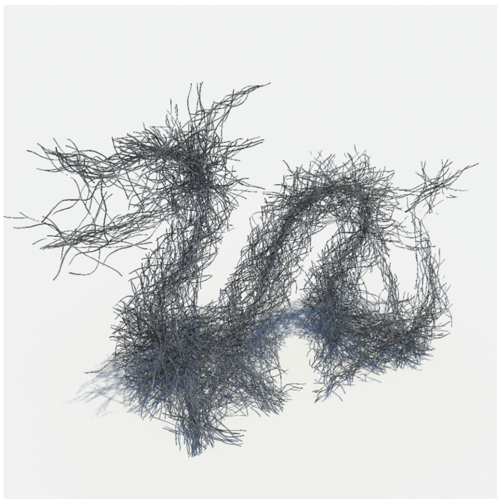

(d)

Figure 5.6: Variations in bunny shape.


(a)


(b)


(c)

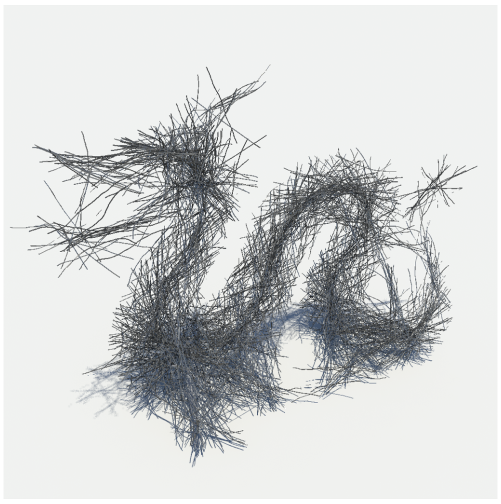

(d)

Figure 5.7: Variations in dragon shape.

Table 5.2: Parameters for Fig. 5.6

	$\theta_{min}(^\circ)$	$\theta_{max}(^\circ)$	$\phi_{min}(^\circ)$	$\phi_{max}(^\circ)$	$t1$	$t2$	I
(a)	0.0	0.0	0.0	0.0	0.015	0.860	35
(b)	0.0	10.8	0.0	0.0	0.016	0.860	54
(c)	-10.8	10.8	0.0	0.0	0.016	0.860	37
(d)	0.0	10.8	0.0	36.0	0.016	0.835	31

Table 5.3: Parameters for Fig. 5.7

	$\theta_{min}(^\circ)$	$\theta_{max}(^\circ)$	$\phi_{min}(^\circ)$	$\phi_{max}(^\circ)$	$t1$	$t2$	I
(a)	0.0	0.0	0.0	0.0	0.016	1.210	28
(b)	0.0	10.8	0.0	0.0	0.016	1.202	37
(c)	-10.8	10.8	0.0	0.0	0.016	1.217	27
(d)	0.0	10.8	0.0	36.0	0.016	1.201	30

Table 5.4: Parameter for Fig. 5.8

	$t1$	$t2$	I
a	0.016	0.863	71
b	0.016	0.859	17

Table 5.5: Parameter for Fig. 5.9

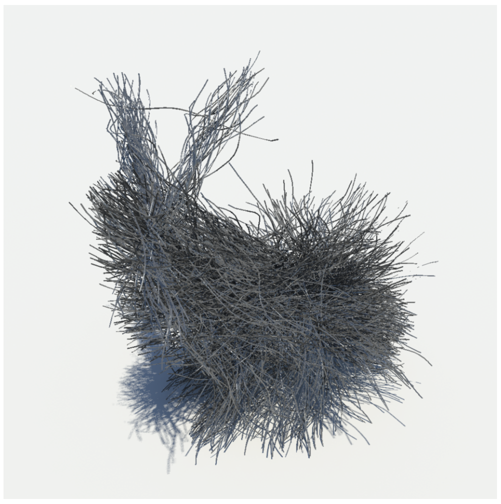
	$t1$	$t2$	I
a	0.016	0.444	31
b	0.016	0.209	32

time for data transfer between the CPU and the GPU significantly depends on the volume of data to be transferred.

Figure 5.8 shows a comparison of changing fiber lengths. The other parameters of the fibers are the same as those in Fig. 5.6 (d). Figure 5.8 (a) shows that the long fibers are not placed around the curved surfaces. Table 5.4 shows that the computation costs of these examples are almost the same, while the number of iterations for the adjustment process depends on the linear fiber length: the longer the fiber, the more iterations executed by the process.

Figure 5.9 shows a comparison in which the number of fibers is changed. The other parameters of the fibers are the same as those in Fig. 5.6 (d). Table 5.5 shows that the average cost for one adjustment process simply depends on the number of fibers.

Figures 5.10 – 5.13 show variations and Table 5.6 shows the parameters. Figures 5.12 and 5.13 use the same parameters. Figure 5.10 shows a rendering image


(a) Fiber length:20.0.

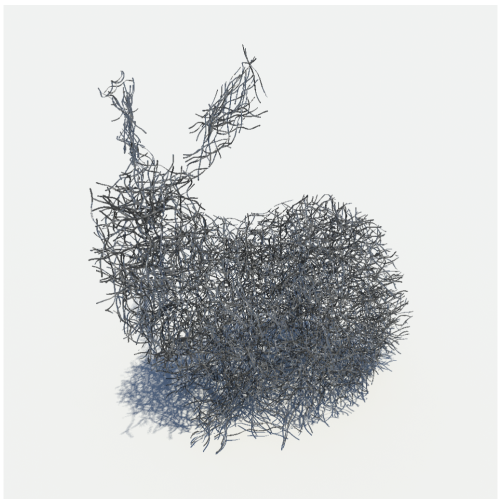
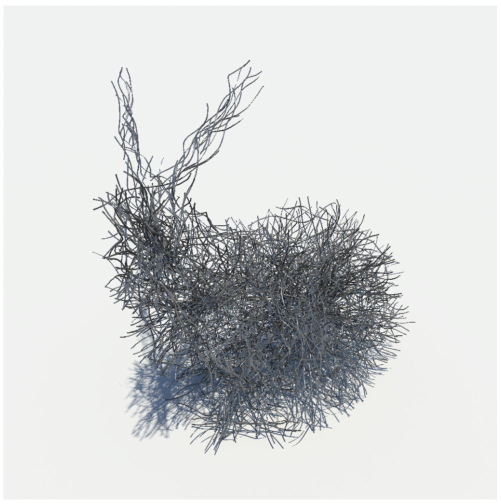

(b) Fiber length:5.0.

