Proc. of the Eighth Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2019

ISBN: 978-1-63248-165-8 DOI : 10.15224/978-1-63248-165-8-06

Constructing a 3D Fractal Model with Two Typical Space Filling Curves

Ningping Sun and Shoji Kugizaki

***Abstract*— We have devised and implemented a tessellation algorithm with two typical space filling curves, Hilbert curve and Sierpinski curve, and applied the tessellation algorithm into the construction of 3D fractal models. In this paper we shall describe our tessellation algorithm with Hilbert curve and Sierpinski curve, and provide the method how to map the two tiled planes on the surface of models. Some result of verification experiments will be provided also.**

***Keywords*—Hilbert curve, Sierpinski curve, recursion, tessellation, modeling and rendering, primitive models.**

I. Introduction

Currently, 3DCG is widely known and used in various scenarios. In order to create a 3D model modeling is necessary, that is, it is necessary to specify polygons by specifying vertices and faces. For modeling, you can employ methods such as assigning coordinates directly in the program, drafting with CAD, processing images taken from scanners or photographs, and more. Creating a complex model is very difficult, and the model patterns that can be done manually are limited. However, by using the concept of fractal geometry, we can use computational formulas to create complex models that are difficult to build by humans [1] [2].

We have devised and implemented a tessellation algorithm with two typical space filling curves, Hilbert curve and Sierpinski curve [3][4]. The effect of further filling the space is improved by overlapping the curves of different recursive times in the same space. When superimposing curves with different recursive times, the necessary adjustments need to be made.

We focused on constructing a 3D fractal model by applying the above tessellation algorithm, have developed a rendering technique to apply fractal patterns to the surface of primitive models such as Riemann sphere, cylinder, cube, cone, and torus, in which the ordinary modeling are not needed any more.

In this paper we shall describe our tessellation algorithm with Hilbert curve and Sierpinski curve, and provide the method how to map the two tiled planes on the surface of models. Some result of verification experiments will be provided also.

Ningping Sun Ph.D
National Institute of Technology, Kumamoto College
Japan

Shoji Kugizaki
National Institute of Technology, Kumamoto College
Japan

II. Tessellation Algorithm with Two Typical Space Filling Curves

Hilbert curve and Sierpinski curve are well known as famous recursive curves and space filling curves. We use these two curves to further fill space and construct fractal models. Here we explain recursive algorithm of these two curves and provide the tessellation methods individually.

A. *Superimposed Hilbert Curves*

Hilbert curve is a famous space filling curve after its inventor, the mathematician D. Hilbert (1891). Hilbert curve of order i, H_i, can be plotted by compositing four patters of H_{i-1} of half size and appropriate rotation shown in Figure 1, in which H_0 is defined as empty and H_1 is represented by three connecting straight lines. Figure 2 gives three Hilbert curves of order 1, 2 and 3 [3].

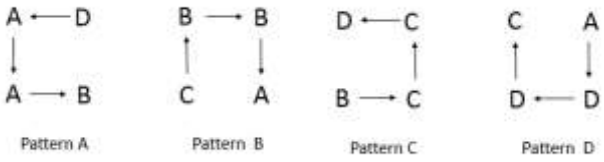


Figure 1. Four patterns of Hilbert curve

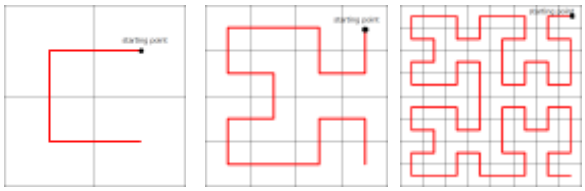


Figure 2. Hilbert curves of order 1, 2, and 3

We can see from Figure 2 that the space is divided into squares for order k. The Hilbert curve passes through all squares from the starting point at the center of the most right upper corner. According to the four patterns, the curve plots traces while walking, and ends at the end point at the center of the bottom right corner. For example, H_1 goes from the upper right center, the upper left center, the lower left center, passes through four squares, and ends at the lower right center.

Because different k-order curves pass through the center of squares, when these curves are drawn on the same plane, they may intersect vertically, but they are not drawn on each other in the parallel direction under the condition that fill rate is less or equal to 100 percent.

