NICOGRAPH International 20xx, pp. 1234-1237

Generating Layout of Nonperiodic Aggregates

Kaisei Sakurai **Kazunori Miyata**

JAIST

{sakurai, miyata} @ jaist.ac.jp

Abstract

We propose a method for modeling nonperiodic aggregates composed of arbitrary elements on a 2-D plane. The method effectively generates a dense aggregate from the elements to be arranged. To date, nonperiodic aggregates were generated using a Poisson disk distribution; however, the approach results in large gaps among the elements. Dense aggregates can be generated using a dart-throwing method to arbitrary elements; however, this approach is time consuming. To reduce the calculation time, we propose a method that fills in elements in the gaps among the already placed elements. To effectively fill these gaps, our method quantifies the gaps and finds additional positions for elements. In our experiment, we show that our method is more effective than the dart-throwing method. Also, to confirm that our method is generic, we show examples using two- and three-dimensional elements.

1. Introduction

Nonperiodic aggregates composed of elements often appear in daily life or natural phenomena, e.g., objects on a table and the cristae on a leather-textured surface. Aggregates are often represented by computer graphics. However, it is tedious to create and manually control the arrangement of the aggregate elements. To reduce the time-consuming work, we propose a method to generate a layout of the aggregate elements.

Aggregates in daily life or natural phenomena have periodic (lattice) or nonperiodic arrangements. In this study, we consider only nonperiodic arrangement because it is easy to create periodic arrangements by placing elements on a grid. Our goal is to effectively generate nonperiodic aggregates composed of arbitrary elements on a 2-D plane.

To achieve this goal, our method refers to the dart-throwing method [1]. The dart-throwing method is used for Poisson disk sampling that 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 [2]. We apply the dart-throwing method to distribute elements randomly. To control the layout, our method uses arbitrary-shaped regions we call exclusive regions, instead of circular regions. In our method, the positions of the elements are not limited to blue noise properties, which uniformly and randomly distributed points have. This is because the exclusive region can be specified as something other than a circle.

(a) Previous method (16 elements) (b) Our method (224 elements)

Figure 1 Comparison of results

To date, several methods use Poisson disk distribution for placing elements [3, 4]. The previous methods can generate various aggregates composed of elements having isotropic shapes, such as a sphere or cylinder. However, the method cannot generate dense aggregates composed of elements that are concave or of different sizes because the large gaps among the elements are produced as shown in Fig. 1 (a). For generating a dense aggregate, we consider following two approaches. (1) One method places the elements while checking for overlaps of elements, similar to the naive dart-throwing method that uses arbitrary-shaped exclusive regions. This approach can generate a dense aggregate composed of many elements. However, the calculation cost of the naive dart-throwing method is too expensive because of its repetitive placement trials. (2) Another approach places elements using collision and reaction in a physical simulation. This approach also generates dense aggregates, however, it may also generate periodic arrangements by using isotropic shapes. To avoid the periodic arrangements, we employ the former approach, despite its time consuming nature. We assume that it is effective to iteratively fill the gaps among placed elements in order to reduce redundant

cancelations. In our experiment, we effectively generate an aggregate that includes only a few gaps, as shown in Fig. 1 (b). Our experiment demonstrates that the approach is more effective than the naive dart-throwing method that iteratively places elements while checking overlaps of exclusive regions.

2. Related works

To date, several methods are used for generating aggregates. Aggregate generation techniques fall roughly into three categories: physical simulation, procedural modeling, and example-based modeling.

There are some methods for generating aggregates that consist of a large number of objects using physical simulation [5, 6, 7]. Also, for ease of control, methods are proposed for generating desired results such as desired piles [8]. However, these methods may also generate periodic arrangements by using elements of isotropic shapes. In addition, each element is unstable in the simulation. That is, it is difficult to determine the termination of simulation. Though the simulation is appropriate to represent dynamics, it is inappropriate to determine the layout of nonperiodic arrangements.

Procedural modeling methods generate an aggregate or volume texture with some parameters. They involve noise-based texturing functions and generation methods for specific textures [9]. Noise-based texturing functions are widely used to represent natural aggregates that have various element shapes, such as clouds and rocks [10, 11, 12]. Specific elements are easily controlled by parameters [13, 14, 15]. However, by looking at the parameters it is difficult for a user to determine what kind of aggregate will be generated. Furthermore, it is difficult to design a new aggregate entirely by the procedural modeling methods.

