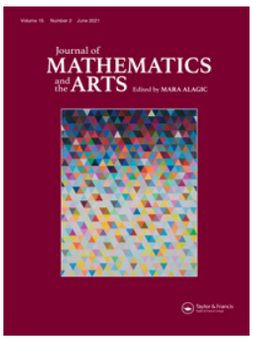


Journal of Mathematics and the Arts

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tmaa20

Iterated inversion system: an algorithm for efficiently visualizing Kleinian groups and extending the possibilities of fractal art

Kento Nakamura

To cite this article: Kento Nakamura (2021) Iterated inversion system: an algorithm for efficiently visualizing Kleinian groups and extending the possibilities of fractal art, Journal of Mathematics and the Arts, 15:2, 106-136, DOI: 10.1080/17513472.2021.1943998

To link to this article: https://doi.org/10.1080/17513472.2021.1943998

Published online: 02 Jul 2021.

JOURNAL OF MATHEMATICS AND THE ARTS
2021, VOL. 15, NO. 2, 106–136
https://doi.org/10.1080/17513472.2021.1943998

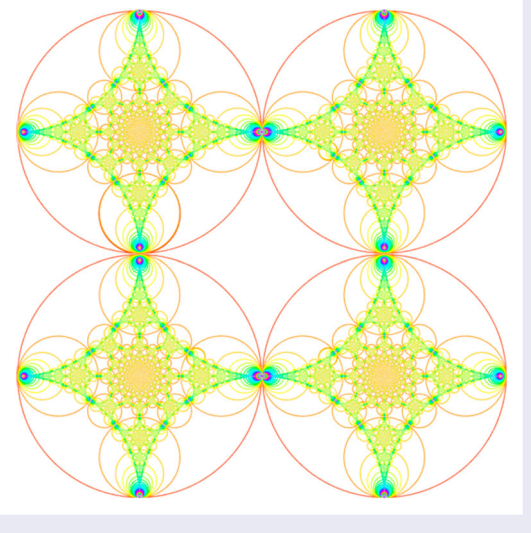
Iterated inversion system: an algorithm for efficiently visualizing Kleinian groups and extending the possibilities of fractal art

Kento Nakamura

Graduate School of Advanced Mathematical Sciences, Meiji University, Tokyo, Japan

ABSTRACT

Kleinian group theory is a branch of mathematics. A visualized Kleinian group often presents a beautiful fractal structure and provides clues for understanding Möbius transformations the mathematical properties of the group. However, it often takes much time to render images of Kleinian groups on a computer. Thus, we propose an efficient algorithm for visualizing some kinds of Kleinian groups: the Iterated Inversion System (IIS), which enables us to render images of Kleinian groups composed of inversions as circles or spheres in real-time. Real-time rendering has various applications; for example, the IIS can be used for experimentation in Kleinian group theory and the creation of mathematical art. The algorithm can also be used to draw both two-dimensional and three-dimensional fractals. The algorithm can extend the possibilities of art originating from Kleinian groups. In this paper, we discuss Kleinian fractals from an artistic viewpoint.



ARTICLE HISTORY
Received 11 August 2020
Accepted 14 June 2021

KEYWORDS
Fractals; visualization; Kleinian groups; circle inversion; sphere inversion

CONTACT Kento Nakamura somaarcr@gmail.com soma-arc.net

1. Background

1.1. Emergence of Kleinian group theory and fractals

The Kleinian group is a discrete subgroup of Möbius transformations. Felix Klein, Robert Fricke, and Henri Poincaré studied the Kleinian groups in the nineteenth century. The Kleinian group is deeply related to hyperbolic geometry and a geometrical concept that has attracted much attention from the beginning.

Klein himself was also deeply interested in the figures of fractals that originated from the Kleinian group. Fricke's students drew the Kleinian fractals by hand. These images are evidence of their quest to study Kleinian groups.

Regarding figures of fractals, Benoit Mandelbrot first used the term *fractal* in 1975. He also advocated the Mandelbrot set, and its wonderful and beautiful graphics were well known. Then, a broad interest in the images of fractals grew. People with computers enjoyed their interesting shapes and colours. Additionally, many programmers developed renderers for the Mandelbrot set or other fractals with their computers.

Various fractals and their rendering methods were introduced with the growth of computer technology. There are some internet communities devoted to discussing fractals. For example, there is a bulletin board system about fractals called *Fractal Forums*.[1] This forum is one of the largest communities around fractals. The participants display rendered fractals or compete with each other to make them, discuss the composition of the new fractals, and so on. Famous three-dimensional fractals called *Mandelbulb* and *Mandelbox* have been published in this forum. Of course, fractals based on Kleinian groups are also discussed.

In 2002, David Mumford, Caroline Series, and David Wright published a book called 'Indra's Pearls' (Mumford et al., 2002). It is not a technical mathematics book, but it is written for the general public. The book also introduces how to render beautiful fractals related to Kleinian groups using a computer. Then, not only mathematics lovers but also programmers became interested in the images of Kleinian groups. However, it is not easy to render all of the Kleinian fractals in real-time using the methods introduced in Indra's Pearls. Additionally, the book addresses only a small part of Kleinian groups.

