An Algorithm for Random Fractal Filling of Space

John Shier[1] and Paul Bourke[2]
Email: paul.bourke@uwa.edu.au

[1]Normandale Community College, Bloomington, Minnesota 55431, USA.
[2]iVEC @ University of Western Australia, 35 Stirling Hwy, Crawley, West Australia 6009, Australia.

Abstract

Computational experiments with a simple algorithm show that it is possible to fill any spatial region with a random fractalization of any shape, with a continuous range of pre-specified fractal dimensions D. The algorithm is presented here in 1, 2, or 3 physical dimensions. The size power-law exponent c or the fractal dimension D can be specified ab initio over a substantial range. The method creates an infinite set of shapes whose areas (lengths, volumes) obey a power law and sum to the area (length, volume) to be filled. The algorithm begins by randomly placing the largest shape and continues using random search to place each smaller shape where it does not overlap or touch any previously placed shape. The resulting gasket is a single connected object.

Categories and Subject Descriptors (according to ACM CCS):G.3 [Mathematics of Computing]: Probability and Statistics--Stochastic processes, I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling--Geometric algorithms, languages, and systems, I.3.m [Computer Graphics]: Miscellaneous

1. Introduction

There has long been interest in systematic ways of filling space with distinct shapes. Two-dimensional tilings in which space is filled by repetition of the same shape have been known since antiquity and are widely used in decorative art. The checkerboard is perhaps the simplest example of such a tiling.

In the early 20th century recursive fractal methods for filling 2D space with an infinite number of shapes were discovered. One of the best known is the triangle construction of Sierpinski [Man77]. Sierpinski started with a filled triangle and successively removed smaller triangles. His construction can be space filling if the opposite procedure is followed where one begins with an empty triangle and recursively adds ever-smaller triangles. Thus 2D space is filled in the limit with an infinite number of ever-smaller similar triangles. Such recursive fractals have a single specific fractal dimension D (D = log(3)/log(2) for Sierpinski triangles). Fractal studies in physics [DW02] [DW03] [DHA08] generally proceed by adding shapes to an empty region, while mathematicians prefer the opposite procedure where a filled region is emptied by successive deletions ("tremas" in Mandelbrot's usage). Where it makes a difference we have followed the physics usage.

Following the publication of Mandelbrot's book [Man77] fractals have been found to be useful models in physics, geology, economics, and many other areas [Buc00] [Bal04]. Such fractals characteristically deal with random events whose statistical distributions follow a power law. The distribution of earthquake magnitudes [Buc00] is one of the most well known examples. Physicists have used fractal ideas to model and study the random packing of various shapes, with the goal of understanding how fractured materials behave [DW02] [DW03] [DHA08]. Such packing studies generally impose the condition that the shapes have mutual contact (kissing) points, and the random patterns that appear are found by computer simulation.

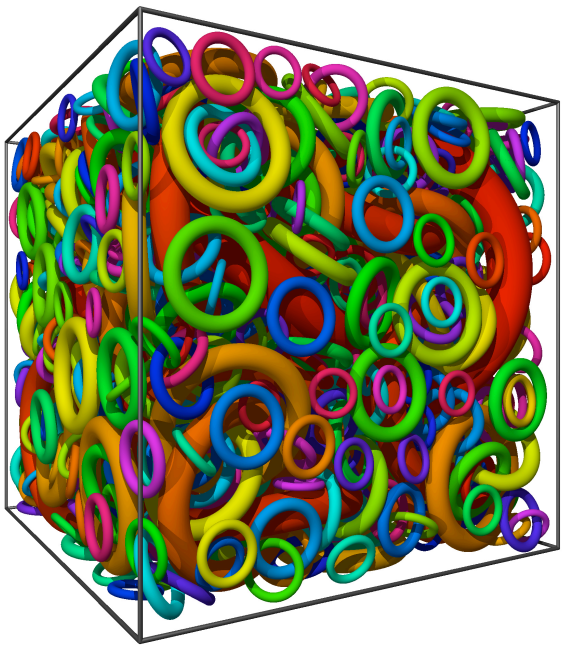

Figure 1: *Toroidal rings fractalized within a cube. The ring cross sectional radius is 15% of the radius from the center of the hole to the center of the ring. The parameters are: 1000 shapes, c = 1.1, N = 2 with a 68% fill. Inclusive boundary conditions.*

The space filling algorithm described here differs from the usual packing algorithms [DW02] [DW03] [DHA08] in that the shapes have no contact points [Sod36] [KS43], and the fractal dimension *D* rather than being determined from simulated data can be specified at the outset over a substantial and continuous range of values. A common class of packing algorithms randomly sample the area for an available location for the next shape and then grow a placed object until it touches a neighbor. The algorithm presented here does not involve growing shapes to fill voids but rather the size of the next shape is entirely predetermined. The algorithm will be described in detail in section 2 of this paper.