We use a superposition of Hilbert curves of different orders to further fill the space, which is called tessellation or

Proc. of the Eighth Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2019

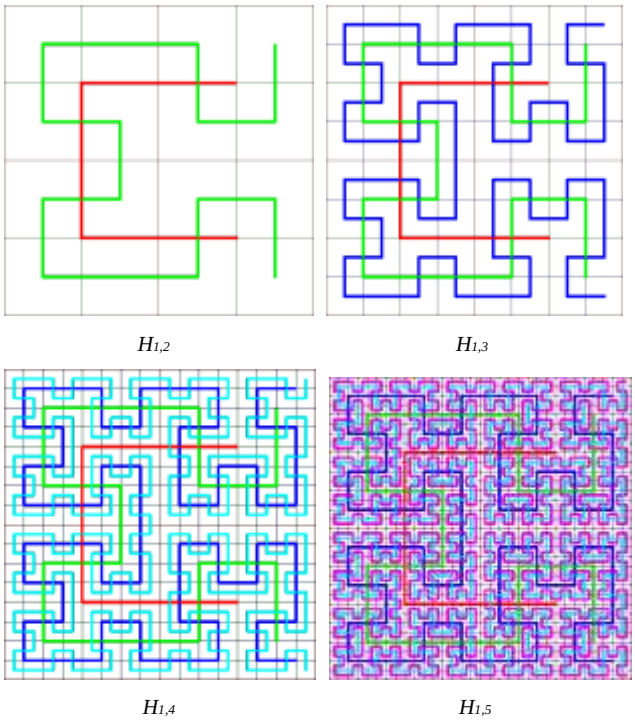
ISBN: 978-1-63248-165-8 DOI : 10.15224/978-1-63248-165-8-06

tilling. Assuming (x_0,y_0) is the coordinate of the starting point of H_k as shown in Figure 2, we plot Hilbert curves of different orders from order *1* to order *n* denoted as $H_{1,n}$ in the plane of (-1,1).

For order *k*, $1 \le k \le n$, the length of a curve segment to be drawn is $\frac{1}{2^{k-1}}$, and

$$x_0 = 1 - \frac{1}{2^k}, \quad y_0 = 1 - \frac{1}{2^k} . \tag{1}$$

In Figure 3, we use different color to represent different *k*. As expected, $H_{1,n}$ tiled up the 2D plane.



$H_{1,2}$ $H_{1,3}$ $H_{1,4}$ $H_{1,5}$

Figure 3. Tessellation with Hilbert curves of different orders

B. *Superimposed Sierpinski Curves*

Sierpinski curve named after mathematician Wacław Franciszek Sierpiński, is also a well-known recursive curve and space filling curve. Different to Hilbert curve, Sierpinski curve is a closed curve designed by four basic patterns such as pattern *A*, *B*, *C*, and *D*, and a recursive schema called pattern *S* shown in Figure 4. Four straight lines of pattern *S* connect the other four patterns that are needed to be rotated for 90 degree. S_i denotes Sierpinski curve of order *i*. Figure 5 gives Sierpinski curve of order *1*, *2*, and *3*. S_1 looks like a leaf block. When the order *i* is increased, S_1 of different sizes are drawn on the outer corners of the order *i*, and the curve becomes more complicated and beautiful [3].

We also use a superimposition of Sierpinski curves of different orders to further fill the space of *(-1,1)*. $S_{1,n}$ denotes Sierpinski curves of different orders from order *1* to order *n*. For order *k*, $1 \le k \le n$, the length of a curve segment to be drawn is $\frac{1}{2^{k+1}}$, and (x_0,y_0) is the coordinate of the starting point of S_k as shown in Figure 5. The tessellation created with $S_{1,n}$ is shown in Figure 6.