Figure 5.8: Changing fiber length.


(a) No. of fibers: 1024.

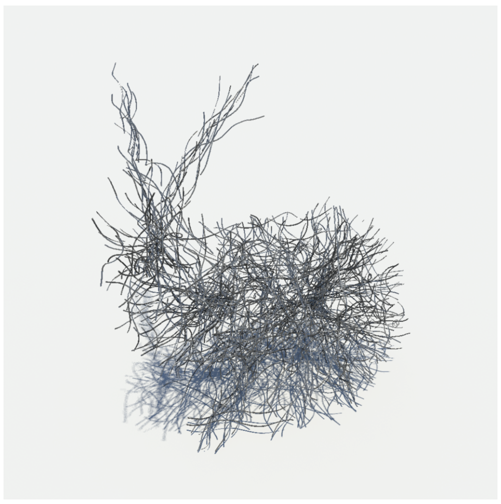

(b) No. of fibers: 512.

Figure 5.9: Changing the number of fibers.

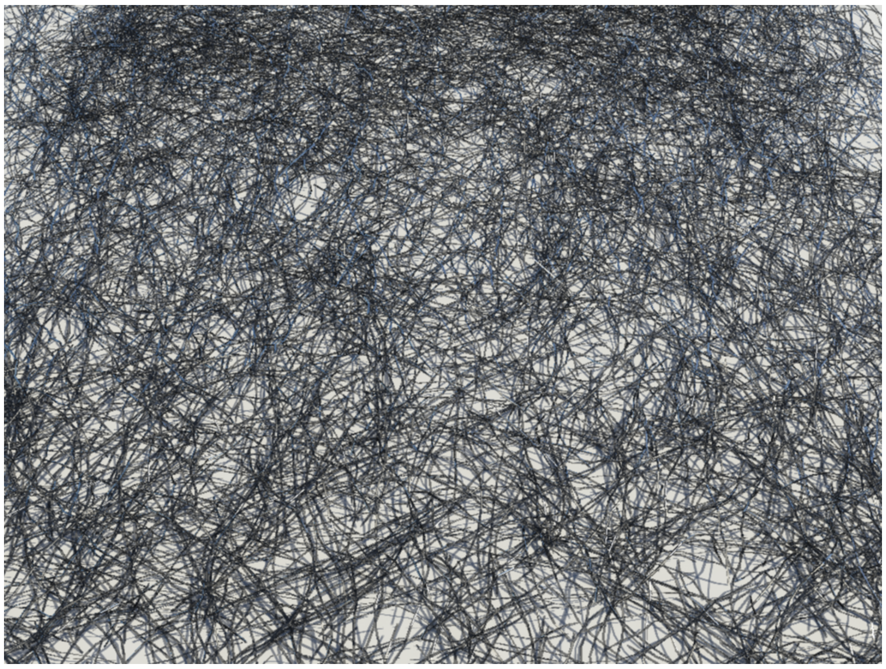


Figure 5.10: Scattered staple fibers.

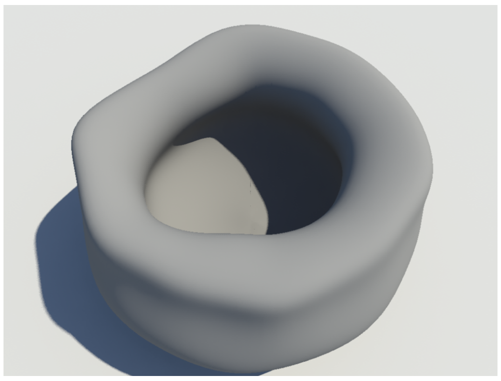
of scattered staple fibers on a table. Figure 5.11 shows generated nests. Figures 5.12 and 5.13 shows examples of changing target shapes, a sphere, and a Gaussian-shaped aggregate. These figures show that the method can be used for representing many fiber types.

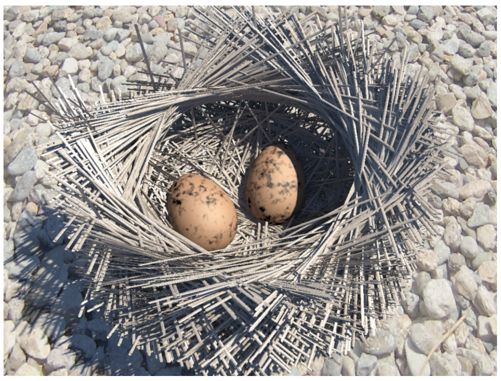
5.4 Evaluation

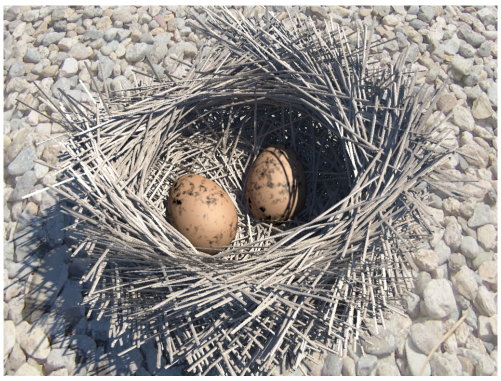
I interviewed two professional groups of creators. They appreciated staple fibers and found their aggregates to be interesting expressions and convenient for expressing dust on clothes and animals. The expression of dust is difficult by manual operation. The proposed method lets creators produce the details of clothes and rooms.