1.2. Iterated inversion system and fractal rendering

The visualization of geometry became more artistic after the publication of Indra's Pearls because of its beautifully rendered images. Thus, from the perspective of art, geometrical objects morph through user manipulations or randomness, as in generative art. Because such a geometric art often requires the real-time reaction of manipulations, it is necessary for us to implement systems that render images in real-time. Therefore, the problem of real-time rendering for geometry and fractals involves subjects from arts to computer science.

In the author's papers, Nakamura and Ahara (2016, 2017, 2018), we propose a new algorithm called the *Iterated Inversion System (IIS)* for rendering fractals depending on inversions in circles or spheres. Then, we argue that the algorithm contributes to rendering fractals based on Kleinian groups in real-time. The details of IIS are written in Section 4. The author has published interactive web applications using the IIS. They are shown in Section 5.

On the other hand, from a mathematical perspective, the author's web applications are dynamic geometrical visualization tools. They succeed in extracting visual geometric

features from parabolic, elliptic, and loxodromic (hyperbolic) Möbius transformations. These are the classes of Möbius transformations. They are discussed in detail in Section 2.2.

It is useful to have a simple simulator for the tiling patterns or limit set generated by a finite generated Möbius transformation group. Such simulators are expected to support the discovery of new mathematical theorems and the study of Kleinian group theory. The IIS provides a good example of how information technology bridges mathematics and art through the rendering of fractals based on Kleinian groups.

1.3. Fractal rendering and the demoscene

To develop the IIS, we learned the rendering techniques used in the *demoscene*. The demoscene is a subculture of computer graphics. Demosceners aim to create beautiful movies and music in real-time with a small-sized programme. Furthermore, in a *demoparty*, demosceners gather and communicate with each other. They also show their works or compete. A demoparty is like an offline meeting among demosceners. There are famous demo parties called *Revision*, *The Gathering*, *Assembly*, and so on. The author also took part in a demoparty held in Tokyo, Japan, which is called *Tokyo Demo Fest (TDF)*;[2] it is the only demoparty in Japan.

Demosceners often use fractals in their artwork because fractals can generate complicated images from simple and short source codes. To visualize Kleinian groups in real-time, we refer to their techniques. For more details on the demoscene, see the documentary film 'Moleman 2 – Demoscene - The Art of the Algorithms (2012)'.[3]

The author took part in TDF and submitted OpenGL Shading Language (GLSL) artworks three times and PC 4K graphics once. Some of the author's works using IIS are highly appreciated. We discuss a number of the author's works in Section 5.4.

In order to render fractals using the IIS in real-time, we have to use parallel processing. To prepare a parallel processing environment easily, we use a fragment shader in the GLSL. For more details about shaders, read 'The Book of Shaders'.[4]

2. Preparation

This section shows some mathematical terms and prerequisites for understanding the IIS algorithm and the basic usage of the IIS.

In this paper, we use the terms for Kleinian groups used in Mumford et al. (2002). The word *group* represents an algebraic group. Additionally, a transformation group is an algebraic group consisting of transformations.

2.1. Möbius transformations

We mainly consider the cases of Möbius transformations of $\hat{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ or $\hat{\mathbb{R}}^3 = \mathbb{R}^3 \cup \{\infty\}$. Note that we generally consider Möbius transformations with orientation preserving transformations. However, in this paper, because we deal with inversions, we consider Möbius transformations as orientation reversing transformations. To distinguish them from the original Möbius transformation, we call them *extended Möbius transformations*.

In the two-dimensional cases, it is well known that extended Möbius transformations on $\hat{\mathbb{C}}$ are generated by circle inversions and line reflections. We often distinguish the inversion

from a line reflection map. However, in this context, by interpreting the line on the complex plane as a circle centred at infinity, we may regard the line reflection as a kind of inversion.

When we consider the IIS, we use the inner part of a circle and consider whether a given point is inside or outside of the circle. By interpreting the line on the complex plane as a circle centred at infinity, we also call it a *half-plane*. The half-plane is also taken to be inside for convenience.

In the same way as a circle inversion, we can consider a sphere inversion. Moreover, a sphere inversion can be extended to a plane reflection in the same way as a line reflection in the two-dimensional case. We also regard the plane reflection as a kind of inversion.

In this paper, we also consider the three-dimensional Möbius transformation on $\hat{\mathbb{R}}^3$. Three-dimensional Möbius transformations are deeply related to the four-dimensional hyperbolic space and four-dimensional hyperbolic manifolds.

The definition of the three-dimensional Möbius transformation is a simple extension of the two-dimensional version. A Möbius transformation is defined as a composition of sphere inversions and plane reflections. The plane on the three-dimensional space is a sphere centred at infinity; we call it *hyperplane*. To represent a three-dimensional Möbius transformation using a matrix, we consider a quaternion 2×2 matrix group called $Sp^k(1,1)$. Regarding this topic, refer to Section 2 of Sakugawa (2010).