$$x_0 = -1 + \frac{1}{2^k}, \quad y_0 = 1 - \frac{1}{2^{k+1}} \tag{2}$$

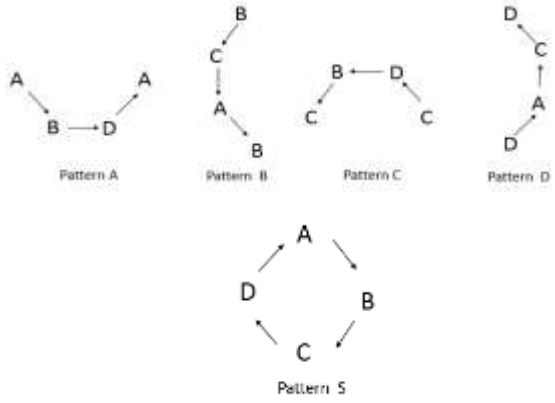
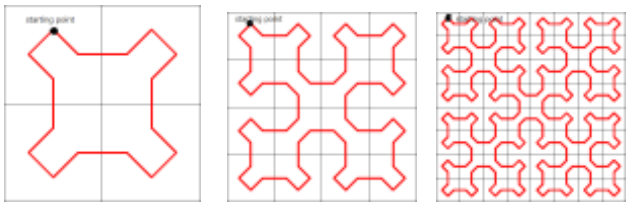
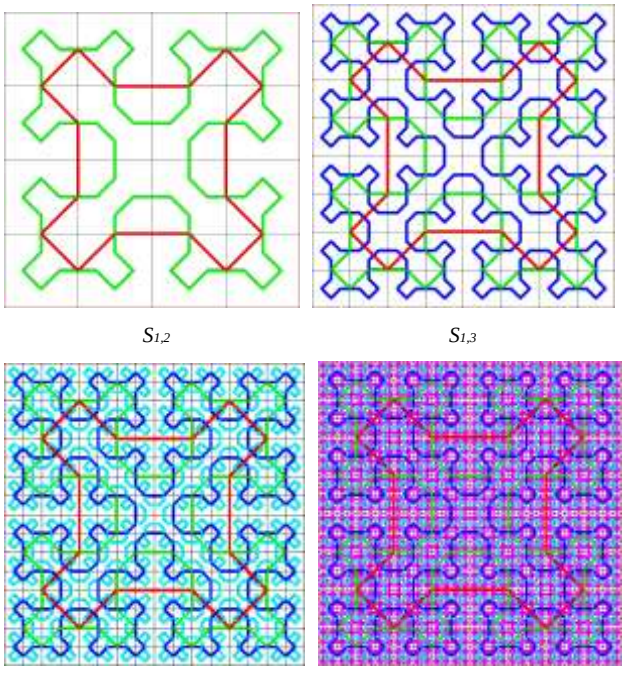


Figure 4. Patterns of Sierpinski curve



S_1 S_2 S_3

Figure 5. Sierpinski curve of order 1, 2, and 3



$S_{1,2}$ $S_{1,3}$ $S_{1,4}$ $S_{1,5}$

Figure 6. Tessellaton with Sierpinski curves of different orders

Proc. of the Eighth Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2019

ISBN: 978-1-63248-165-8 DOI : 10.15224/978-1-63248-165-8-06

c. Discussions about Fill Rate

The fill rate denotes the proportion of the area filled with the superimposed curves in the plane of *(- 1, 1)*. It is obvious the higher the order, the larger the filled area, which is proved with our validation experiment data shown in Figure 7, where the width of line is *0.02*. From Figure 7 we can see that the superimposed Hilbert curves and Sierpinski curves has better fill rate than their single curve. In particular, $S_{1,n}$ has superior exponential filling properties. The approximate functions of filling rate of the two superimposed curves are given as follows (3) and (4). For example, when $n = 6$, the fill rate of $H_{1,6}$ is *158%* and the fill rate of $S_{1,6}$ is *185%*. In fact, when the fill rate of a curve is *100%*, the plane of *(-1, 1)* will be filled without any gap at all. Theoretically, the recursive curve is infinite, so if you exceed the filling rate of *100%* you can imagine that the line segments collapse and overlap each other and the filled plane does not change visually. In our work, we concern the curves with fill rate below *100%* and pay attention to choosing an appropriate order *n*.