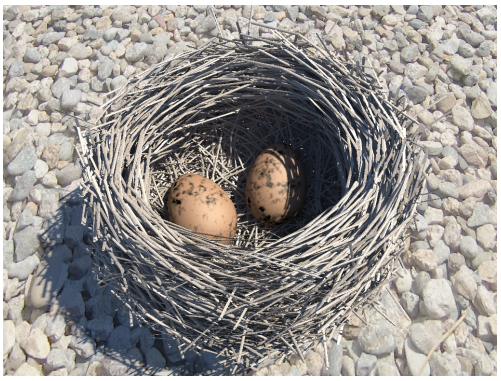
The method generates variations in components, and a creator said, “Variations make natural appearances.” The answers given in my interview indicate that parametric control of the shapes of components is effective in expressing natural aggregates.

One creator pointed out the proposed method does not consider the flow of fibers such as hair. The creator has an interest in arbitrary-shaped hair, and he


(a) Target shape.


(b) Generated aggregate 1.


(c) Generated aggregate 2.


(d) Generated aggregate 3.

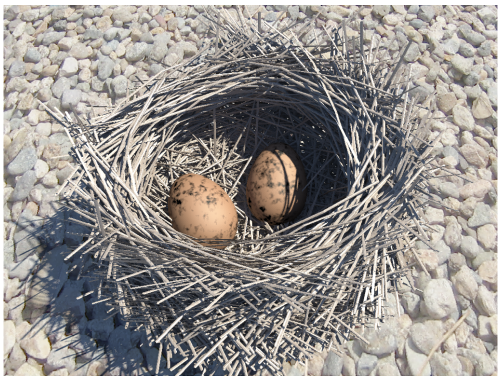
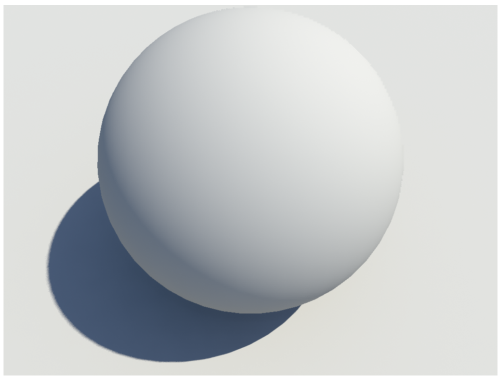
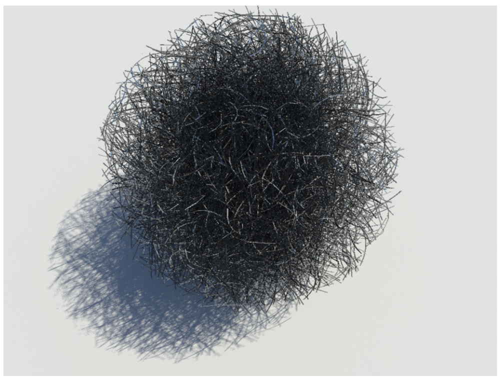
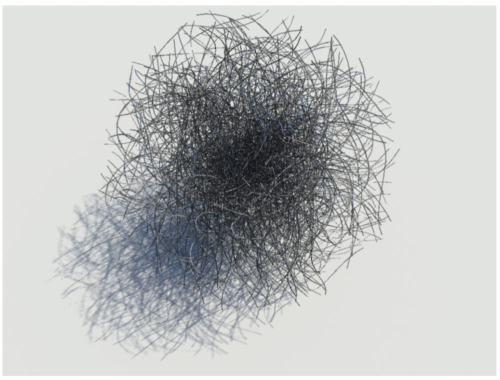
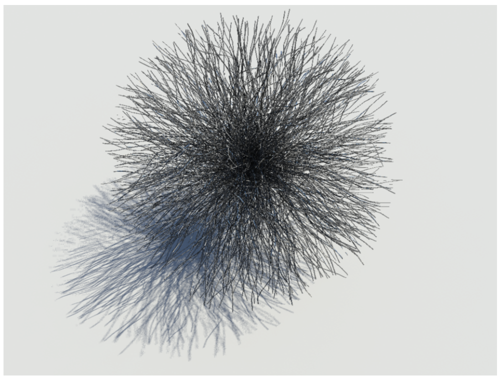

(e) Generated aggregate 4.

Figure 5.11: Bird nest.


(a) Target shape.


(b) Generated aggregate 1.


(c) Generated aggregate 2.


(d) Generated aggregate 3.