Researching the structures of the Möbius transformation groups is closely related to studying hyperbolic geometry and hence hyperbolic manifolds. For more details, refer Marden (2016). The book introduces the Möbius transformation, the hyperbolic geometry, and the hyperbolic manifolds.

2.2. Classification of Möbius transformations

Excluding the identity map, we classify Möbius transformations into three types: *elliptic*, *parabolic*, and *loxodromic*. Here, $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, $ad-bc = 1$, $\tau_A = \operatorname{tr} A = \pm(a+d)$, and $z \in \hat{\mathbb{C}}$.

When A is elliptic, A is conjugate to $z \mapsto ze^{2i\theta}$, with $2\theta \not\equiv 2\pi$ and $\tau_A \in (-2,+2)$. The fixed points are two distinct points. See Figure 1. It shows an orbit of the elliptic Möbius transformation. The white-edged disk is a seed of an orbit. It has a circular orbit.

If A is parabolic, A is conjugate to $z \mapsto z+1$, with $\tau_A = \pm 2$ and $A \neq id$. The number of fixed points is one. The orbit of the parabolic transformation is shown in Figure 2. It converges to the fixed point.

When A is loxodromic, A is conjugate to $z \mapsto \lambda^2 z$, with $|\lambda| \neq 1$ and $\tau_A \in \mathbb{C}\backslash[-2,+2]$. The number of fixed points of A is two. The image of an orbit of loxodromic transformation is drawn in Figure 3. The orbit has a spiral structure and converges to two fixed points. One of the fixed points is called an attracting fixed point, and the other is called a repelling fixed point. This is because the orbit space of $f(z)$ of a general point z_0 has two limit points; one of the two is the limit of $f^n(z_0)$ for $n \to \infty$, and the other is the limit of $f^n(z_0)$ for $n \to -\infty$. Additionally, a loxodromic transformation whose trace is real is called a *hyperbolic* transformation. It is shown in Figure 4. It does not have a spiral structure, but it has two fixed points in the same as loxodromic transformations.

There are detailed classification tables in Figure 3.8 of Mumford et al. (2002) and Section 1.1 of Marden (2016).

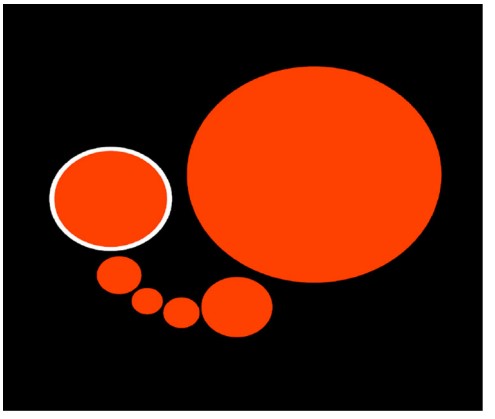


Figure 1. The orbit of an elliptic transformation.

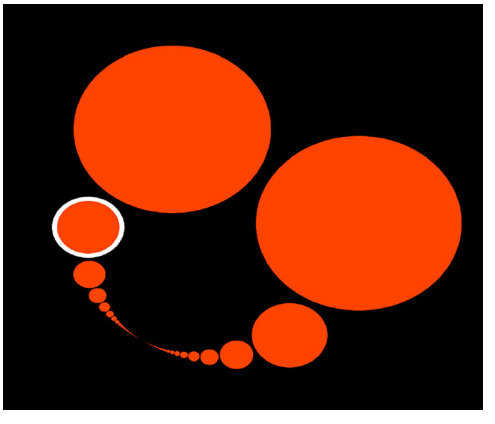


Figure 2. The orbit of a parabolic transformation.

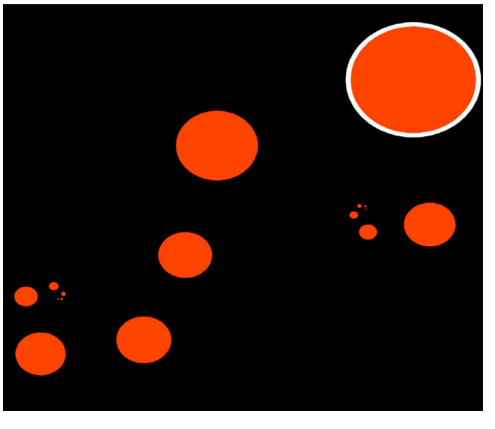


Figure 3. The orbit of a loxodromic transformation.

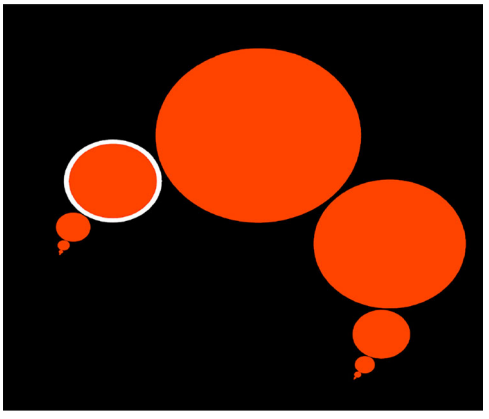


Figure 4. The orbit of a hyperbolic transformation.