.

$$Fillrate(H_{1,n}) = 0.76e^{0.89n} \quad (3)$$

$$Fillrate(S_{1,n}) = 1.72e^{0.78n} \quad (4)$$

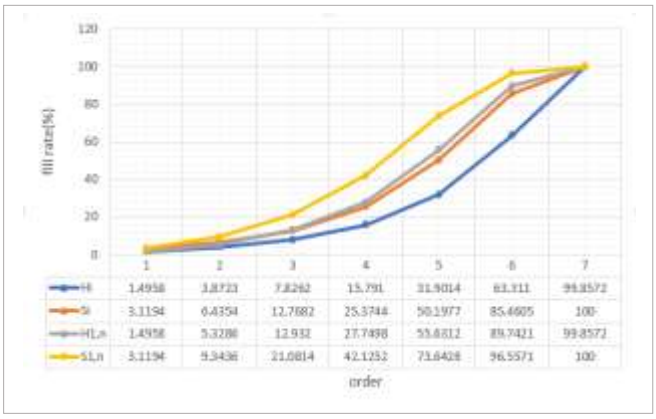


Figure 7. Comparison of fill rate of four kind of tessellations (*w=0.02*)

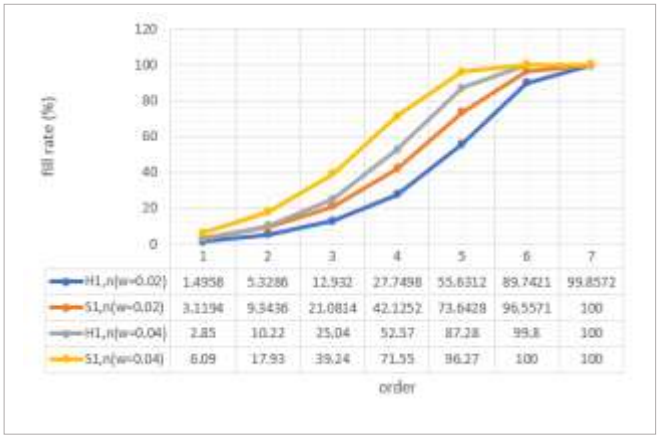
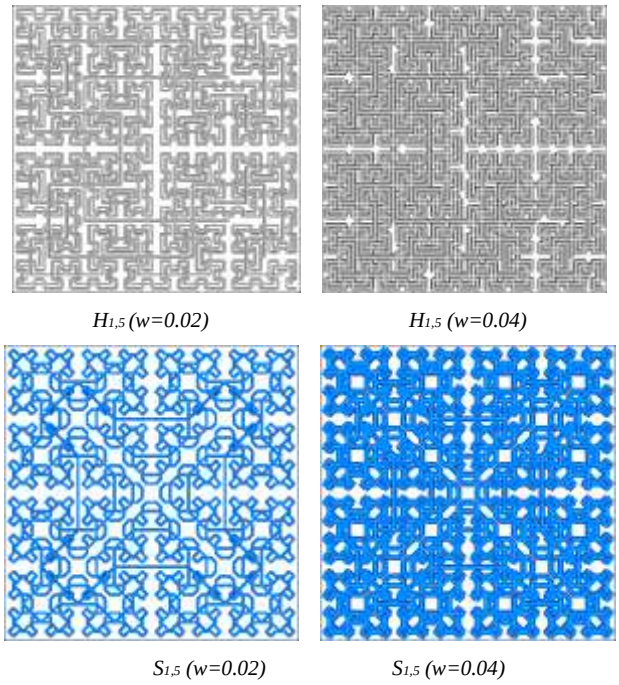


Figure 8. Comparison of fill rate of *w=0.02* and *w=0.04*

On the other hand, the line width used to draw the curve also plays an important role in the calculation of the fill rate. Our results of experiments in Figure 8 tell us that doubling the width of the line segment will double the filling rate. Figure 9 presents a comparison of the relationship between the line width and rendering effect for a same pattern. We say designs can change as line widths and fill rates change, so the possibility of selecting patterns based on personal preferences can be extended.



$H_{1,5}$ *(w=0.02)* $H_{1,5}$ *(w=0.04)*

$S_{1,5}$ *(w=0.02)* $S_{1,5}$ *(w=0.04)*

Figure 9. Comparison of line width and rendering effct

As the order increases, the number of lines drawn increases exponentially, the time required also increases exponentially. The time required for filling was measured in the experimental environment of TABLE I. The results are shown in the Figure 10. Rendering $H_{1,n}$ and $S_{1,n}$ takes time (5) and (6).