Example-based modeling uses exemplars that designers draw and edit, and it is effective for generating large-scale aggregates. Many methods are proposed for generating textures. Our methods employ the same approach as example-based modeling to generate an aggregate composed of elements.

Example-based modeling methods include placement and synthesis approaches. Among the placement approaches, mosaic generation methods [16, 17, 18, 19], the element distribution method [3], and tiling methods [20, 21] generate an intended texture by controlling layouts of user-specified elements. We employed this approach because our method controls densities of arbitrary elements. The mosaic generation methods pack elements into containers. These methods cannot control the density of placed elements. Controlling the density of elements is important for changing appearance of an aggregate or texture. The element distribution method [3] is designed to control densities of aggregates on a 2-D plane by specifying parameters for the Poisson disk distribution. However, this method is not adequate for distributing arbitrary-shaped elements. Tiling methods determine the position and orientation of each element on the basis of the knowledge of patterns [22]. These methods pack elements into specified regions without controlling the position and orientation of the elements; the results are deterministically generated. However, the results have artifacts, such as periodic arrangements. Our method prevents the occurrence of periodic patterns by using random positioning.

The synthesis methods that generate an aggregate or texture from an exemplar, which is a small patch or sample of an aggregate or texture, is called texture synthesis. The results significantly depend on the exemplar that is used. Two approaches have been proposed for texture synthesis: pixel-based and patch-based [23]. Pixel-based approaches do not consider discrete elements, whereas patch-based approaches do consider them. Pixel-based approaches synthesize a texture by finding and copying pixels that have the most similar local neighborhoods [24, 25, 26, 27]. The methods do not maintain boundaries of elements in the exemplars while generating textures. A number of patch-based methods that use discrete elements have been proposed [28, 29, 30, 31, 32, 33, 34, 35]. These patch-based methods synthesize a texture from exemplars that include discrete elements and their relative distances. Both the pixel-based and patch-based approaches effectively generate large textures; however, it is hard to edit the position and orientation of elements in the exemplars. In contrast, our method does not need to input the exemplar, and it generates aggregates from elements.

To easily control densities and use arbitrary shaped elements, we adopt a placement approach for generating a layout of nonperiodic aggregates.

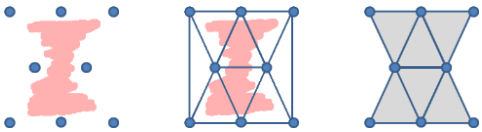
3. Preliminary experimentation

As a preliminary experiment, we tried to determine the layout of elements by the naive dart-throwing method using arbitrary-shaped exclusive regions. In the preliminary experiment, we confirmed the effects and drawbacks of the naive dart-throwing method.

3.1 Exclusive region

Our method considers placing exclusive regions only on a 2-D plane. The exclusive region is independent of the shapes of the elements, and it determines the distance between an element and each of its neighbors. The position, orientation, and size of the exclusive regions are specified by transformation matrices, and the elements are attached to the exclusive regions.

The exclusive regions are defined by 2-D triangular meshes. By using meshes, calculation of the transformation and checking overlaps can be accelerated using a graphics processing unit (GPU). A user specifies the vertices of the triangular meshes and the inner regions of the shapes, as shown in Fig. 2(a). Through a Delaunay triangulation, a triangular mesh is used to construct a convex hull from the vertices, as shown in Fig. 2(b). To construct a concave mesh, the triangular mesh is extracted from the inner regions, as shown in Fig. 2(c). Aggregate



(a) Input (b) Triangulation (c) Exclusive region

Figure 2 Specification of an exclusive region.

coordinates are specified simultaneously when constructing a mesh.