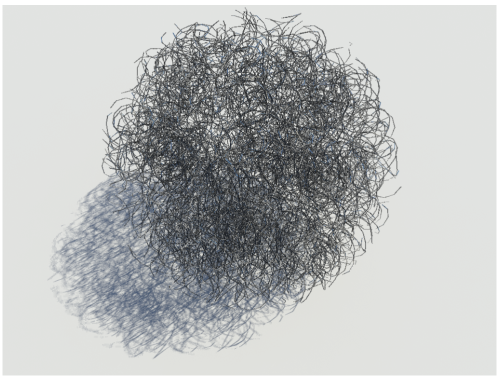
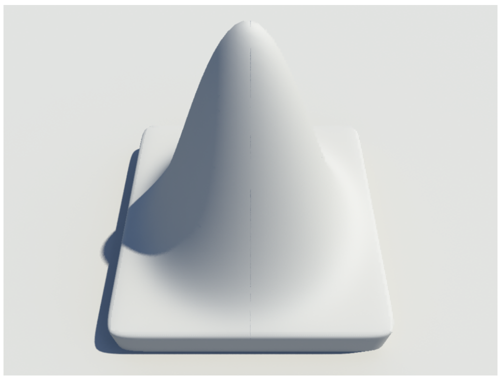
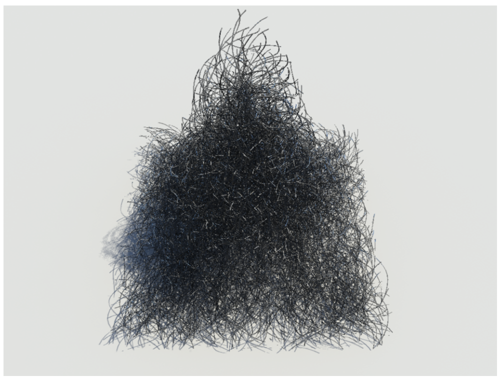
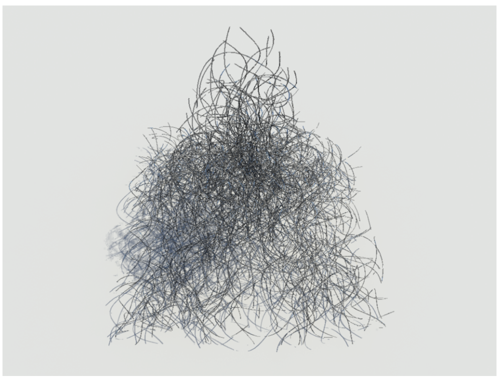
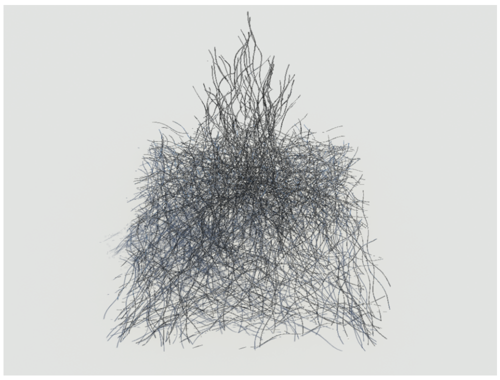

(e) Generated aggregate 4.

Figure 5.12: Sphere-shaped aggregate.


(a) Target shape.


(b) Generated aggregate 1.


(c) Generated aggregate 2.


(d) Generated aggregate 3.

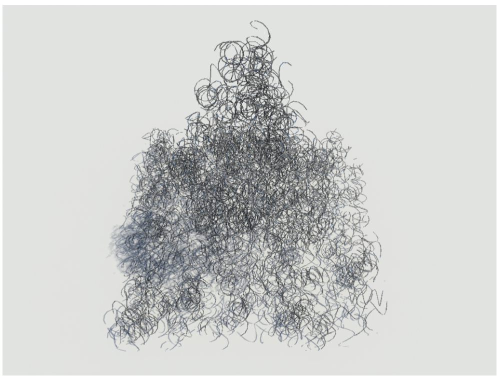

(e) Generated aggregate 4.

Figure 5.13: Gaussian-shaped aggregate.

Table 5.6: Parameters for Figs. 5.10 – 5.12

	5.10	5.11 (b)	5.11 (c)	5.11 (d)	5.11 (e)
$\theta_{min}(^\circ)$	-25.2	0	-21.6	21.6	0
$\theta_{max}(^\circ)$	25.2	0	21.6	21.6	21.6
$\phi_{min}(^\circ)$	-25.2	0	0	0	0
$\phi_{max}(^\circ)$	25.2	0	0	0	0
No. of Segments	32	6	6	6	6
No. of fibers	2048	2048	2048	2048	2048
	5.12 (b)	5.12 (c)	5.12 (d)	5.12 (e)	
$\theta_{min}(^\circ)$	7.2	7.2	7.2	28.8	
$\theta_{max}(^\circ)$	14.4	14.4	14.4	36.0	
$\phi_{min}(^\circ)$	7.2	7.2	28.8	7.2	
$\phi_{max}(^\circ)$	14.4	14.4	43.2	14.4	
No. of Segments	32	32	32	32	
No. of fibers	2048	512	512	512	

asked whether the method could generate it. The question implies that modeling methods for expressing fibers are demanded.

Another creator wanted to express novel characters consisting of fibers in an animation by using this method. This proposal indicates that the method can be used for not only dust but also many new expressions.

5.5 Conclusion

The proposed method generates various aggregates by changing the fiber parameters. This method supports only the fibers and does not include collision detection.

In reality, fibers form an aggregate by colliding and supporting each other, and it is necessary to consider such collisions and frictions among fibers to adequately represent an aggregate of fibers. More varied aggregates, such as a bowl of noodles, could be generated by considering the deformation of elements. In future, I would like to accelerate the adjustment process by optimizing the data transfer volume.

Chapter 6

Conclusion and Future Work

This chapter concludes this dissertation by giving a summary and discussing future work.

6.1 Summary and Contribution

A goal of this study is to develop algorithms for easy modeling. To achieve this, this dissertation focused on generating aggregates as an objective. An aggregate is composed of components whose positions and orientations are nonperiodic. For example, a grain of rice is a component and a rice ball or steamed rice in a bowl is an aggregate. However, creating an aggregate model is an arduous work because creators need to model many components using graphics software. For automatic modeling, this dissertation presented three methods that generate aggregates composed of 2D components, 3D components, and staple fibers.

The first and second methods generate aggregates from arbitrary-shaped components as inputs. Although the creators need to make input models, this dissertation assumed that the workload involved in preparing the inputs is smaller than that in a previous method [71]. The third method generates aggregates composed of freeform 3D curves as staple fibers. The curves are controlled by parameters and it is easy to generate variations in staple fibers.

I interviewed professional creators about the results and usability of all proposed methods to evaluate the methods. They appreciated the breadth of the applications of proposed methods, the reduction in creation costs, the variations in the expressions, and the ease of preparing inputs for the methods.

The contribution of this dissertation to knowledge science is to suggest an approach for solving problems faced by working creators. Through the experience that I gained while working with creators for 18 months, I understand their problems and abilities. Creators usually make objects by manual operations. The editing process is that they create a rough object, and then refine it meticulously. Because this manual editing represents a heavy workload, it is necessary to reduce the workload in order to reduce the production time. Usually, creators discuss the quality of a product by comparing variations. Hence, it is helpful for creators to provide a variety of objects automatically to improve the quality of products. This dissertation proposed methods for reducing the workloads of creators and generating variations to support creators. Owing to these methods, the creators can spend more time on creative work.