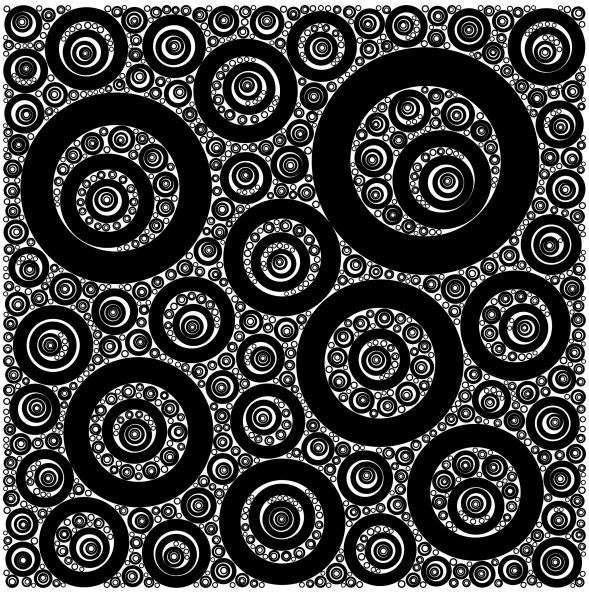

Figure 2: *A two-dimensional fractal based upon an annular ring shape. The inner circle is 2/3 the radius of the outer circle. The parameters are: 2000 shapes, c = 1.2, N = 2 with a 78% fill. Inclusive boundary conditions.*

In determining the behavior and features of the algorithm the authors have proceeded using the inductive method commonly applied in physics. By carrying out computational experiments, simulations if you will, the authors have reached conclusions based on consistent patterns seen in the results of thousands of computer runs. Thus the paper does not offer any formal proofs. From the viewpoint of pure mathematics all of the mathematical claims made here can be thought of as conjectures. The terminology for fractals used in the paper generally follows Mandelbrot [Man77].

What uses can be found for this algorithm? The paper [BS13] by the authors describes how it might be used to create procedural textures in computer games and graphics in general. The web sites [SB12] contain many examples of the 2D algorithm used as art, in which it can be seen that the algorithm can produce a rich variety of engaging imagery. The many real-world fractal distributions described in [Buc00] [Bal04] largely lack physical or geometric models. It is anticipated that the algorithm described, should it become widely known, may allow geometric modeling of some of the random phenomena found in various fields. Several natural objects have a visual appearance similar to the images generated here; these include but are not limited to bubbles on the surface of liquids, pore spaces in breads, and certain geological structures.

Figure 1 is an example of applying the algorithm in three dimensions, filling a cubic region using toroidal rings. There are in total five degrees of freedom in the placement of each ring, position (*x*,*y*,*z)* and two orientation angles. It can be seen that this fractal has random topology; note for example how some of the rings are chain-linked to others.

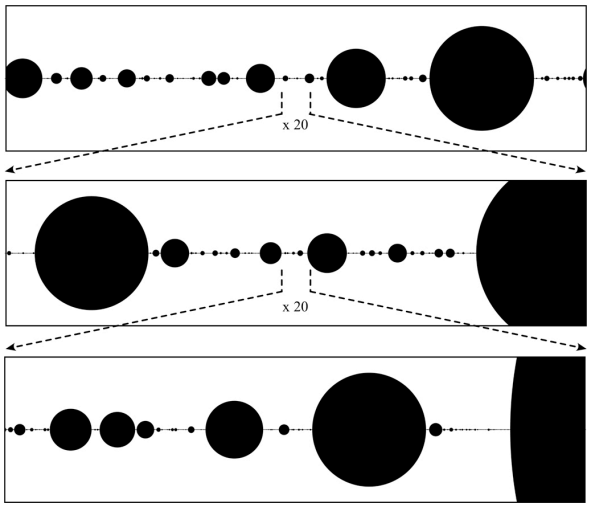

Figure 3: *A one-dimensional fractal. The upper image shows the complete fractal with periodic boundary conditions. The subsequent images are zoomed into the central 5% of the previous image with obvious self-similarity. The parameters are: 30,000 shapes, c = 1.4, N = 1 with a 99.98% fill.*

Figure 2 presents a two dimensional case where a square bounded region is filled with annular rings. With hollow shapes such as these it can be seen that the smaller shapes may end up being placed either inside or outside of previously placed shapes. There is strong correlation in the form of "nesting".

Figure 3 is an example in one dimension. The one-dimensional case is particularly simple and there are a limited number of possible shapes. Very large percentage fills can be achieved and the self-similar nature at each level of zoom is evident. Here the line segments of length L_i are represented as circles of diameter L_i to improve the visualization.

2. The statistical geometry algorithm

In the following description of the algorithm we will consider the two-dimensional case. Let A be the total area to be filled. We define a sequence of areas A_i (i = 0, 1, …) given by the rule

$$A_i = \frac{A}{\zeta(c,N)(i+N)^c} \qquad (1)$$

where $\zeta(c,N)$ is the Hurwitz zeta function [AS64] defined by

$$\zeta(c,N) = \sum_{i=0}^{\infty} \frac{1}{(i+N)^c} \qquad (2)$$

This is known to converge for $c > 1$ and $N > 0$.

In view of equation 2 one can write

$$\sum_{i=0}^{\infty} A_i = \sum_{i=0}^{\infty} \frac{A}{\zeta(c,N)(i+N)^c} = A \qquad (3)$$

such that the sum of all areas A_i is the total area A to be filled, that is, if the algorithm does not halt then it is space-filling. The parameters c and N can be chosen over a substantial range of values. Parameter N need not be an integer in the formulation above, but integer values are used in the examples and discussion presented. The general influence of parameters c and N can be illustrated in figure 4. For higher values of c the initial shapes have a larger area but as i increases the area falls off more quickly. Similarly for smaller N the initial shapes are larger but fall off more quickly as i increases.

The algorithm as implemented by a computer simulation can be described as follows.

Step 1. Let $i = 0$. Place a shape with area A_0 at a random position within the area A of the region to be filled such that it does not overlap the boundary. (See later where this is relaxed in the case of periodic boundary conditions). Place the position and dimensions of the shape in the placed-shapes database. Increment i. This is referred to as the initial placement.