The center of transformation is the center of the bounding box of an exclusive region. Transformation is determined by Eq. 1:

$$\mathbf{x} = (\mathbf{RS}+\mathbf{T})\,\mathbf{x}_0 \qquad (1)$$

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

3.2 Procedure of the naive dart-throwing method

The steps in the naive dart-throwing method are as follows:

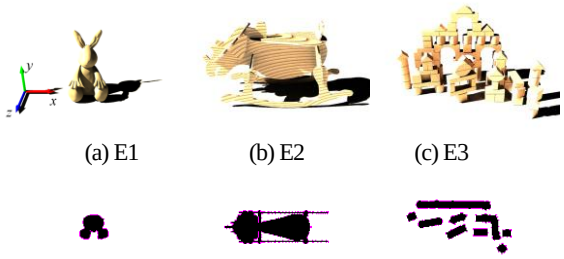
(S1) Randomly rotate and place an exclusive region.
(S2) Detect collisions of the region and the placed regions.
(S3) Cancel placement in case of overlaps, and repeat step S1 until the termination conditions are satisfied.

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

In the dart-throwing method for the Poisson disk distribution, we can roughly estimate the number of placed points from the sizes of placed circles [36]. 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. Therefore, we use the number of consecutive cancelations as the termination condition.

3.3 Results of preliminary experimentation

The naive dart-throwing method was implemented on a Windows PC with Intel Core i7, 3.07 GHz CPU, and 12.0 GB RAM. We use elements, E1, E2, and E3 as shown in Fig. 3 (a)–(c). The exclusive regions are created by the projection into the XZ-plane in 3-D as shown in Fig. 3 (d)–(e). Figure 4 shows the result of the naive



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

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

Figure 3 Elements and exclusive regions. Image (a)–(c) are elements and (d) – (f) are their exclusive regions. A black region denotes an inner region, and a magenta pixel indicates a vertex in (d)–(f).

(a) Placed exclusive regions (b) Placed elements

Figure 4 Result of the naive dart-throwing method (210 elements).

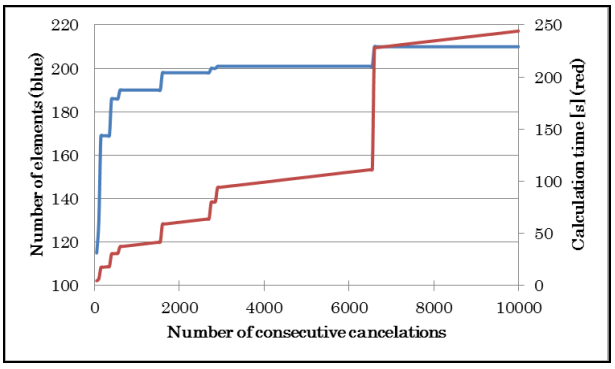
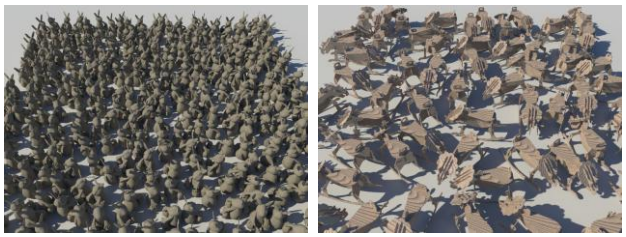


Figure 5 Relation of Number of consecutive cancelations, Number of elements, and Calculation time of Fig. 3.

dart-throwing method whose termination condition is specified as 10000 consecutive cancelations. Here, we demonstrate the generation of the aggregate without scaling. Figure 4 (a) shows the placed exclusive regions, and their transformation matrixes are applied to the elements as shown in Fig. 4 (b). The dart-throwing method generates nonperiodic dense aggregates without overlapping of the elements on the XZ-plane.

Figure 5 shows the number of placed elements and the calculation time of every termination condition for generating Fig. 3. The graphs are plotted for each 50 consecutive cancelations. Overall, a positive correlation appears in the graph. The number of elements (blue line) dramatically increases until approximately 2000 consecutive cancelations, and then it gently increases. However, it is unexpected when it stops increasing. For example, for approximately 7000



(a) E1 (b) E2

(c) E3

Figure 6 Results of each kind of three kinds of elements.

consecutive cancelations, the calculation time (red line) dramatically increases while only nine elements are added. The calculation time of the naive dart-throwing method depends on the position and orientation of the elements, which are randomly determined. The method is not robust and the calculation cost is high. As another example, Fig.6 shows the results of distributing elements E1, E2, and E3. Figure 7 confirms that the calculation costs are in the same range as in Fig. 5.