The set of three-dimensional Möbius transformations also has a similar classification into six types. In the three-dimensional case, each type – 'parabolic', 'elliptic', and 'loxodromic' – has a modifier: *simple* or *compound*. Thus, there are six variations of classification types; for example, 'simple elliptic' or 'compound loxodromic', but 'simple' is often omitted. The classification of these six terms requires technical mathematics. Thus, for more details, see Section 2.2 of Sakugawa (2010).

2.3. Kleinian groups

A Kleinian group is a group named after Felix Klein. It follows that there exists a fundamental domain with positive volume in $\mathbb{H}^3$. Because a Möbius transformation yields an isometric transformation of $\mathbb{H}^3$, a Kleinian group yields a hyperbolic three-orbifold as a quotient space of the group. The basic properties of a Kleinian group are described in Section 2 of Marden (2016). Note that we call a discrete subgroup of the extended Möbius transformation group *an extended Kleinian group*.

Let G be a Kleinian group. For a general point $z_0 \in \hat{\mathbb{C}}$, a point $p \in \hat{\mathbb{C}}$ is a limit point of the orbit space of z_0 if and only if there exists a sequence $\{g_i\} \subset G$ such that $\lim_{i\to\infty} g_i(z_0) = p$. We call the closure of all limit points of an orbit space Gz_0 *the limit set* of G. The limit set is denoted by $\Lambda(G)$. Properties of the limit set are described in Section 2.4.1 of Marden (2016).

Concerning the limit set, there are concepts of an algebraic limit and a geometric limit. For more details, they are described in Sections 4.1 and 4.4 of Marden (2016).

3. Basic methods for visualization

In this section, we introduce basic methods for visualizing extended Kleinian groups.

In order to visualize a group, we consider a *Cayley graph*, as shown in Figure 5. It is composed of four inversions of circles. When we make a tiling of the group, we arrange the tiles along the words of the Cayley graph. The visualized image of the group composed of inversions of circles is shown in Figure 6. In the image, the disks nest infinitely. We call

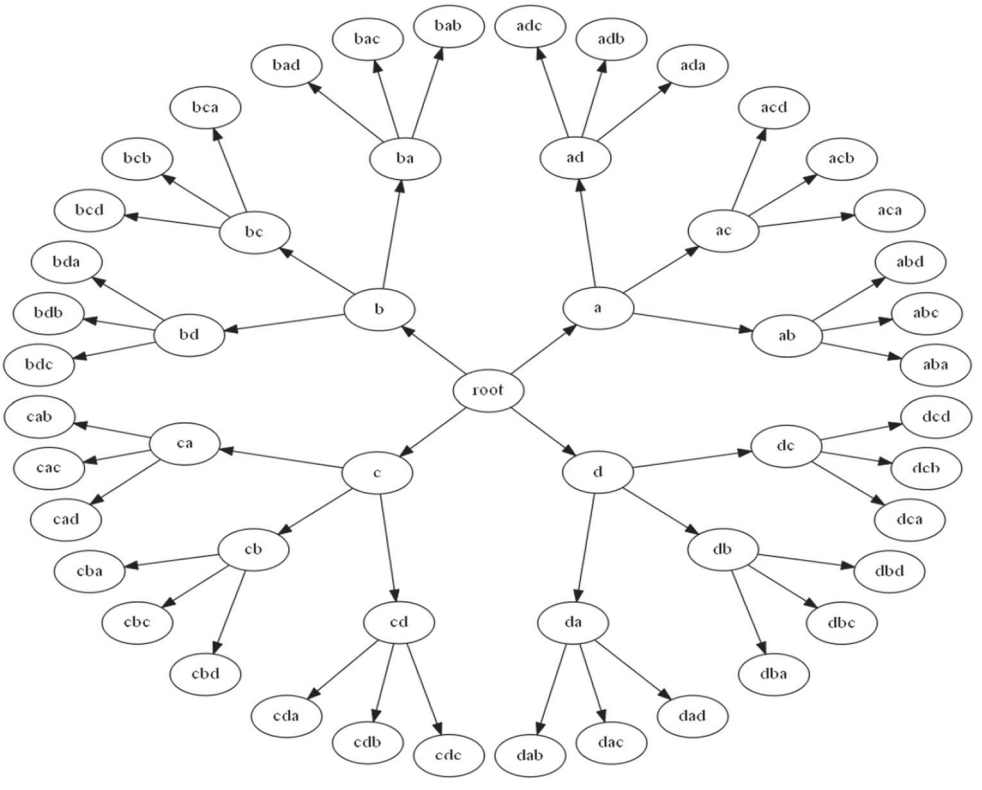


Figure 5. Cayley graph.