Creators convey the objects they have created in a scene to other creators and workers solely by words in a discussion. However, it is difficult to understand the details of creators' imaginations solely by words, and thus discrepancies in objects resulting from the way others understand them often occur. Explicit knowledge such as images and numeric values for expressing models effectively prevent such discrepancies. The proposed methods generate aggregates from parameters, components, and target shapes. In the interviews, I drew the conclusion that creators understood what these inputs mean, and they had common imaginations for the results because they discussed the results with few questions about what input means. Thus, these methods prevent discrepancies in discussions. Preventing such discrepancies is a contribution of this dissertation to knowledge science.

However, barriers remain in achieving the goal. The methods proposed in this dissertation are insufficient to allow expressions for whole types of aggregate. To achieve the goal, I will consider the five major issues listed below.

6.2 Limitation and Future Work

The proposed methods are based on two presuppositions: components have their area and volume in 2D and 3D aggregates, respectively. For example, points as exclusive regions cannot be distributed in 2D aggregates, and sheets, which are open meshes, cannot be accumulated.

One of the issues concerns aperiodic patterns. Although this dissertation discussed how to generate nonperiodic aggregates, it did not discuss aperiodic patterns, which are nonperiodic patterns that involve packing without gaps among the components. According to a pattern book [39], aperiodic patterns are composed of specific components. Thus, this dissertation has not considered that arbitrary-shaped components are applied to aperiodic patterns. However, it is possible to consider deforming arbitrary shaped components to existing aperiodic components. By this deformation, aperiodic patterns including little collision or few gaps can be generated using arbitrary-shaped components.

The second issue is the representation of variations in component shapes. In the real world, all the shapes of components are different. To express more realistic CG, I will consider variations in component shapes in future. For this, methods [52, 56, 111, 126, 127], which generate shape variations from a few input shapes, will be used to generate component shapes in future work.

The third issue is classical mechanics in physics. The laws of physics constrain the shapes of generated aggregate. However, these laws are frequently used to improve realistic representation [20, 44, 71]. For realistic representation, I will consider applying a physical simulation to a method to generate an arbitrary aggregate in future.

The fourth issue is the control of arrangement. The proposed method controls the appearances of aggregates by changing parameters and exclusive regions. However, this dissertation has not discussed whether they are useful for creators. According to a paper by Zhou et al. [130], distributions having pink- or red-noise properties are common in nature, and those with green-noise properties have been used for halftoning and producing clustered dot printing, and have represented distribution of plumed seeds. If creators know what the components construct, it is easier to use specific noise as an initial positioning for obtaining the desired model. However, the proposed methods do not specify the orientation of a component at a position. In studies on volume modeling, Takayama et al. specified orientations by editing the vector or scalar field [109, 110]. To improve the controllability of the proposed methods, I will consider including these methods. In addition, I will consider multiple types of components. The distribution rates of the number of components are determined by exclusive regions or components shapes. A user study is important for specifying distribution rates.

The final issue is fabrication. Currently, some devices can output real objects from geometric data in a computer. 3D printers directly output objects with little manual work. However, they can only create objects consisting of a single material. Thus, we cannot edit the material. In addition, printed objects using the proposed methods may be fragile because the proposed methods do not consider physical laws. The printing aggregates ought to be checked by a physically-based simulation to keep a form or to fragment them before printing. Stava et al. proposed a method for maintaining the shape of a continuous object [107]. However, the method does not consider aggregates that potentially include gaps. Hence, it is necessary to fill the gaps without artifacts.

References

[1] Pierre Alliez, Éric Colin de Verdière, Olivier Devillers, and Martin Isenburg. Isotropic surface remeshing. In *Proceedings of the Shape Modeling International 2003*, SMI '03, pages 49–, Washington, DC, USA, 2003. IEEE Computer Society. 12

[2] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, 2001. 53, 57

[3] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 217–226, New York, NY, USA, 2001. ACM. 18

[4] Michael Balzer, Thomas Schlömer, and Oliver Deussen. Capacity-constrained point distributions: a variant of lloyd's method. *ACM Trans. Graph.*, **28**[3]:86:1–86:8, jul 2009. 12, 24

[5] Pascal Barla, Simon Breslav, Joelle Thollot, Francois X. Sillion, and Lee Markosian. Stroke pattern analysis and synthesis. *Comput. Graph. Forum*, **25**[3]:663–671, 2006. 19

[6] M. S. Bartlett. *Introduction to Stochastic Processes with Special Reference to Methods and Applications.* 1955. 15

[7] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. High-quality single-shot capture of facial geometry. *ACM Trans. Graph.*, **29**:40:1–40:9, July 2010. 18

[8] Bedrich Benes, Ondrej Stava, Radomir Mech, and Gavin Miller. Guided procedural modeling. *Computer Graphics Forum*, **30**:325–334, 2011. 16

[9] Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévêque. Super-helices for predicting the dynamics of natural hair. *ACM Trans. Graph.*, **25**[3]:1180–1187, jul 2006. 71

[10] Florence Bertails, Sunil Hadap, Marie-Paule Cani, Ming Lin, Tae-Yong Kim, Steve Marschner, Kelly Ward, and Zoran Kačić-Alesić. Realistic hair simulation: animation and rendering. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pages 89:1–89:154, New York, NY, USA, 2008. ACM. 71

[11] Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. Multi-scale capture of facial geometry and motion. *ACM Trans. Graph.*, **26**[3], jul 2007. 18

[12] Jules Bloomenthal and Brian Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., 1997. 57

[13] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, **29**:104:1–104:10, July 2010. 16

[14] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, **29**[6]:166:1–166:10, dec 2010. 10, 11, 34

[15] Eddy Boxerman and Uri Ascher. Decomposing cloth. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 153–161, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. 71