To reduce calculation time, we refer to the dart-throwing method for producing uniform points. There are two approaches for placing points: parallelization [37, 38, 39] and specification of regions [40, 41]. Parallelization approaches are only used for points having blue-noise properties. We cannot use these methods because our nonperiodic arrangements do not have blue noise properties. However, the specification approaches reduce the number of placement cancelations.

4. Fill-gap method

We assume that having many placement cancelations increases the calculation cost. Thus, to reduce this cost, we prepared the set of candidate points P where the exclusive region could be placed. In the preliminary experiment (Section 3), we shortened the calculation time by setting the termination condition to less than 100 consecutive cancelations. From this result, our method specified the number of P to 100, thereby limiting the number of consecutive cancelations to 100. To avoid overlaps of exclusive regions as much 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 elements as describe in Section 3.1. Our method consists of two steps:

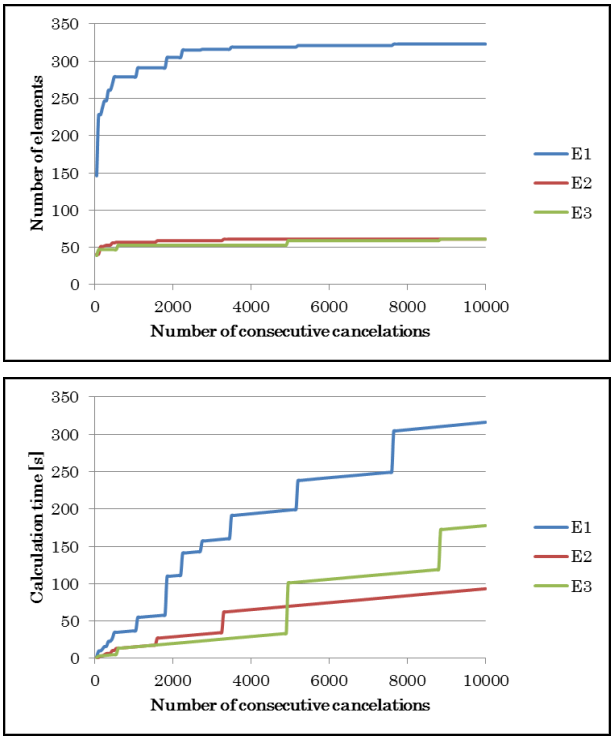


Figure 7 Relationship between the number of consecutive cancelations, number of elements, and the calculation time of Fig. 6.

(a) Step 1 (b) Step 2 (1 iter) (c) Step 2 (termination)

Figure 8 Overview.

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

Figure 8 shows an illustration of the two steps. In Fig. 8, black regions indicate exclusive regions and orange points indicate instances of P.

Step 1 is the initial placement resulting in gaps. The step ensures that there is no overlap of exclusive regions by Poisson disk distribution. Step 2 iteratively fills the gaps among the placed exclusive regions. This step specifies positions for P.

In Step 2, the cancelations of placements occur frequently when P and the boundaries of placed exclusive regions are too close. In our experiment, we specify the termination condition to be 200 consecutive placement cancelations.

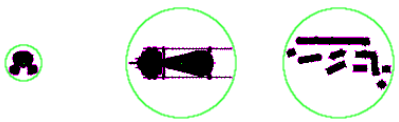


Figure 9 Circular region *O*. The green circle indicates the circular region's boundary for each element.

4.2. Sparse placement

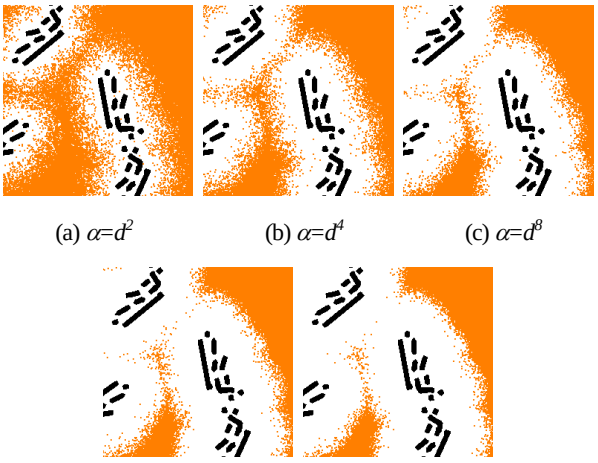
Step 1 sparsely places exclusive regions at the points distributed by the dart-throwing method for Poisson disk distribution. The Poisson disk distribution ensures that there is no overlap of circular regions. By specifying a circular region *O* involving an exclusive region, nonperiodic arrangements can be quickly calculated with no overlaps of exclusive regions. The value of **T** is determined by the position of a placed point. The values of **R** and **S** refer to a random angle and scale in specified range. The dart-throwing method uses an exclusive circular region *O* for point distribution. The center of *O* is located at the placed point. Here, the largest circle is specified for Poisson disk distribution, so the radius of *O* is determined by the distance between the center and the farthest point on the boundary as shown in Fig. 9.