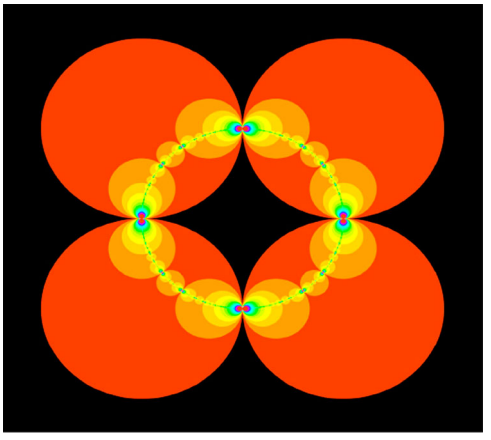


Figure 6. Orbit space of the disks.

the images *circle inversion fractals*. We draw the orbit space of the disks. Figures 7 and 8 show other examples of the fractals. For more details about circle inversion fractals, see the 'Introduction' of Nakamura and Ahara (2017).

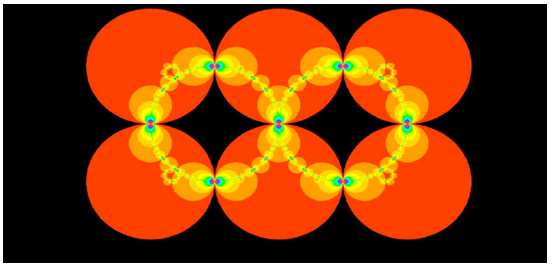


Figure 7. Circle inversion fractal composed of six disks.

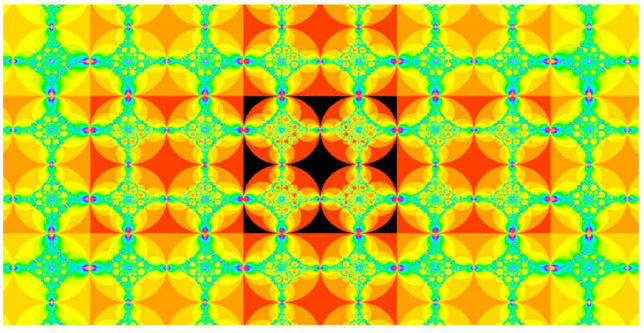


Figure 8. Circle inversion fractal composed of four disks and four half-planes.

To draw the orbit space of the disks within a short time, we use a *breadth-first search* algorithm. Additionally, there is another method of traversing the graph, called a *depth-first search* algorithm. Using this algorithm, we can draw only the limit set directly; see Figure 9. It shows an example of the limit set of a Kleinian group. In Indra's Pearls, this method is primarily used.

Here is another example: Keita Sakugawa introduced a family of four-dimensional Kleinian groups in his master's thesis. However, he did not publish his paper, and it is written in Japanese. The limit sets of the four-dimensional Kleinian groups have three-dimensional torsion, as shown in Figure 10. They are rendered using a depth-first search of the Cayley graph by the author.

Despite the attractiveness of the images, there are few examples of the families of four-dimensional Kleinian groups because of the complexity of the generators. The image of the limit set is more complicated than the group introduced in Indra's Pearls. The calculation of quaternions takes much time.

Visualization by graph traversal methods is easy for us to implement and understand. However, the methods have the drawbacks that the computational complexity to traverse the Cayley graph is easy to increase exponentially, and it takes too much time. For more details on graph traversal methods, see Sections 4 and 5 of Indra's Pearls.

About Figures 5, 6, and 9, refer Nakamura (2018).

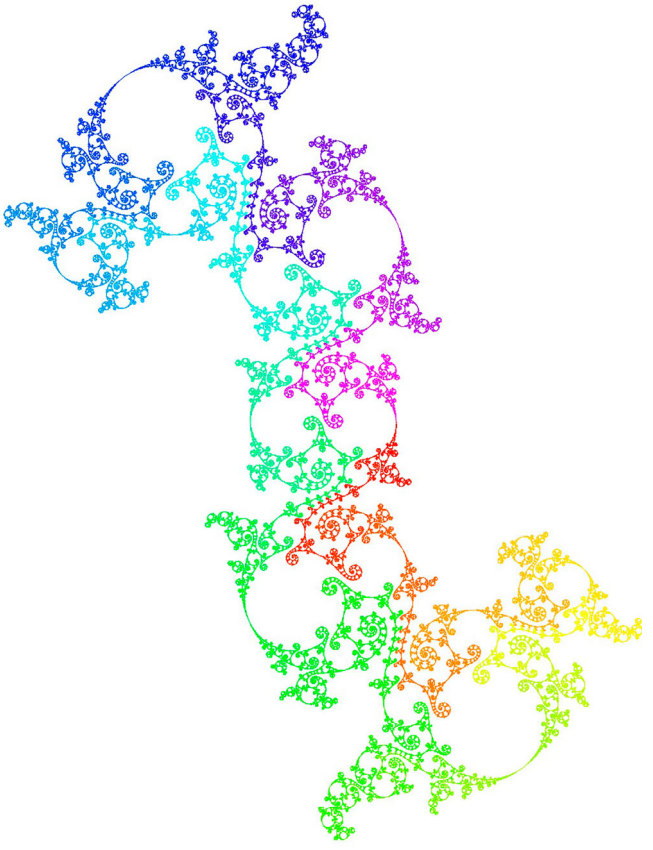


Figure 9. Limit set of the Kleinian group.