$$T(H_{1,n}) = 0.38e^{0.86n} \quad (5)$$

$$T(S_{1,n}) = 0.35e^{1.07n} \quad (6)$$

TABLE I. EXPERIMENTAL ENVIRONMENT

Operating System	Windows 10 Home 64-bit
Processor	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz (8 CPUs), ~3.6GHz
Memory	8192MB RAM
GPU	NVIDIA GeForce GTX 1060 6GB
Display Memory	10122 MB

Proc. of the Eighth Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2019

ISBN: 978-1-63248-165-8 DOI : 10.15224/978-1-63248-165-8-06

Figure 10. Time for tessellations

III. Construct 3D Fractal Models

In order to construct 3D fractal models by applying the above tessellation algorithm, we have developed a rendering technique to apply tessellated $H_{1,n}$ and $S_{1,n}$ to the surface of primitive models such as Riemann sphere, cylinder, cube, cone, and torus. Here we explain our approaches and methods.

A. *Rendering 3D Curves on Model*

In the previous examples, the curve was drawn by connecting multiple line segments to the curve, but to draw the curve on the surface of the 3D model, we would prefer to connect the coordinates through a slender cylindrical object instead of a straight line, we call it "joint yarn".

Polygons are typically used to render 3D objects on 3DCG. Curved surfaces of objects such as spheres can also be drawn using subdivided triangles or quadrilateral polygons. Although the elongated cylinder, joint yarn, we created here also has curved sides, it is also possible to create the necessary polygons by specifying the coordinates of multiple vertices on the circumference of the cylinder, and to simulate the surface by vertical segmentation.

When the height of the cylinder is *h*, the radius of the bottom and top of the cylinder is *r*, and the number of divisions of the circumference is *T*, the vertex coordinates of the bottom of the cylinder are

$$(r\cos(360\frac{t}{T}),\ r\sin(360\frac{t}{T}),0)$$

and the vertex coordinates of the top of the cylinder are

$$(r\cos(360\frac{t}{T}),\ r\sin(360\frac{t}{T}),h)$$

where $0 \le t \le T$.

Operations such as scaling, rotation, translation, etc. are performed to concatenate the coordinates of two points, (x_1, y_1, z_1) and (x_2, y_2, z_2) on the recursive curve.

First, scaling, to change the height of the cylinder to match the height of the cylinder to the distance between the two points on the curve. OpenGL's glScaled function has three magnification parameters in the *x*-axis, *y*-axis, and *z*-axis directions. Since the cylinder is mounted in the positive direction from the origin along the *z*-axis, the magnification in the *z*-axis direction must be set. The distance *d* between the two points can be expressed as (7). The enlargement ratio of the *z* axis can be obtained as *d* / *h* obtained by dividing the distance between two points by the height of the cylinder.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (7)$$

Second, rotation, since the cylinder is installed in the positive direction along the *z* axis from the origin, it is necessary to rotate the central axis of the cylinder and make it parallel to the vector of the line segment connecting the two points. We find the angle to rotate with the rotation axis and use OpenGL's glRotated function to do rotation. The central axis of the cylinder is parallel to the *z* axis, hence the unit vector of the cylinder is $\boldsymbol{a} = (0,\ 0,\ 1)$ and the unit vector of the line segment vector is

$$\boldsymbol{b} = (\frac{x_1 - x_2}{d},\ \frac{y_1 - y_2}{d},\ \frac{z_1 - z_2}{d}). \quad (8)$$

Angle to rotate θ can be obtained by (9).

$$\theta = cos^{-1}(\boldsymbol{a} \cdot \boldsymbol{b}) = cos^{-1}(\frac{z_1 - z_2}{d}) \quad (9)$$

Finally, translation, we use OpenGL's glTranslated function to move the cylinder and connects the two points. glTranslated function has three arguments of movement amount in *x* axis, *y* axis, *z* axis direction. Scaling and rotation of the joint yarn are completed in the above steps 1 and 2, if the movement amount in the *x* axis direction is x_1, the movement amount in the *y* axis direction is y_1, and the movement amount in the *z* axis direction is z_1, the two point can be connected by a joint yarn.

B. *Mapping Algorithm*

We designed an algorithm to create a new three-dimensional fractal model by constructing the above-mentioned fractal plane on the surface of a three-dimensional model with a specified shape.