4.3. Placement in gaps

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. To avoid concentration of *Q* at few positions, *Q* points 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 for quickly calculating 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. By using a normalized distance field, the method can place exclusive regions of any size in the same manner. Also, 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.

We determine 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 10 shows the comparisons of points of different multipliers n. The result shows that the 16th power yields acceptable results. The points *Q* placed between 16th and 32th powers are similarly distributed. We deduce that it is enough to use the 16th power of normalized distances for the probability α.

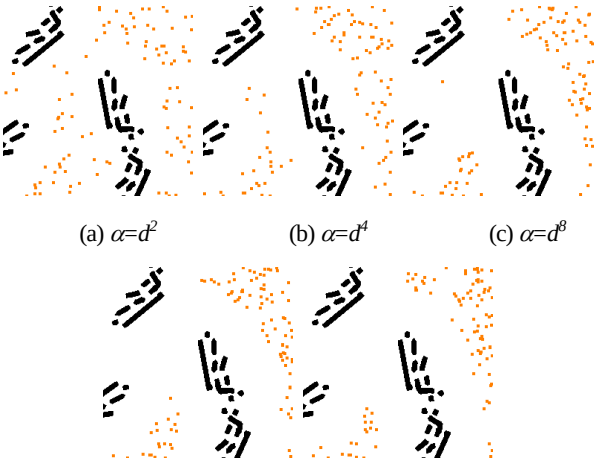
The next step is to choose 100 points *P* from the placed points *Q*. Figure 11 shows comparisons of points *P* taken from points *Q* for each probability α. Points *P* chosen from $\alpha=d^{16}$ and $\alpha=d^{32}$ are similarly distributed in positions farther from the boundaries of placed exclusive regions. The results shows that it is sufficient to use the 16th power for probability α.



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

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

Figure 10 Comparisons of yielded points *Q* (orange).



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

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

Figure 11 Comparisons of points *P* (orange) chosen from Fig. 10.

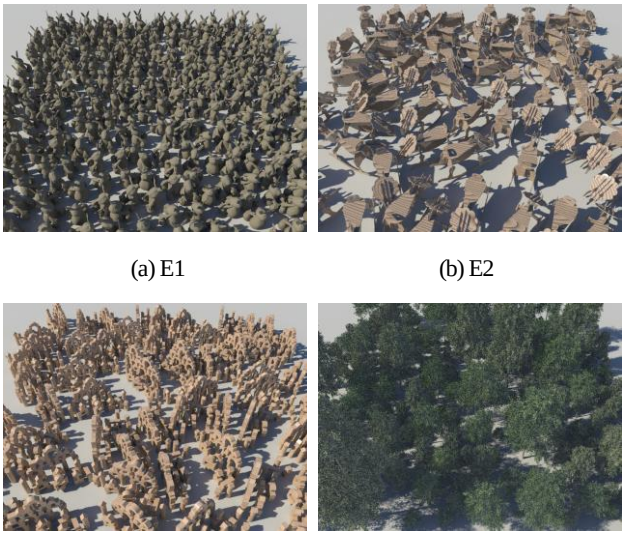
5. Results

In this section, we discuss the results of our method. Sections 5.1 and 5.2 show the calculation times of our method and examples of changing exclusive regions. Section 5.3 shows variations of the generated aggregates.

5.1. Calculation time

Our method implemented on a Windows PC with Intel Core i7, 3.07 GHz CPU, and 12.0 GB RAM. Figure 1 (b) which is an aggregate having three kinds of elements is calculated in 18.3 s, and includes 224 elements. Figure 12 shows placed exclusive regions. The naive

Figure 12 Exclusive regions placed by our method.