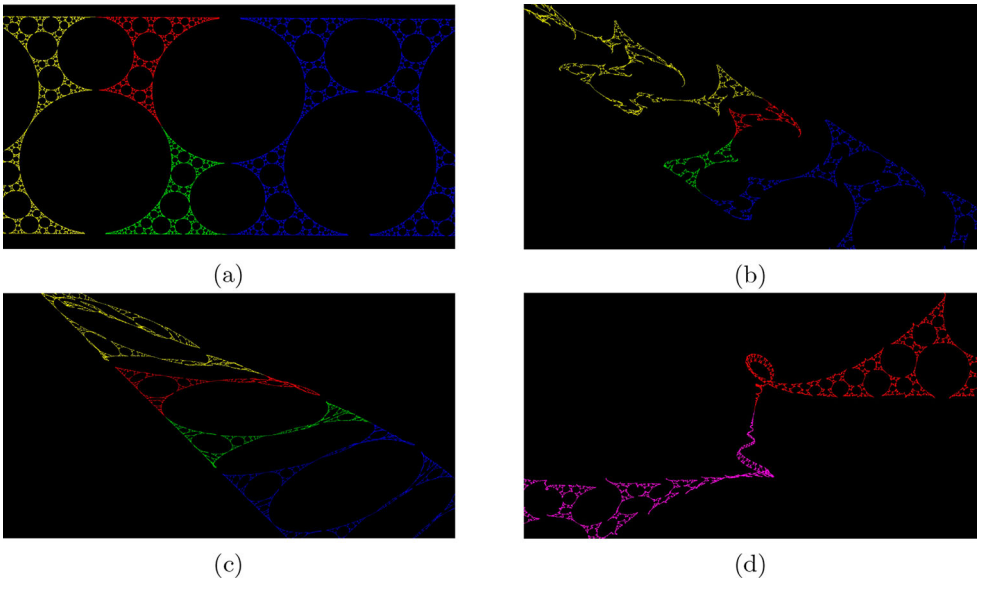


Figure 10. The limit set of the four-dimensional Kleinian groups.

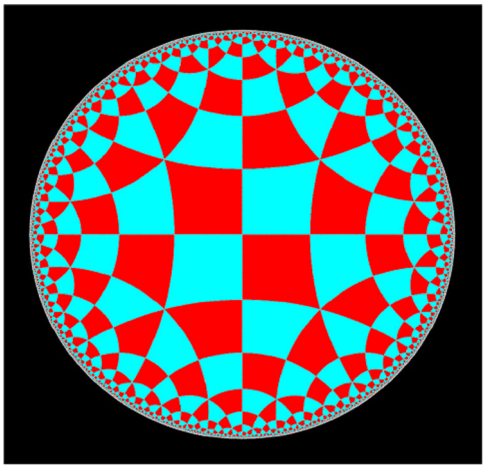


Figure 11. Hyperbolic tessellation.

4. IIS in two-dimensions and three-dimensions

4.1. Two-dimensional IIS

To solve the problems of graph traversal approaches, we focus on circle inversions and propose an efficient algorithm to visualize the fractals originating from circle inversions. This algorithm is called the *Iterated Inversion System (IIS)*. It can visualize not only two-dimensional fractals but also three-dimensional fractals originating from sphere inversions.

The IIS calculates the depth of the overlapping disks point by point. The process of the IIS is as follows: For each point on the plane, if the point is contained in one of the initial inversion disks, we apply inversion in the circle to the point. We continue iterating inversions until the point has moved to the outside of the initial disks. Finally, we colour the original point according to the number of iterations of inversions or choose a colour for the final point.

Mathematically, when the group is *an automatic group*, the algorithm will stop eventually. However, technically we have to predetermine the maximum number of iterations so that the number of iterations does not become too large.

The IIS has various applications other than circle inversion fractals. For example, we can render the hyperbolic tessellation shown in Figure 11 using the IIS. To render a hyperbolic tessellation with the IIS, we first prepare the fundamental first tile and the circles forming the edges of the tile. Figure 12 shows the first tile in the case of Figure 11. The tile is formed by four circles (two normal circles and two half-planes.)

In summary, we need a fundamental initial tile and a set of transformations that transform a point into the fundamental initial tile to use the IIS. The process of the IIS is the reverse of tiling. We simply apply circle inversions to the point. Additionally, the circle inversion fractal can be seen as a tiling of the black area in Figure 6.