[16] David Breen, Ron Fedkiw, Ken Museth, Stanley Osher, Guillermo Sapiro, and Ross Whitaker. Level set and pde methods for computer graphics. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, New York, NY, USA, 2004. ACM. 18

[17] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 sketches*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. 10

[18] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, **10**:350–355, 1978. 55

[19] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive procedural street modeling. *ACM Trans. Graph.*, **27**[3]:103:1–103:10, aug 2008. 16

[20] Han Cho. Dressing and modeling food. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 7–21, New York, NY, USA, 2007. ACM. 3, 23, 88

[21] Ming Chuang and Michael Kazhdan. Interactive and anisotropic geometry processing using the screened poisson equation. *ACM Trans. Graph.*, **30**[4]:57:1–57:10, jul 2011. 18

[22] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, **5**[1]:51–72, jan 1986. 3, 5, 6, 7, 8, 11, 27, 53

[23] Ketan Dalal, Allison W. Klein, Yunjun Liu, and Kaleigh Smith. A spectral approach to npr packing. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 71–78, 2006. 12, 21

[24] Agnès Desolneux, Lionel Moisan, and Jean-Michel Morel. Meaningful alignments. *Int. J. Comput. Vision*, **40**[1]:7–23, oct 2000. 19

[25] Gianpiero di Blasi and Giovanni Gallo. Artificial mosaics. *The Visual Computer*, **21**[6]:373–383, 2005. 21

[26] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. *SIGGRAPH Comput. Graph.*, **19**[3]:69–78, jul 1985. 15

[27] Jean-Michel Dischler, Karl Maritaud, Bruno Levy, and Djamchid Ghazanfarpour. Texture particles. *Comput. Graph. Forum*, **21**[3]:401–410, 2002. 19

[28] Daniel Dunbar and Greg Humphreys. A spatial data structure for fast poisson-disk sample generation. *ACM Transactions on Graphics*, **25**[3]:503–508, jul 2006. vii, 9, 10, 34, 53

[29] Mohamed S. Ebeida, Andrew A. Davidson, Anjul Patney, Patrick M. Knupp, Scott A. Mitchell, and John D. Owens. Efficient maximal poisson-disk sampling. *ACM Trans. Graph.*, **30**[4]:49:1–49:12, jul 2011. 10

[30] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. *Texturing and Modeling, Third Edition: A Procedural Approach (The Morgan Kaufmann Series in Computer Graphics).* Morgan Kaufmann, 2002. 15, 48

[31] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision*, pages 1033–1038, 1999. 18

[32] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. 18

[33] Raanan Fattal. Blue-noise point sampling using kernel density model. *ACM Trans. Graph.*, **30**[4]:48:1–48:12, jul 2011. vii, 8, 13, 14

[34] Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 3d collage: expressive non-realistic modeling. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 7–14, 2007. vii, 21, 22, 23, 24

[35] Manuel N. Gamito and Steve C. Maddock. Accurate multidimensional poisson-disk sampling. *ACM Trans. Graph.*, **29**[1]:8:1–8:19, dec 2009. 10

[36] R. Geist and R. Reynolds. Colored noise inversion in digital halftoning. In *Proceedings on Graphics interface '90*, pages 31–38, Toronto, Ont., Canada, Canada, 1990. Canadian Information Processing Society. 15

[37] Robert Geist, Robert Reynolds, and Darrell Suggs. A markovian framework for digital halftoning. *ACM Trans. Graph.*, **12**[2]:136–159, apr 1993. 15

[38] U. Grenander and M. I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society*, **56**:549–603, 1994. 13

[39] Branko Gruenbaum and G. C. Shephard. *Tilings and Patterns.* W H Freeman & Co., 1986. 21, 87

[40] Rajeev Gupta and Nadia Magnenat-Thalmann. Interactive rendering of optical effects in wet hair. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, VRST '07, pages 133–140, New York, NY, USA, 2007. ACM. 71

[41] Alejo 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. 21

[42] Stefan Hiller, Heino Hellwig, and Oliver Deussen. Beyond stippling - methods for distributing objects on the plane. *Comput. Graph. Forum*, **22**[3]:515–522, 2003. vii, 13, 14, 21, 24

[43] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, **26**[2]:71–78, jul 1992. 73

[44] Shu-Wei Hsu and John Keyser. Piles of objects. *ACM Trans. Graph.*, **29**[6]:155:1–155:6, dec 2010. 23, 88

[45] T. Hurtut, P.-E. Landes, J. Thollot, Y. Gousseau, R. Drouillhet, and J.-F. Coeurjolly. Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '09, pages 51–60, New York, NY, USA, 2009. ACM. vii, 19

[46] Takashi Ijiri, Radomir Mech, Takeo Igarashi, and Gavin S. P. Miller. An example-based procedural system for element arrangement. *Comput. Graph. Forum*, **27**[2]:429–436, 2008. 19

[47] Piti Irawan and Steve Marschner. Specular reflection from woven cloth. *ACM Trans. Graph.*, **31**[1]:11:1–11:20, feb 2012. 71

[48] Takayuki Itoh, Kazunori Miyata, and Kenji Shimada. Generating organic textures with controlled anisotropy and directionality. *IEEE Comput. Graph. Appl.*, **23**[3]:38–45, may 2003. 16

[49] Thouis R. Jones. Efficient generation of poisson-disk sampling patterns. *journal of graphics, gpu, and game tools*, **11**[2]:27–36, 2006. vii, 9

[50] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.*, **23**[3]:271–280, jul 1989. 71

[51] Jonathan M. Kaldor, Doug L. James, and Steve Marschner. Simulating knitted cloth at the yarn level. *ACM Trans. Graph.*, **27**[3]:65:1–65:9, aug 2008. 71

[52] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, **31**[4]:55:1–55:11, jul 2012. 18, 88

[53] Craig S. Kaplan and David H. Salesin. Escherization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 499–510, 2000. 21

[54] Craig S. Kaplan and David H. Salesin. Dihedral escherization. In *Proceedings of Graphics Interface 2004*, pages 255–262, 2004. 23