(a) E1 (b) E2

(c) E3 (d) Trees

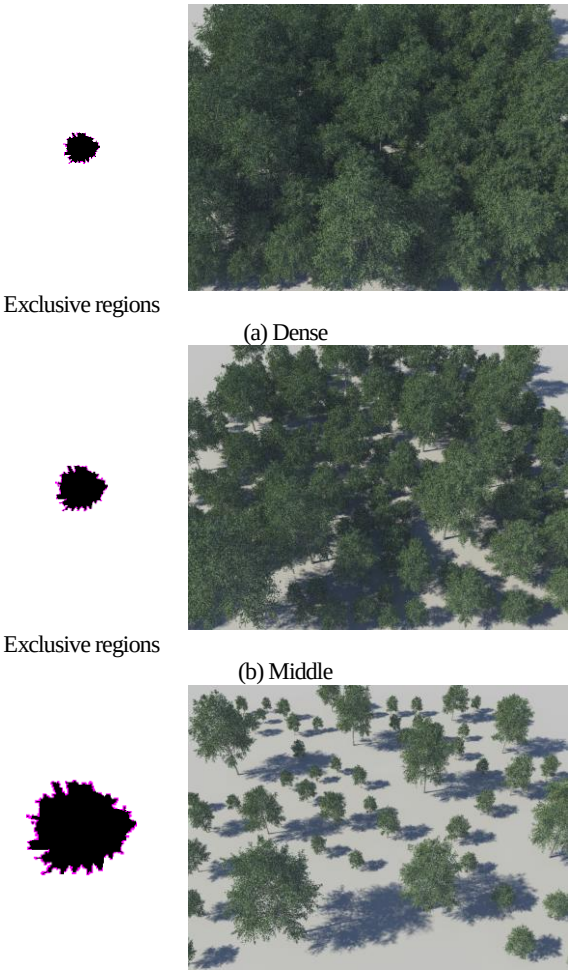
Figure 13 Generated aggregates.

Table 1 Comparison of calculation time between our method and the naive dart-throwing method

	Our method		Naive method	
	Time [s]	Num. of element	Time [s]	Num. of element
Fig. 13 (a)	17.2	321	316.2	323
Fig. 13 (b)	11.5	64	93.3	61
Fig. 13 (c)	13.2	58	177.8	61
Fig. 13 (d)	31.5	244	488.0	258

dart-throwing method needs approximately 250 s for generating an aggregate composed of 210 elements as shown in Fig. 3. By way of comparison between the number of placed elements by our method and the naive dart-throwing method, our method placed more 14 elements. That is, our method can place a sufficient number of elements.

The other results have the same trend with the above as shown in Fig. 13. Table 1 shows the comparison of calculation times between our method and the naive dart-throwing method. Figure 13 (d) is composed of two types of trees with scaling range from 0.2 to 1.0.



(a) Dense

(b) Middle

(c) Sparse

Figure 14 Variations of densities.

5.2. Changing exclusive regions

Our method can control densities of generated aggregates by changing the size of the exclusive regions. Figure 14 shows controlled densities of aggregates accomplished by changing exclusive regions. Smaller and larger exclusive regions create denser or sparser aggregates, respectively.

5.3. Examples

Our method places arbitrary elements not only in 2-D, but also in 3-D. Figure 15 shows examples of placing 2-D and 3-D elements. The method can also be used to generate height maps. Figure 16 shows the 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 our method is a generic method for generating aggregates.

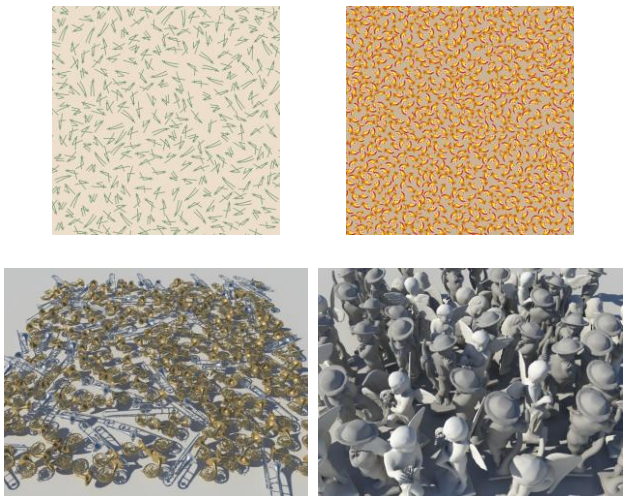


Figure 15 Variations of examples.