The breadth-first search approach has to compute the centres and radii of circles and draw them. The number of circles increases exponentially. The number of circles is determined as follows: Let d be the depth of the reflections, and let n be the initial number of

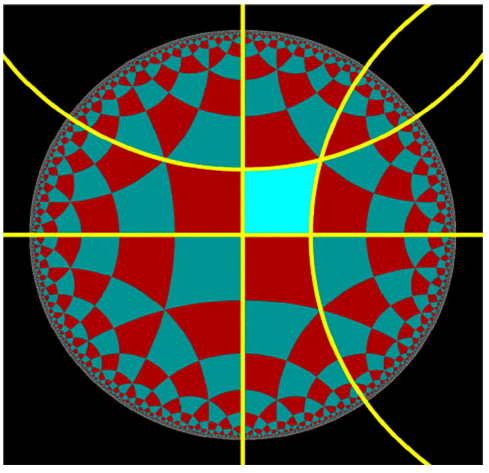


Figure 12. Fundamental tile.

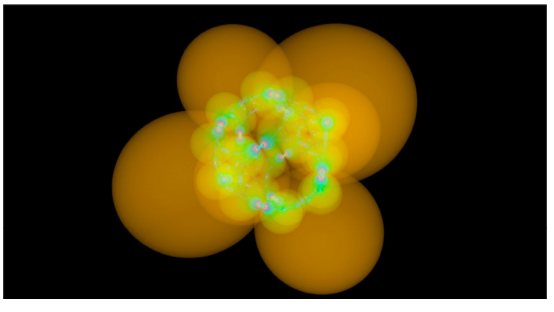


Figure 13. Volume-rendered nesting spheres

disks. The number of disks is calculated by $\sum_{k=1}^{d} n(n-1)^{k-1}$. On the other hand, the computational complexity of IIS is a polynomial order of the number of iterations and the number of inversion circles for each pixel. Each computation is atomic; thus, we can perform parallel processing for the IIS.

For more details about the two-dimensional IIS, see Nakamura and Ahara (2016), 'Iterated Inversion System' in Nakamura and Ahara (2017) or Section 3.2.1 of Nakamura (2018).

4.2. Three-dimensional extension

In a similar manner to the two-dimensional algorithm, we extend the IIS to visualize three-dimensional fractals related to Kleinian groups. We replace the circle inversions with sphere inversions in the IIS, and we compute the nesting depth of the spheres voxel by voxel using the IIS. In order to visualize the voxel data, there is a *volume rendering* algorithm. The volume-rendered sphere inversion fractals are shown in Figure 13. It has possibilities for artistic expression, but it is not easy to efficiently render the data, and the visualized images are not helpful in studying Kleinian groups.

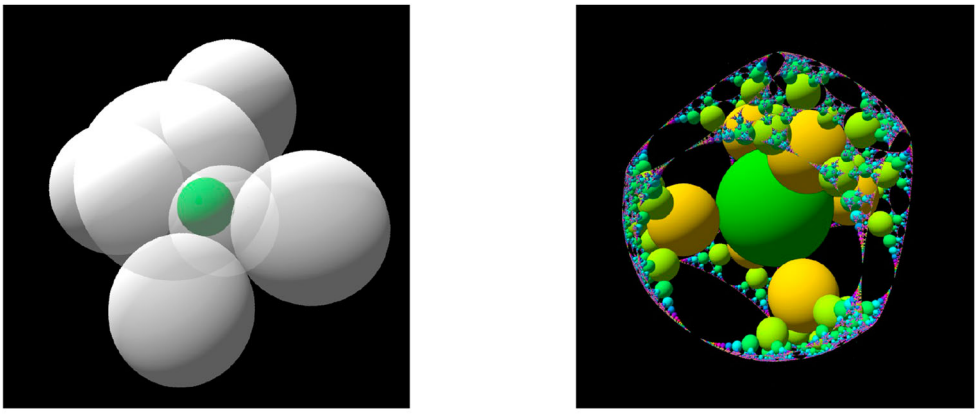


Figure 14. Sphere inversion fractal. (a) Generator and (b) The orbit of spheres.

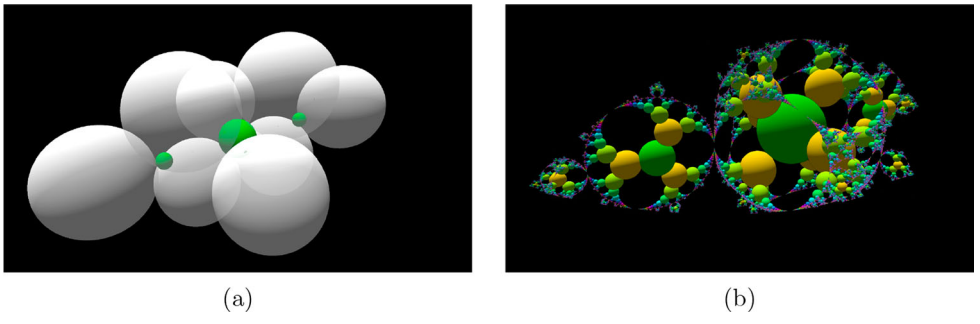


Figure 15. Another sphere inversion fractal. (a) Generator and (b) The orbit of spheres.