Step 2. (iteractive). Choose a random candidate position for a new shape with area A_i, again, entirely within the boundary of the region to be filled. This is a trial. Test whether shape A_i overlaps any previously placed shape? If it does then repeat step 2. If not then place the position and dimensions of the shape in the placed-shapes database, increment i, and repeat step 2. This is referred to as a placement. Stop when a desired number n of placed shapes has been reached or a chosen percentage fill has been achieved.

The result is a random fractal space-filling collection of shapes within a fixed boundary whose areas follow a power-law sequence.

Only step 2 is iterative. This is a very simple process. The two parameters c and N allow the statistical and visual properties to be varied. The algorithm is not dependent on the exact shape; experimental evidence to date indicates that the algorithm works (within some range of c values) for any shape. Implementing this space-filling algorithm for any particular shape requires three things:

(1) A relationship between the linear dimensions (scale) and the area of the shape.

(2) An intersection test between two shapes, that is, "Does shape 1 in the current proposed position, scale, and orientation intersect with shape 2 at its existing position, scale, and orientation?"

(3) An intersection test of a shape at a particular position, scale, and orientation with the boundary of the region being filled.

The position probability distribution for trials in the algorithm as presented is taken to be uniform, that is, every position is equally probable. This is not a requirement but whatever distribution is used it must obviously span the entire area, a uniform distribution is chosen here as an unbiased sampling.

It is believed that the algorithm can be used with any shape or combination of multiple shapes having the area sequence defined in equation 1. Geometric similarity of the shapes is not a requirement. Shapes studied in 2D include circles, squares, squares with random rotation, rectangles with various aspect ratios, mixed squares and circles, annular rings, blobs, eights, ells, triangles, arrows, lenses, crescents, stars, gears, diamonds, bicircles, quadcircles, and quadsquares. 3D shapes tested include spheres, cubes, tetrahedra, octahedra, and torii. These and other examples can be found at the authors' web sites [SB12].

The algorithm description and equations 1 to 3 refer to areas since the algorithm has been introduced in two dimensions. In one dimension the formulation can be reduced to line segments (lengths) and in three dimensions it can be extended to volumes. As such it can also be extended into higher dimensions and to the corresponding hyper-volumes.

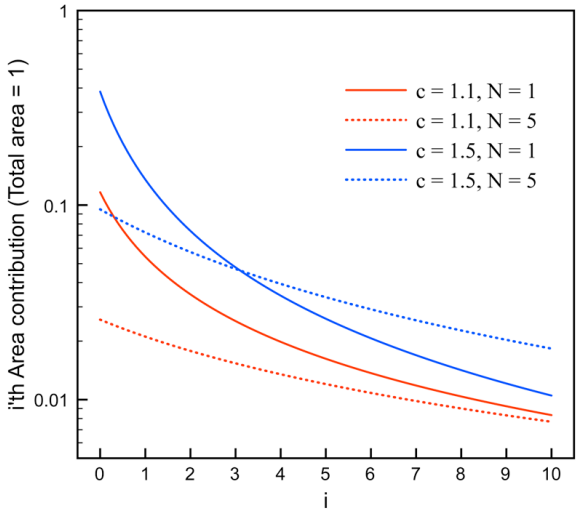

Figure 4. *The effect of different values of c and N on the function for the area on the i'th iteration from equation 1.*

The algorithm was introduced for shapes not crossing the boundary of A (referred to here as an inclusive boundary). It works equally well with a periodic rectangular boundary of lengths L_x and L_y. In this case the intersection test for placement needs to consider the object also appearing at positions $x \pm L_x$ and $y \pm L_y$. When a new shape placement is accepted and if it crosses any boundary then the same shape is

placed in the database at $x \pm L_x$ and/or $y \pm L_y$ to ensure periodicity. See figure 8 as an example of periodic boundary conditions, in these cases the image can be seamlessly tiled.

The c parameter relates directly to fractal dimension D (section 5). The larger the value of c is, the more rapidly the sizes of the shapes diminish as more shapes are placed. There is a largest c value that varies depending on the shape in question and the region being filled (section 3). The number of trials required depends very strongly on c, see figure 5. The N parameter can be used to adjust the size of the largest shape. When N is large, the first few shapes are smaller than with a low N value, and their areas fall off more slowly with the number of placements. The fractal dimension D is independent of the value of N.

The concept of a maximum c value is developed more fully in section 3 of the paper. The minimum c value only has the requirement that c be > 1. Compact and convex shapes such as circles or squares generally have higher maximum c values than concave or convoluted shapes. Of the 2D shapes studied, squares have the highest maximum c value. Squares require fewer trials per placement than any other shape studied in 2D. The highest c value in 1D is about 2.7 (with N = 1); for 2D it is about 1.57 (squares), and for 3D it is about 1.2 (cubes). The dividing line between c values where fractalization is possible or not is not a sharp one since it can depend on the position of the first random placement, see later.

If the random numbers and length parameters used in the algorithm are thought of as having infinite precision the probability of two shapes actually touching is vanishingly small. The placement of a given shape is random, but this is a highly constrained randomness, influenced by all of the previous placements. At any given step placement is dependent on the entire prior history of the process.

3. Does the algorithm halt?

We say that the algorithm has halted when it arrives at a state where there is no place in the gasket large enough to accommodate the next-to-be-placed shape. We present evidence that within a range of c values > 1 and below some limiting value the algorithm runs without halting. This evidence is based upon large numbers of computer experiments often designed explicitly to find halting examples.