6. Conclusion

We proposed a procedure for determining the layout of nonperiodic aggregates composed of arbitrary elements. Our method was implemented based on the dart-throwing method using exclusive regions that prohibit each element from overlapping one another. Our approach was to fill the gaps among placed exclusive regions. Our method is also dramatically faster than the naive dart-throwing method. We have illustrated few representative examples of nonperiodic textures in the figures.

Acknowledgments

This work was supported by a Grant-in-Aid for Scientific Research on Innovative Areas (No. 23135513) from the Ministry of Education, Science, Culture, Sports and Science, Japan.

References

[1] Cook R. L., Stochastic sampling in computer graphics, ACM Transactions on Graphics, vol. 5, 1, 1986

[2] Li H., Wei L-Y, Sander P. V., Fu C-W, Anisotropic blue noise sampling, ACM. Transactions on Graphics, vol. 29, 6, pp. 167:1-167:12, 2010

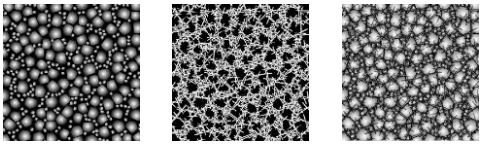
[3] Lagae A., Dutre P., A procedural object distribution function, ACM Transactions on Graphics, vol. 24, pp. 1442–1461, 2005

[4] Ramanarayanan G., Bala K., Ferwerda J. A., Perception of complex aggregates, ACM Transactions on Graphics, vol. 27, 3, pp. 60:1-60:10, 2008

[5] Muller M., Stam J., James D., Thurey N., Real time physics: class notes, ACM SIGGRAPH 2008 classes, pp. 88:1--88:90, 2008

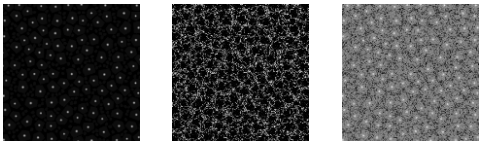
[6] Liu F., Harada T, Lee Y., Kim Y. J., Real-time collision culling of a million bodies on graphics processing units, ACM Transactions on Graphics, vol. 29, 6, pp154:1-154:8, 2010

[7] Hadap S., Eberle D., Volino P., Lin M. C., Redon S., Ericson C.,



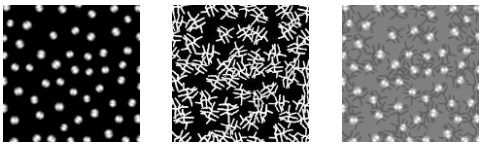
(a) Cristae #1 (b) Sulci #1 (c) Synthesis of (a) and (b)

(d) Rendered image of (c)



(e) Cristae #2 (f) Sulci #2 (g) Synthesis of (e) and (f)

(h) Rendered image of (g)



(i) Cristae #3 (j) Sulci #3 (k) Synthesis of (i) and (j)

(l) Rendered image of (k)

Figure 16 Generated height maps of leather textures.

Collision detection and proximity queries, ACM SIGGRAPH 2004 Course Notes, 15, 2004

[8] Hsu S-W., Keyser J., Piles of objects, ACM Transactions on Graphics, vol. 29, 6, pp. 155-115:6, 2010

[9] Ebert D. S., Musgrave F. K., Peachey D., Perlin K., Worley S., Texturing and Modeling: A Procedural Approach, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002

[10] Perlin K., An image synthesizer, Proceedings of the 12th annual conference on Computer graphics and interactive techniques, pp.287-296, 1985

[11] Worley S., A cellular texture basis function, Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp.291-294, 1996

[12] Benes B., Stava O., Mech R., Miller G., Guided procedural modeling, Computer graphics forum, vol. 30, pp. 325-334 2011

[13] Peytavie A., Galin E., Grosjean J., Merillou S., Procedural generation of rock piles using aperiodic tiling, Computer Graphics Forum, vol. 28, 2009

[14] Iben H. N., O'Brien J. F., Generating surface crack patterns, In Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '06, Eurographics Association, pp. 177-185, 2006