[55] Junhwan Kim and Fabio Pellacini. Jigsaw image mosaics. *ACM Transactions on Graphics*, **21**:657–664, 2002. 12, 21

[56] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph.*, **31**[4]:54:1–54:11, jul 2012. 18, 88

[57] Naoki Kita and Kazunori Miyata. Interactive procedural modeling of pebble mosaics. In *SIGGRAPH Asia 2011 Sketches*, SA '11, pages 35:1–35:2, New York, NY, USA, 2011. ACM. 18

[58] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. Recursive wang tiles for real-time blue noise. *ACM Trans. Graph.*, **25**[3]:509–518, jul 2006. 7

[59] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics*, **22**[3]:277–286, jul 2003. 19

[60] Ares Lagae and Philip Dutré. A procedural object distribution function. *ACM Trans. Graph.*, **24**[4]:1442–1461, oct 2005. vii, 6, 7, 21, 22, 23, 24, 27

[61] Ares Lagae and Philip Dutré. An alternative for wang tiles: colored edges versus colored corners. *ACM Trans. Graph.*, **25**[4]:1442–1459, oct 2006. 7

[62] Ares Lagae and Philip Dutré. A comparison of methods for generating Poisson disk distributions. *Computer Graphics Forum*, **27**[1]:114–129, March 2008. 31

[63] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Trans. Graph.*, **24**[3]:777–786, jul 2005. 18, 20

[64] Bruno Lévy and Yang Liu. Lp centroidal voronoi tessellation and its applications. *ACM Trans. Graph.*, **29**[4]:119:1–119:11, jul 2010. 12

[65] Hongwei Li, Li-Yi Wei, Pedro V. Sander, and Chi-Wing Fu. Anisotropic blue noise sampling. *ACM Trans. Graph.*, **29**[6]:167, 2010. 27

[66] Fuchang Liu, Takahiro Harada, Youngeun Lee, and Young J. Kim. Real-time collision culling of a million bodies on graphics processing units. *ACM Trans. Graph.*, **29**[6]:154:1–154:8, dec 2010. 23

[67] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Transactions on Graphics, volume*, **25**[3]:681–689, jul 2006. 50

[68] Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On centroidal voronoi tessellation energy smoothness and fast computation. *ACM Trans. Graph.*, **28**[4]:101:1–101:17, sep 2009. 12

[69] Yiming Liu, Jiaping Wang, Su Xue, Xin Tong, Sing Bing Kang, and Baining Guo. Texture splicing. *Comput. Graph. Forum*, **28**[7]:1907–1915, 2009. 20

[70] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, **28**:129–137, 1982. 12

[71] Chongyang Ma, Li-Yi Wei, and Xin Tong. Discrete element textures. *ACM Transactions on Graphics*, **30**[4]:62:1–62:10, aug 2011. vii, 20, 23, 24, 25, 47, 50, 59, 60, 86, 88

[72] Kazuo Matsufuji, Naoki Kawai, Eriko Kimura, Kazunori Miyata, and Kaisei Sakurai. Tokyo koukai 2009–031911. *JP patent*, 2009. 18

[73] Kazuo Matsufuji, Naoki Kawai, Eriko Kimura, Kazunori Miyata, and Kaisei Sakurai. Tokyo koukai 2009–116725. *JP patent*, 2009. 18

[74] Kazuo Matsufuji, Naoki Kawai, Eriko Kimura, Kazunori Miyata, and Kaisei Sakurai. Tokyo koukai 2010–39662. *JP patent*, 2010. 18

[75] Michael McCool and Eugene Fiume. Hierarchical poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface '92*, pages 94–105, 1992. 8, 12

[76] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH Comput. Graph.*, **25**[4]:157–164, jul 1991. 8

[77] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, **16**[4]:647–668, aug 1987. 11

[78] Kazunori Miyata. A method of generating stone wall patterns. *SIGGRAPH Comput. Graph.*, **24**[4]:387–394, sep 1990. 16

[79] Kazunori Miyata, Takayuki Itoh, and Kenji Shimada. A method for generating pavement textures using the square packing technique. *The Visual Computer*, **17**[8]:475–490, 2001. 20

[80] Matthias Müller, Jos Stam, Doug James, and Nils Thürey. Real time physics: class notes. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pages 88:1–88:90, New York, NY, USA, 2008. ACM. 23

[81] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, **25**[3]:614–623, jul 2006. 16

[82] Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, **26**[3], jul 2007. 16

[83] Jeff Orchard and Craig S. Kaplan. Cut-out image mosaics. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, NPAR '08, pages 79–87, New York, NY, USA, 2008. ACM. 21

[84] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics*, **21**[4]:807–832, oct 2002. 11, 55

[85] Victor Ostromoukhov. Sampling with polyominoes. *ACM Trans. Graph.*, **26**[3], jul 2007. 12, 34

[86] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph.*, **23**[3]:488–495, aug 2004. 11

[87] Oktar Ozgen, Marcelo Kallmann, Lynnette Es Ramirez, and Carlos Fm Coimbra. Underwater cloth simulation with fractional derivatives. *ACM Trans. Graph.*, **29**[3]:23:1–23:9, jul 2010. 71

[88] Sylvain Paris, Hector M. Briceño, and François X. Sillion. Capture of hair geometry from multiple images. *ACM Trans. Graph.*, **23**[3]:712–719, aug 2004. 18

[89] Sylvain Paris, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. Hair photobooth: geometric and photometric acquisition of real hairstyles. *ACM Trans. Graph.*, **27**[3]:30:1–30:9, aug 2008. 71

[90] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001. ACM. 16

[91] Ken Perlin. An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 287–296, 1985. vii, 15, 16

[92] Adrien Peytavie, Eric Galin, Stephane Merillou, and Jerome Grosjean. Procedural generation of rock piles using aperiodic tiling. *Computer Graphics Forum*, **28**:1801–1809, 2009. vii, 17