It can be seen in figure 5 that the trials data follow a reasonably straight line in log-log coordinates as the cumulative number of trials becomes large, indicating that the cumulative number of trials obeys an approximate power law versus n. The data has less scatter for low values of c, and increasing scatter for higher values of c. Such an approximate power law was seen for all cases studied. This indicates that any number of shapes can be placed with a finite (but possibly large) number of trials for the c and N values shown in figure 5 and that the algorithm does not halt simply because n becomes large. It is evident from figure 5 that as n increases (area of remaining gasket decreases) the number of trials needed to place a shape can become extremely large. Thus the algorithm becomes increasingly inefficient and this is especially so for large c values. The amount of noise and fluctuation in the process is also greater for large c values (upper curves).

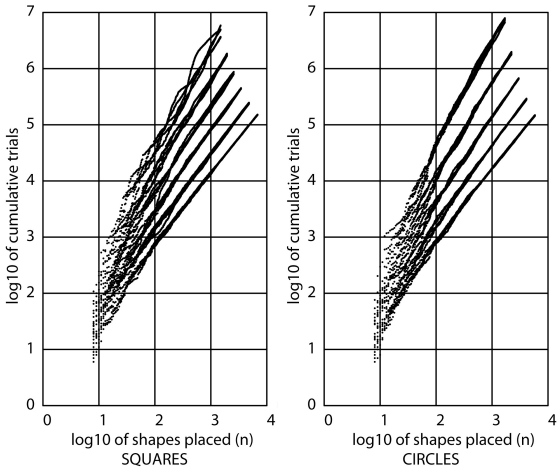

Figure 5: *Log-log plots for the cumulative number n_t of trials (vertical) versus the number n of placed shapes (horizontal) for squares and circles. For the circles the c values are (from the bottom): 1.20, 1.25, 1.30, 1.35, 1.40. For the squares the c values are (from the bottom): 1.20, 1.25, 1.30, 1.35, 1.40, 1.45. Data from five runs is shown for each c value. In all cases N = 1.*

If we undertake regression of the data of figure 5 to estimate the exponent in the underlying power law, the exponent for the smooth data at large n values is found to be approximately equal to c.

It is usual in programming random searches to place an upper limit on the number of trials, stopping the algorithm when this is exceeded. A failure of this kind is not halting in the sense defined above. For many of the examples presented here the value of n has been chosen relatively low to improve the appearance of the figure given the limited resolution available.

The algorithm does halt for large c values. One cause of this is the geometry of the first few shapes. Consider the case of squares fractalized within a square. The best case for placement of the first two squares is one where the first square abuts one corner of the bounding square and the second abuts the diagonally opposite corner. This state leaves the greatest amount of room for subsequent squares. If the widths of the first squares are w_0 and w_1 respectively it is evident that if $w_0 + w_1$ exceeds the width of the bounding square it is not possible to place both square 0 and square 1 by random search and the algorithm halts. Calculations show that when N = 1 this occurs for c = 1.5224, which is thus a hard upper limit on usable c values for squares. When high values of c are

tried with $N = 1$ it is found that as one approaches $c = 1.50$ from below the number of runs which halt increases rapidly. Most observed halting events of this kind have been found in the first ~100 placements.

Changes in the value of N has only a modest effect on the maximum allowed value of c, with larger N values leading to somewhat increased maximum c values. In practice the algorithm has been observed to run without halting for any c value greater than 1 and less than the maximum c.

If $N > 1$ the requirements imposed by the geometry of the first few shapes are more relaxed since the first few shapes are nearly the same size or smaller. For larger N values it has been found possible to make successful computer runs for squares with c values up to ~1.57.

For the example of squares where $N = 1$, and $c \leq 1.45$, as in the data of figure 5, halting has not been observed to date in hundreds of runs. For these parameters the evidence is that the algorithm does not halt. While formal proofs are lacking, the algorithm runs without stopping in practice and applications in games, visual art, and modeling are also not subject to halting problems. The largest usable c value depends on the shape fractalized, and is in general highest for simple shapes (for example, circles and squares) and lowest for concave and convoluted shapes.

It is difficult to make meaningful comparisons between the 1D, 2D, and 3D cases because, for example, a circle is not a sphere, and the maximum c values do depend on the dimension. Several trends can however be observed. If 2D and 3D cases are run near their maximum c values, it generally requires far more trials to reach a given percentage fill for 3D. The same holds for 1D and 2D cases; the 2D case requires more trials.

The evidence from experience running the algorithm with a wide variety of c, N, and shapes is that *there is a wide range of c values from 1 up to some critical value where halting is never observed*. Proof of this would be interesting, but at the present time it must be a conjecture supported by data. In order to put this on a quantitative basis a large number of runs were made for circles fractalized in a square ($N = 1$) with various c values and the number of halting events was determined, see figure 6.

The open circles show data from 2000 circle runs for each point, with inclusive boundaries. There were no halting events at all for the point with lowest c. The squares show similar data from 3000 circle runs with periodic boundaries. For inclusive boundaries a run was taken to be non-halting if it did not halt in 8,000,000 trials at any placement. For periodic boundaries the run was taken to be non-halting if it did not halt in 6,000,000 trials. Most halting events occurred by the first 50 placements.

These results can be summarized as "The algorithm never halts for sufficiently low c and always halts for sufficiently high values of c". The halting probability traces out a smooth curve between the non-halting and the halting regions.