[15] Adabala N., Magnenat-Thalmann N., A procedural thread texture model, journal of graphics, gpu, and game tools, vol. 8, 3, pp. 33–40, 2003

[16] Kim J., Pellacini F., Jigsaw image mosaics, ACM Transactions on Graphics, vol. 21, pp. 657–664, 2002

[17] Gal R., Sorkine O., Popa T., Sheffer A., Cohen-Or D., 3D collage: expressive non-realistic modeling, Proceedings of the 5th international symposium on Non-photorealistic animation and rendering, 2007

[18] Smith K., Liu Y., Klein A., Animosaics, SCA '05**:** Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2005

[19] Dalal K., Klein A. W., Liu Y., Smith K., A spectral approach to NPR packing, NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering, 2006

[20] Kaplan C. S., Salesin D. H., Escherization, In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 499–510. 2000

[21] Kaplan C. S., Salesin D, Dihedral Escherization, GI '04 Proceedings of Graphics Interface 2004, 2004

[22] Grünbaum B., Shephard G. C., Tilings and Patterns. W. H. Freeman, 1987.

[23] Wei L.-Y., Lefebvre S., Kwatra V., Turk G., State of the art in example-based texture synthesis. In Eurographics 2009 State of the Art Report, 2009

[24] Efros A. A., Leung T. K., Texture Synthesis by Non-Parametric Sampling, Proceedings of the International Conference on Computer Vision,Vol. 2, pp. 1033-1038, 1999

[25] Wei L-Y, Levoy M., Fast texture synthesis using tree-structured vector quantization, Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 479-488, 2000

[26] Tong X., Zhang J., Liu L., Wang X., Guo B., Shum H.-Y., Synthesis of bidirectional texture functions on arbitrary surfaces, ACM Transactions on Graphics, vol. 21, 3, pp. 665-672, 2002

[27] Ashikhmin M., Synthesizing natural textures, Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 217-226, 2001

[28] Praun E., Finkelstein A., Hoppe H., Lapped textures, Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00), pp. 465-470, 2000

[29] Efros A. A. and Freeman W. T., Image quilting for texture synthesis and transfer, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 341-346, 2001

[30] Dischler J., Maritaud K., Levy B., Ghazanfarpour D., Texture particles, Computer graphics forum, vol. 21, pp. 401-410. 2002

[31] Kwatra V., Schodl A., Essa I., Turk G., Bobick A., Graphcut textures: image and video synthesis using graph cuts, ACM Transactions on Graphics, vol. 22, 3, pp. 277-286, 2003

[32] Wu Q., Yu Y., Feature matching and deformation for texture synthesis, ACM Transactions on Graphics, vol. 23, 3, pp. 364-367, 2004

[33] Hurtut T., Landes P.-E., Thollot J., Gousseau Y., Drouillhet R., Coeurjolly J.-F., Appearance-guided synthesis of element arrangements by example, Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering, 2009

[34] Ijiri T., Mech R., Igarashi T., Miller G., An example-based procedural system for element arrangement, Computer graphics forum, vol. 27, pp. 429-436. 2008

[35] Ma C., Wei L.-Y., Tong X., Discrete element textures, ACM Transactions on Graphics vol. 30, 4, pp. 62:1--62:10, 2011

[36] Lagae, A., Dutre, P., A comparison of methods for generating Poisson disk distributions, Computer Graphics Forum, vol. 27, 1, pp. 114-129, 2008

[37] Wei L.-Y., Parallel Poisson disk sampling, ACM Transactions on Graphics, vol.27, 3, 2008

[38] Bowers J., Wang R., Wei L.-Y., Maletz D., Parallel Poisson disk sampling with spectrum analysis on surfaces, ACM Transactions on Graphics, vol.29 6, 2010

[39] Xiang Y., Xin S.-Q., Sun Q., He Y., Parallel and accurate Poisson disk sampling on arbitrary surfaces, SIGGRAPH Asia 2011 Sketches, pp. 18:1-18:2, 2011

[40] Dunbar D., Greg H., A spatial data structure for fast Poisson disk sample generation, ACM Transactions on Graphics, vol.25, 3, 2006

[41] Ostromoukhov V., Sampling with polyominoes, ACM Transactions on Graphics, vol.26, 3, 2007