- Obtain the appropriate point coordinates belonging to the either Hilbert curve or Sirerpinski curve according to the recursive algorithm of the Hilbert curve or the Sherpinski curve. Adjust the acquired coordinates in order to superimpose curves of different orders.
- Map the coordinates of the recursive curve to the mathematical formula of the surface of the 3D

Proc. of the Eighth Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2019

ISBN: 978-1-63248-165-8 DOI : 10.15224/978-1-63248-165-8-06

model and convert them to 3D coordinates. Interpolate coordinates when the distance between two points on the 3D model is expanded to a large extent.

- Render between the coordinates with the joint yarn.
- Perform above operations on the recursive curves from order *1* to order *n*.

C. *Developed Models*

We have developed some new approaches to construct the tessellating Hilbert curves and Sierpinski curves on the surface of primitive models such as Riemann sphere, cylinder, cube, cone, and torus. These three-dimensional fractal models show their unique style and good shape. Some 3D printed models are also presented here.

- Riemann sphere provides a method to project a plane to a half-sphere by (10), where (X, Y) is the coordinate of the space filling curves, $-1 \le X \le 1$ and $-1 \le Y \le 1$, and (x, y, z) is the coordinate on the sphere. The developed Riemann sphere models are shown in Figure 11 and 12, a 3D printed model is shown in Figure 13.

$$x = \frac{2X}{X^2+Y^2+1}, \quad y = \frac{2Y}{X^2+Y^2+1}, \quad z = \frac{X^2+Y^2-1}{X^2+Y^2+1} \qquad (10)$$

$H_{1,5}$ *(r = 0.015)*

$S_{1,5}$ *(r = 0.013)*

Figure 11. Fractal Riemann spheres

Figure 12. Fractal Riemann sphere whth $S_{1,3}$ *(r = 0.05)*

Figure 13. 3D printed model of Figure 13

- The coordinates *(x, y, z)* on the surface of a cylinder can be obtained by (11) where (X,Y) indicates the coordinate on the recursive curve, $-1 \le X \le 1$ and $-1 \le Y \le 1$. We provide some cylinder models with different *r* and orders in Figure 14 and 15.

$$x = sin(\pi X), \quad y = Y, \quad z = cos(\pi X) \qquad (11)$$

Figure 14. Fractal cylinders with $S_{1,6}$ *(r = 0.005)*

Proc. of the Eighth Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2019

ISBN: 978-1-63248-165-8 DOI : 10.15224/978-1-63248-165-8-06

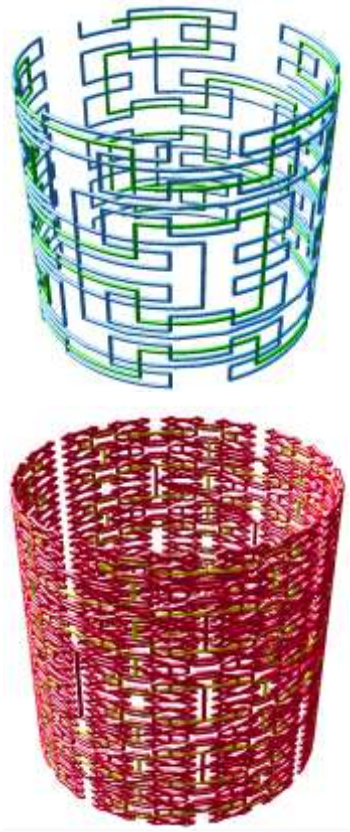


Figure 15. Fractal cylinder with $H_{1,4}$ *(r = 0.014)* and $S_{1,5}$*(r = 0.014)*

- For a fractal torus model, we can map (*X*, *Y*) to the surface of a 3D torus by (12), $-1 \le X \le 1$ and $-1 \le Y \le 1$. A 3D printed $S_{1,3}$ torus is shown in Figure 16.

$$\begin{aligned} x &= cos(\pi Y)\big(sin(\pi X) - 1.\ \ 5\big), \\ y &= \sin(\pi Y)\big(\sin(\pi X) - 1.\ \ 5\big), \\ z &= \cos(\pi X) \end{aligned} \tag{12}$$

Figure 16. 3D printed fractal torus with $S_{1,3}$ *(r=0.05).*

D. *Discussions about 3D Filling Effect*

We have mapped a tessellated $H_{1,n}$ or $S_{1,n}$ to the surface of a 3D model, in other words, we have knitted a fractal plane onto the surface of a three-dimensional object.