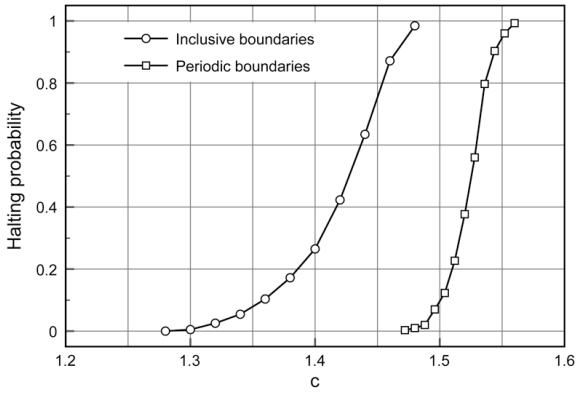

Figure 6. *Halting probability as a function of c for circles fractalized in a square with inclusive and periodic boundaries.*

The argument about fitting of the first two shapes made above for squares becomes much more complicated when we consider periodic boundaries. Here each shape which overlaps the boundary has one or more "partners" located at $x \pm L_x$ and/or $y \pm L_y$, where L_x and L_y are the dimensions of the bounding rectangle. The fact that runs with periodic boundaries do not halt over a wider c range suggests that the fitting of the first few shapes is less constrained with periodic boundaries.

Why is the maximum c lower for a less compact shape? Consider a shape that is a square frame of side length u, with a square of side length du cut out of its middle (i.e., it is a hollow shape). Assume this shape is fractalized (inclusive boundaries) in a square of side length s. Consider the first three iterations resulting in a placement of shape number 0, 1 and 2. The side lengths s_0 and s_1 of the first two shapes will be larger than for a solid square of the same area, and it is found when $d = 2/3$ that $s_0 + s_1 > s$ when c = 1.2313. However, at this c value it is possible for shape 1 to fit inside of shape 0 (possible but improbable in a random search). When c reaches 1.2936 we find so that $s_0 + s_2 > s$ so that shapes 0 and 2 no longer fit and all runs will halt for $c > 1.2936$. This is much lower than the corresponding value $c = 1.5224$ for a compact non-hollow square, and shows by example how a less compact shape lowers the maximum c value.

4. The average gasket width

Is there always a large enough space in the gasket to place more shapes? It is interesting to track the average gasket width versus the linear dimension(s) of the shapes being placed. Let $A_g(n)$ be the total gasket area after the n^{th} placement, and $P_g(n)$ be the total gasket perimeter after the n^{th} placement. The gasket has a quite irregular shape, so the following definition of the average gasket width has been assumed (in 2D):

$$\overline{w}_g(c,N,n) = \frac{A_g(n)}{P_g(n)} \tag{4}$$

This quantity has the units of length and can be computed for any shape. In what follows we will assume circles. If we divide $\overline{w}_g$ by the diameter of the next-to-be-placed circle we have a dimensionless average gasket width *b(c,N,n)*. This is a measure of the relative amount of space available for the next circle and is given by

$$b(c,N,n) = \frac{A_g(n)}{P_g(n)\cdot 2r_{n+1}} \tag{5}$$

where r_{n+1} is the radius of circle *n+1*. This is a dimensionless non-random quantity that is invariant with respect to the particular random placements used. For circles the area and radius of the i^{th} circle are

$$A_i = \frac{A}{\zeta(c,N)(i+N)^c} \tag{6}$$

$$r_i = \sqrt{\frac{A}{\pi\zeta(c,N)}}\,\frac{1}{(i+N)^{c/2}} \tag{7}$$

whence

$$b(c,N,n) = \frac{1}{4}\,\frac{\zeta(c,N) - \sum_{i=0}^{n}\frac{1}{(i+N)^c}}{\frac{1}{(1+n+N)^{c/2}}\sum_{i=0}^{n}\frac{1}{(i+N)^{c/2}}} \tag{8}$$

This quantity can be readily computed and the results for the case of a circle are given in table 1 (for $N = 1$) along with the average number of trials per placement.

It is seen that *b(c,1,n)* appears to go to a finite limit as $n \to \infty$, and that *b(c,1,n)* changes very little with *n* for large values of *n*, although it does show a strong variation with *c*. Since *b(c,1,n)* is a measure of the relative space available for placement of the next circle, the interpretation of these results is that there is a "just in time" relationship such that the space available for placement falls in direct proportion to the size of the circle to be placed. This helps to explain why the algorithm does not halt. Computer runs placing a million circles have been performed with no sign of halting.

The variation of *b(c,1,n)* with *c* shows that *c* controls the average spacing between shapes. With low *c* the average spacing between shapes is relatively large, while for high *c* values the spacing can become quite small. It is also seen that the patterns are more ordered (less random) for large *c* values than for small *c* values.

The dimensionless gasket width *b*(*c*,*N*,*n*) provides an average measure of the amount of "wiggle room" available at placement for a given *c*, *N*, and *n*. It can be seen that *b* drops rapidly as *c* increases for all *n* values, indicating that as *c* rises there are fewer places that can accommodate random placement of the next circle. This dependence shows itself in the steep increase in the number of trials needed for a placement as *c* increases and the steadily tighter fitting of the shapes. As *n* increases *b* decreases, but at a steadily smaller rate. For the highest *n* values in table 1 the drop is very small. For example when *c* = 1.32, *b* only drops by 0.9 percent as *n* goes from 100,000 to 1,000,000. This shows that for large values of *n* the available space for the next placement is falling at about the same rate as the diameter of the next circle, which helps to explain the observation that the algorithm does not halt.

The much steeper drop of *b* as *n* increases during the first 100 placements supports the observation that when a run halts it does so during the early placements.