[93] Helmut Pottmann, Yang Liu, Johannes Wallner, Alexander Bobenko, and Wenping Wang. Geometry of multi-layer freeform structures for architecture. *ACM Transactions on Graphics*, **26**[3], jul 2007. 50

[94] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, 2000. 18

[95] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 1988. 57, 58

[96] Ravi Ramamoorthi, John Anderson, Mark Meyer, and Derek Nowrouzezahrai. A theory of monte carlo visibility sampling. *ACM Trans. Graph.*, **31**[5]:121:1–121:16, sep 2012. 6

[97] Ganesh Ramanarayanan, Kavita Bala, and James A. Ferwerda. Perception of complex aggregates. *ACM Trans. Graph.*, **27**[3]:60:1–60:10, aug 2008. 27

[98] A H Robinson. Multidimensional fourier transforms and image processing with finite scanning apertures. *Appl Opt*, **12**[10]:2344–52, 1973. 15

[99] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph.*, **29**[6]:157:1–157:8, dec 2010. 71

[100] Kaisei Sakurai and Kazuo Matsufuji. A procedural modeling of woven textiles with fuzz. In *ACM SIGGRAPH ASIA 2009 Posters*, SIGGRAPH ASIA '09, pages 58:1–58:1, New York, NY, USA, 2009. ACM. 6, 71

[101] Kaisei Sakurai, Kazuo Matsufuji, and Yu Okamoto. Tokyo koukai 2010–287204. *JP patent*, 2010. 18

[102] Kaisei Sakurai, Kazuo Matsufuji, and Yu Okamoto. Tokyo koukai 2011–148184. *JP patent*, 2011. 18

[103] Kaisei Sakurai and Kazunori Miyata. Procedural modeling of multiple rocks piled on flat ground. In *ACM SIGGRAPH ASIA 2010 Posters*, SA '10, pages 35:1–35:2, New York, NY, USA, 2010. ACM. 17

[104] Kaisei Sakurai and Kazunori Miyata. Generating layout of nonperiodic aggregates. In *Proceedings of Nicograph International 2012*, pages 68–75, 2012. 27

[105] Andrew Selle, Michael Lentine, and Ronald Fedkiw. A mass spring model for hair simulation. *ACM Trans. Graph.*, **27**[3]:64:1–64:11, aug 2008. 71

[106] Kaleigh Smith, Yunjun Liu, and Allison Klein. Animosaics. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 201–208, 2005. 21

[107] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. Stress relief: improving structural strength of 3d printable objects. *ACM Trans. Graph.*, **31**[4]:48:1–48:11, jul 2012. 88

[108] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, **24**[3]:553–560, jul 2005. 11

[109] Kenshi Takayama, Makoto Okabe, Takashi Ijiri, and Takeo Igarashi. Lapped solid textures: filling a model with anisotropic textures. *ACM Trans. Graph.*, **27**[3]:53:1–53:9, aug 2008. 18, 88

[110] Kenshi Takayama, Olga Sorkine, Andrew Nealen, and Takeo Igarashi. Volumetric modeling with diffusion surfaces. *ACM Trans. Graph.*, **29**[6]:180:1–180:8, dec 2010. 18, 88

[111] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, pages 63–74, New York, NY, USA, 2012. ACM. 18, 88

[112] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, **30**[2]:11:1–11:14, apr 2011. 16

[113] Robert A. Ulichney. Dithering with blue noise. *Proceedings of the IEEE*, **76**:56–79, 1988. 15

[114] Luiz Velho and Jonas de Miranda Gomes. Digital halftoning with space filling curves. *SIGGRAPH Comput. Graph.*, **25**[4]:81–90, jul 1991. 15

[115] H. Wang, American Telephone, and Telegraph Company. *Proving Theorems by Pattern Recognition -II.* American Telephone and Telegraph Company, 1961. 7

[116] Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. Example-based hair geometry synthesis. *ACM Trans. Graph.*, **28**[3]:56:1–56:9, jul 2009. 71

[117] Li-Yi Wei. Parallel poisson disk sampling. *ACM Trans. Graph.*, **27**[3]:20:1–20:9, aug 2008. 10, 11, 34, 56

[118] Li-Yi Wei. Multi-class blue noise sampling. *ACM Trans. Graph.*, **29**:79:1–79:8, July 2010. vii, 8, 9, 69

[119] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report*, 2009. 18

[120] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 18

[121] Li-Yi Wei and Rui Wang. Differential domain analysis for non-uniform sampling. *ACM Trans. Graph.*, **30**[4]:50:1–50:10, jul 2011. 15

[122] Emily Whiting, John Ochsendorf, and Frédo Durand. Procedural modeling of structurally-sound masonry buildings. *ACM Trans. Graph.*, **28**[5]:112:1–112:9, dec 2009. 16

[123] Steven Worley. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294, 1996. vii, 16, 17

[124] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics*, **23**[3]:364–367, aug 2004. 19

[125] Ying Xiang, Shi-Qing Xin, Qian Sun, and Ying He. Parallel and accurate poisson disk sampling on arbitrary surfaces. In *SIGGRAPH Asia 2011 Sketches*, 2011. 11, 34, 54

[126] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: set evolution for inspiring 3d shape galleries. *ACM Trans. Graph.*, **31**[4]:57:1–57:10, jul 2012. 18, 88

[127] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D. Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Trans. Graph.*, **31**[4]:56:1–56:11, jul 2012. 18, 88

[128] Cem Yuksel, Scott Schaefer, and John Keyser. Hair meshes. *ACM Trans. Graph.*, **28**[5]:166:1–166:7, dec 2009. 71

[129] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector field design on surfaces. *ACM Transactions on Graphics*, **25**[4]:1294–1326, oct 2006. 69

[130] Yahan Zhou, Haibin Huang, Li-Yi Wei, and Rui Wang. Point sampling with general noise spectrum. *ACM Trans. Graph.*, **31**[4]:76:1–76:11, jul 2012. 88