Generally, in many cases, the area of the plane (-1, 1) is not equal to the surface area of the object to be mapped. For convenient, we give an example of the cylinder model. When the side length of a square is *2R*, its area is $4R^2$. However, when the radius of a cylinder is *R* and its height is *2R*, the surface area of the cylinder is $4\pi R^2$ that is π times larger than the square. The area that will be created on the surface of the 3D model is larger than the area we prepared beforehand, therefore, in this case, mapping the tessellated plane to the 3D model requires some adjustments, including the use of higher order curves as we have done in the instances of Figure 14 and 15. We investigated the fill rate of the three-dimensional surface again, Figure 17 gives the results of our tests.

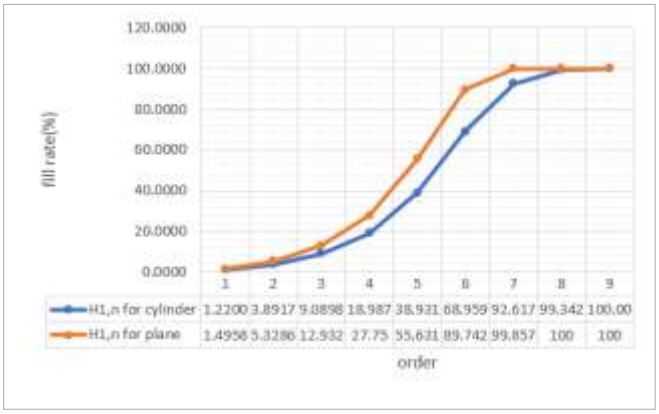


Figure 17. Comparison of fill rate of 3D fractal cylinder with $H_{1,n}$

In Figure 18 we provide some results of verification experiment for rendering time. We can see that for the same pattern the shape of the 3D model does not affect the drawing time. Even if we use an order that achieves a fill rate of *100%*, the rendering time can be within an acceptable range.

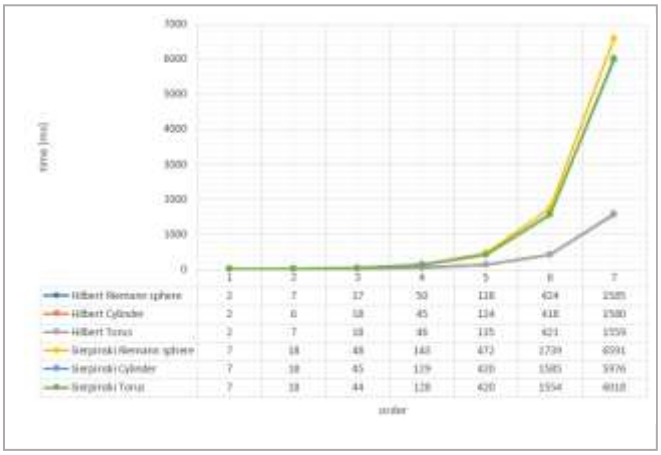


Figure 18. Time for rendering of 3D fractal models

Proc. of the Eighth Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2019

ISBN: 978-1-63248-165-8 DOI : 10.15224/978-1-63248-165-8-06

IV. Conclusion

The proposed algorithm and method for creating a fractal 3D model using a tessellation patterns of superimposed Hilbert curves and Sierpinski curves provides a means to extend the potential of 3D design. Tessellating fractal mapping onto the surface of primitive models gave us significant hints to expand and apply our methods to models having arbitrary shapes in our future work.

References

[1] Ningping Sun, Fuko Nakagami, Complex Mapping of 3DCG Models with Julia Set and Mandelbrot Set, International Journal of Advancements in Electronics and Electrical Engineering, IRED, USA, Volume 6, issue 1, pp.37 - 41, Apr. 2017.

[2] Ningping Sun, Ryo Miyazaki, Naoki Yoshida, Complex Mapping with the Interpolated Julia Set and Mandelbrot Set, Technical Sketches and Poster, ISBN 978-1-4503-0439-9/10/0012, SIGGRAPH Asia 2010, Korea, Dec. 2010.

[3] Niklaus Wirth, Algorithms and Data Structures, pp.90-94, Prentice-Hall, 1986.

[4] Sagan, H., Space Filling Curves, Springer-Verlag, New York, 1994.