It is not possible to determine if *b* goes to a finite limit when $n \to \infty$ from the numerical data in table 1, although *b* changes very slowly when *n* is large.

Table 1. *Values of the dimensionless gasket width for circles with N = 1 versus placement number n and exponent c.*

b(*c*,1,*n*)	*c*=1.24	*c*=1.32	*c*=1.40	*c*=1.48
n=10	0.6024	0.4314	0.3291	0.2613
n=100	0.4602	0.3209	0.2379	0.1831
n=1,000	0.4200	0.2881	0.2096	0.1579
n=10,000	0.4055	0.2754	0.1979	0.1469
n=100,000	0.3998	0.2700	0.1926	0.1415
n=1,000,000	0.3975	0.2676	0.1900	0.1387

5. Fractal dimension

The scaling discussion of length and area in [Man77] leads to the relationships *D = 2/c* for the 2D case and *D = 3/c* for the 3D case. This 2D result can also be shown to follow from equations in [DHA08]. Their prescription for the 2D involves making a sorted list of all the radii (or another linear dimension) of the given shape, i.e., pairs (r_1,1), (r_2, 2), …, (r_i, *i*), where the second (integer) number is the sequential order of the shape by size. One then makes a log-log plot of *i* versus r_i and the best-fit slope of this plot is an estimate of their $(1-\alpha)$ parameter. Because the sequence of areas or radii in a statistical geometry fractalization is an *exact* power law in this case the best-fit line will pass exactly through each point and will have slope $-2/c$. By parallel reasoning one finds *D = 1/c* for the 1D case. Because the area and length sequences have no randomness, these are exact results. Thus it is possible to specify the fractal dimension a priori over a substantial range. Box counting estimates of *D* performed numerically on the images have confirmed these relationships. The fractal dimension is unaffected by the randomness of the placement process.

6. Mixed shapes

The definition of the algorithm in section 2

depends only upon the area of the shape. This would suggest one could use a mixture of shapes as long as the area relationship is maintained.

Figure 7 is an example of two shapes, circles and squares, with the shape alternating after each placement. It can be seen that there is strong clustering (correlation) in the placed positions, with squares mostly near other squares and circles near circles. Penetration of circles into mostly-square regions and vice versa occurs at all length scales. This cluster correlation becomes more pronounced with larger *c* values.

The type of correlation or "segregation" demonstrated here has been found in most of the multi-shape fractalizations that have been studied to date.

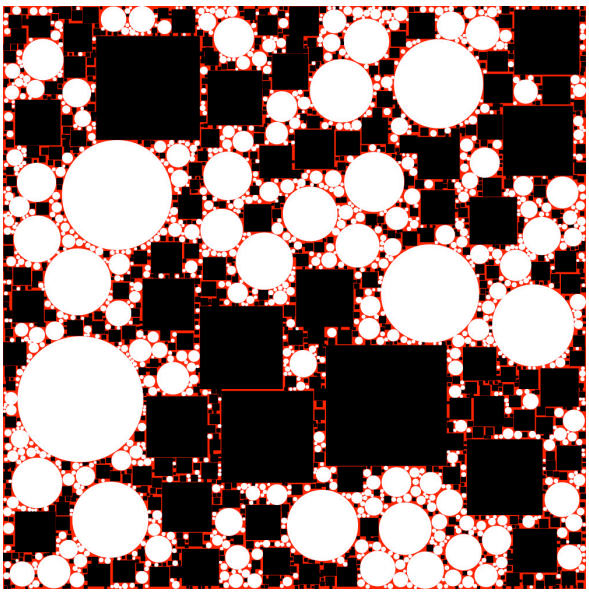

Figure 7: *Mixture of circles and squares. The parameters are: 2500 shapes, c = 1.35, N = 8, fill = 87%. Inclusive boundary conditions.*

7. Irregular shapes and boundaries

To further explore the algorithm with irregular shapes, a fractal was constructed using an irregular shape (blob) that changes randomly with every trial; it is defined in polar coordinates by

$$r(\theta) = R\left\{1 + \delta\left(\sum_{j=2}^{4} \cos(j\theta + \varphi_j)\right)\right\} \qquad (9)$$

Here the φ_j are phase angles that are each varied randomly over the range 0 to 2π. The area of such a shape is conveniently independent of the phase angles. The area is determined by the parameter *R* and the "non-circularity" is controlled by the parameter δ. No two shapes are ever the same and there are five degrees of freedom. See figure 8 for an example of filling space with this type of shape.

The authors conclude that similarity of shapes in the sense of this word in geometry is not a requirement for the algorithm.

Figure 9 shows a filling with a highly convex and "sharp" shapes. The value of *c* is close to the maximum for this particular shape. This is a further example [SB12] of an extreme shape, all of which have been experimentally demonstrated to being able to be fractalized.

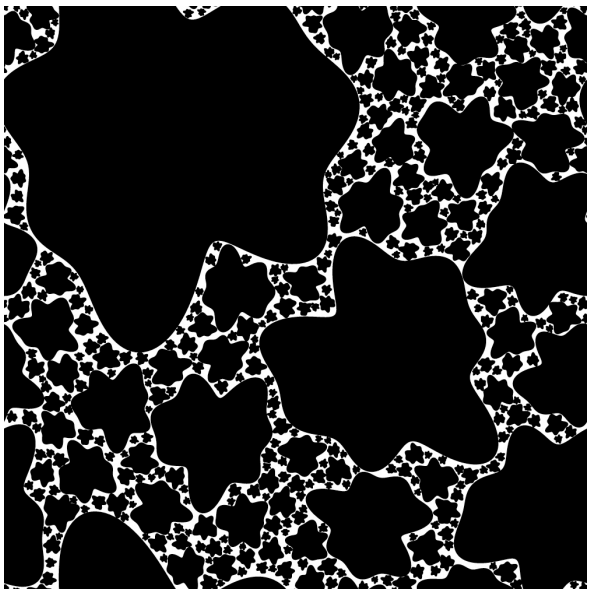

Figure 8: *Irregular "blob" shapes fractalized. The parameters are: 400 shapes, c = 1.32, N = 1, δ = 0.12, fill = 88%. Periodic boundary conditions.*

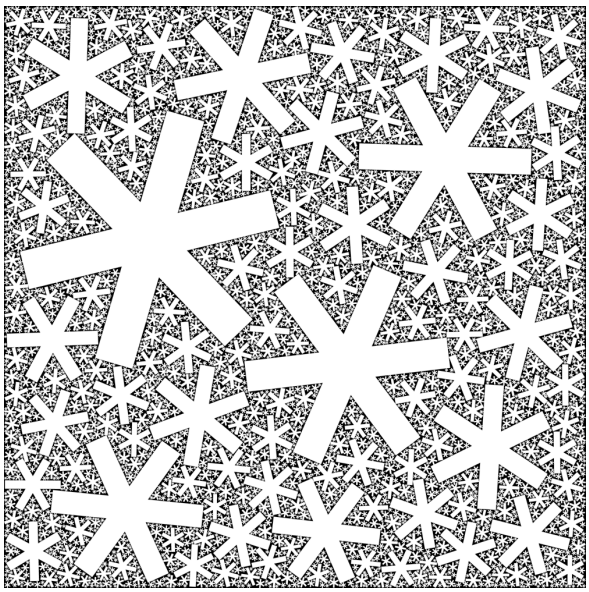

Figure 9: *Extremely concave shapes. 5000 shapes, c = 1.16, N = 2, fill = 73%. Inclusive boundary conditions.*

In addition, the description of the algorithm is not restricted to filling only rectangular bounded regions. All that is required is the calculation of the area of the region to be filled and an intersection test between the filling shapes and the boundary. The example in figure 10 shows, in 2 dimensions, randomly orientated squares filling an annulus. Figure 11 is an example of

a polygonal boundary shape with sharp pointed features and additionally filled with “sharp” triangular shapes. Figure 12 is a further example, this time in 3 dimensions showing cubes filling a sphere. In these last three examples there is no opportunity to have periodic boundary conditions, at least not in the sense of a regular rectangular tiling.

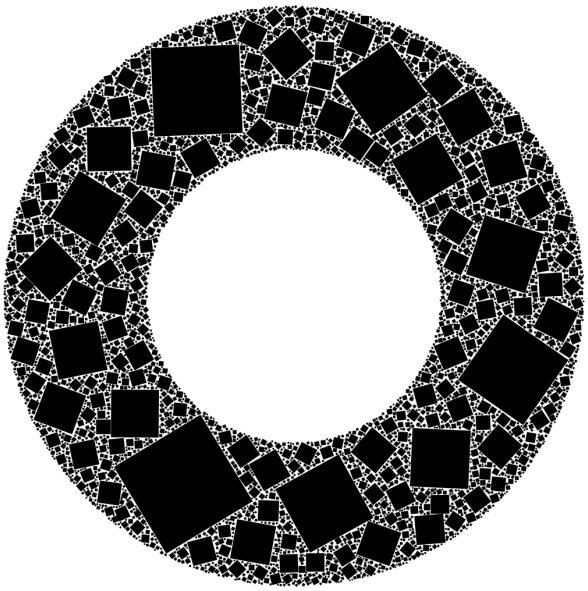

Figure 10. *Non-rectangular boundaries, in this case an annulus. 4000 square shapes, c = 1.2, N = 3, fill = 80%. Inclusive boundary conditions.*

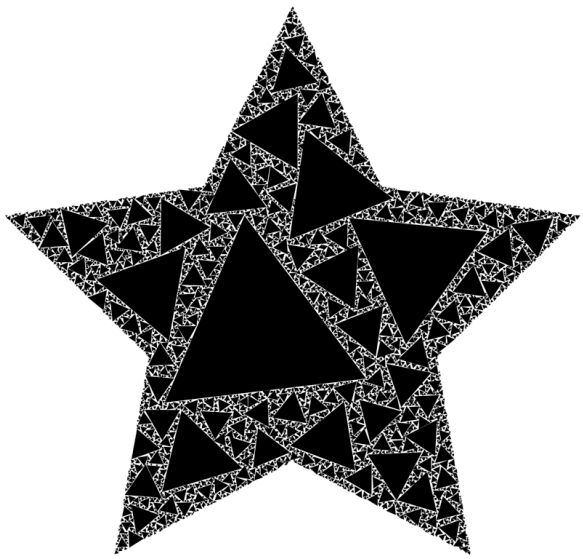

Figure 11. *Non-regular arbitrary polygonal boundaries, in this case a star figure filled with 4000 triangles. c = 1.1, N = 1, fill = 72%. Inclusive boundary conditions.*

8. Discussion and conclusions

The results presented here are based upon experience, relationships and patterns arising from numerous computer experiments. The authors do not give proofs of results such as "any shape can be fractalized into any region" because such proofs are currently lacking. It is hoped that proofs will be found for some of the claims and observations made here when this algorithm becomes more widely known to the research community. Until that time our results must be viewed as conjectures from the viewpoint of pure mathematics.

The halting problem may be the most interesting mathematical aspect of the work. The evidence for non-halting is of two kinds: actual run-time data (section 3) and a mathematical demonstration that the relative amount of space available for placement remains nearly constant as the algorithm progresses (section 4).

The basis for the claim that the algorithm can fractalize any shape up to a limiting *c* value is that the authors have examined a large number of shapes (including sparse and non-compact shapes) [SB12] in one, two, and three dimensions. While there was the expectation to find some that cannot be fractalized, thus far none have been found. A formal proof or disproof of this claimed property would be interesting and challenging.

Based on the experimental evidence, the main claims are:

1. The algorithm is space-filling if it does not halt.
2. The algorithm does not halt within a wide range of *c* and *N* values.
3. The algorithm works with any shape sequence obeying the area relation in equation 1.
4. Any bounded area can be filled for some range of *c* and *N*.

Such fractals exist whether one constructs them or not. They can be viewed as another way of representing space, namely a random fractal representation.

Suppose the algorithm is set up for an area of, say, $1m^2$, and tiles are made of the first *n* shapes. A tile-setter could then mark off a circle or square of area $1m^2$ and be assured that he could place the tiles within it anywhere he wished and they would always fit provided only that he always proceeds in order of size beginning with the largest tile.

Random fractals found in nature [Buc00] [Bal04] can be difficult for non-specialists to grasp. Such important properties as statistical self-similarity and "scale-free" are not easily understood. Images of these fractals provide "pure" examples of such behavior and can be useful in conveying the nature of random fractals to non-specialists.

The fractals described here differ substantially from traditional packing algorithms that have been the subject of a number of interesting papers [DW02] [DW03] [DHA08]. Here the shapes are non-touching and the gasket is a continuous whole, unlike usual packing methodologies. The fractal dimension *D* can be specified ab initio, rather than being numerically computed. Seeding and other initial conditions are not

required.

A large number of high-resolution color fractal images can be seen at the authors' web sites [SB12]. An earlier less-complete account of this work by Shier can be found in the conference proceedings of ISAMA 11 [Shi11].

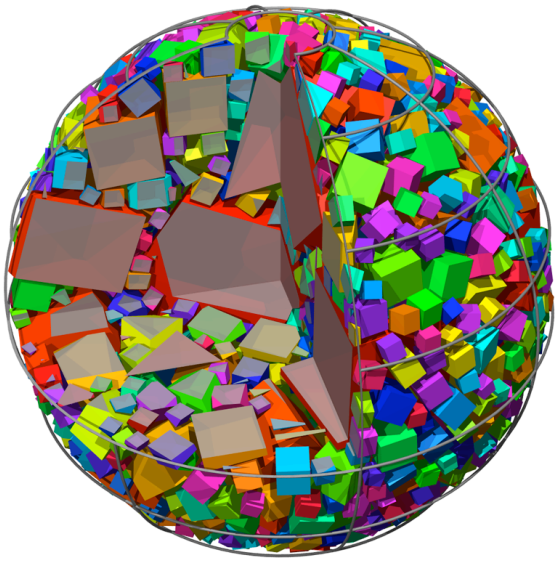

Figure 12. *Non-rectangular boundaries in 3 dimensions, cubes contained within a sphere. A cut-away section shows the interior structure. 1000 cubes, c = 1.1, N = 1, fill = 55%. Inclusive boundary conditions.*

Acknowledgements

The work was supported by iVEC through the use of advanced computing resources located at iVEC@UWA.

References

[AS64] ABRAMOWITZ, M and STEGUN, I.A., *Handbook of Mathematical Functions*, (1964) Dover Publications, New York. ISBN 0-486-61272-4.

[Bal04] BALL, P.: *Critical Mass - How One Thing Leads to Another*. Farrar, Straus, and Giroux, New York, 2004.

[BS13] BOURKE, P., SHIER, J.: *Space Filling: A new algorithm for procedural creation of game assets*. Proceedings of the 5th Annual International Conference on Computer Games Multimedia & Allied Technology (CGAT 2013). [In Press].

[Buc00] BUCHANAN, M.: *Ubiquity*. Three Rivers Press, New York, 2000.

[DHA08] DELANEY, G. W., HUTZLER, S., ASTE, T.: *Relation Between Grain Shape and Fractal Properties in Random Apollonian Packing with Grain Rotation.* Phys. Rev. Letters, 101, 120602, 2008.

[DW02] DODDS, P. S., WEITZ, J. S.: *Packing-limited growth.* Phys. Rev. E, 65, 056108-1, 2002.

[DW03] DODDS, P. S., WEITZ, J. S.: *Packing-limited growth of irregular objects.* Phys. Rev. E, 67, 016117-1 2003.

[KS43] KASNER, E. and SUPNICK, F. *On Apollonian Packing of Circles.* Proc. Natl. Acad. Sci. USA 29, 378-384, 1943.

[Man77] MANDELBROT, B.: *Fractals; Form, Chance, and Dimension.* W. H. Freeman and Company, San Francisco, 1977.

[SB12] SHIER, J. BOURKE, P.: The authors' web sites http://john-art.com (JS) and http://paulbourke.net/randomtile (PB).

[Shi11] SHIER J.: Proceedings of ISAMA 11, June 13-17, 2011. Hyperseeing, Summer, pp. 13140 (2011).

[Sod36] SODDY, F. *The Kiss Precise*. Nature 137, pp 1021, 1936.