Therefore, we render the orbit of the spheres; see Figure 14(a), which shows six white inversion spheres and a green seed sphere. Figure 14(b) shows the orbit of the green seed sphere transformed by inversions in the white spheres. The smaller the sphere's radius, the closer to the limit set of the Kleinian group. Additionally, Figure 15 shows other generators and orbits. It has three seed spheres and eight inversion spheres.

4.2.1. Rendering technique

We use *ray tracing* to visualize three-dimensional objects. Ray tracing computes an intersection between a ray and objects algebraically. However, the fractal is composed of many spheres. Thus, it is difficult to compute all of the intersections. Therefore, we use the *sphere tracing* (Hart, 1996) algorithm, which is a kind of ray tracing technique.

In sphere tracing, we use a *distance function*. The distance function returns the minimum distance between a given point and the objects. We make the tip of the ray march along the direction by the distance.

However, regarding fractal rendering, it is not easy to obtain an actual distance from a given point to the objects. Thus, we use a lower estimated distance. We approximate the distance using a technique called *distance estimation*.

When we compute the distance to the orbit of the sphere and the tip of the ray, we apply the IIS to the tip of the ray using sphere inversions. Finally, we compute the distance between the seed sphere and the transformed tip of the ray. However, the inversion transformations may cause an error. Therefore, we use the Jacobian (Jacobian determinant) of the inversions. We accumulate the Jacobian by multiplying it for each inversion. Then, we divide the distance between the seed sphere and the transformed point at the outer area by the accumulated Jacobian. Finally, we obtain the approximate distance between the tip of the ray and the nearest sphere. For the above case, the distance is a lower bound for the spheres.

Additionally, the above computation is a rough estimate, which causes artefacts. For example, the transformed point is outside of the limit set, the distance is unintentionally long, and the ray can pass through the orbit of the spheres. To avoid artefacts, we shorten the estimated distance by multiplying by a scaling factor. Shortening the distance increases the number of steps of sphere tracing, but we can eventually obtain the intersection of the ray and the spheres.

For more details about three-dimensional IIS and pseudo-code, see 'Iterated Inversion System' in Nakamura and Ahara (2017) and Section 3.2.2 of Nakamura (2018). About Figures 11 and 14 refer Nakamura (2018) and Nakamura and Ahara (2017) respectively.

4.3. Related works

Aaron Montag introduced the texture-based approach in his bachelor's thesis (Montag, 2014). The algorithm is as follows: First, he prepares the initial seed disk in the texture. Next, he applies the generators of the group to each point on the plane. If the transformed point is on the seed disk or filled point, he fills the original point. He continues iterating the second process; he obtains an image of the limit set of the group. His implementation can be seen in the gallery of CindyJS.[5] However, the algorithm is difficult to extend in three-dimensions because three-dimensional voxel data require considerable computer memory. Additionally, it is difficult to draw enlarged images because they require more pixels.

Martin von Gagern and Jürgen Richter-Gebert introduced an algorithm called *Reverse Pixel Lookup* (von Gagern & Richter-Gebert, 2009). They aimed to render two-dimensional hyperbolic tessellations. This algorithm is similar to the IIS, but we aim to use the method to render Kleinian fractals and three-dimensional objects.

Vladimir Bulatov visualized H^3 by the tiling of the hyperbolic polyhedra.[6] The hyperbolic polyhedra can be tiled with inversions of spheres that form each surface of a polyhedron. However, the tiling converges to a sphere, so it is not a beneficent visualization. Thus, Bulatov tiled polyhedra by generators of the subgroup of the whole group. He selects a subset of the faces of the polyhedron and visualizes the subgroup of the group generated by sphere inversions by tiling. He succeeded in drawing various interesting subgroups using the method.

Michael Woodard et al. developed a VR simulation of three-dimensional hyperbolic space using sphere tracing.[7] They uses a sphere inversion approach to make distance function representing three-dimensional hyperbolic space H^3. They consider the cubical tiling of the ideal cube. The ideal cube has six spherical faces. They apply inversions in the spherical faces to the cube, and they obtain a cubical tiling of H^3. It is also shown in the website of '3-dimensional.space'.[8]

Roice Nelson and Henry Segerman worked on the visualization of tilings corresponding to Schläfli symbols (Nelson & Segerman, 2015). They call these tilings in three-dimension *honeycombs* and fundamental solids *cells*. They apply sphere inversions in each face of cells and tile the cells. Then, they can draw patterns on the surface of the sphere. By applying stereographic projection to the sphere, the patterns on the sphere can be transferred to the plane. Nelson et al. summarize these patterns for each Schläfli symbol.

David Bachman, Matthias Goerner, Saul Schleimer, and Henry Segerman introduced *cohomology fractals* (Bachman et al., 2020). Cohomology fractals are images generated by a hyperbolic three-manifold equipped with a cohomology class. They show Cannon–Thurston maps for *degenerate groups*. In order to render cohomology fractals, they use the ray casting algorithm pixel by pixel. There are examples of GLSL code to render cohomology fractals.[9] They are also rendered in real-time. The implementation of the algorithm is similar to IIS. They test whether the given point needs a Möbius transformation. Then, they apply the transformation to the point. They continue the process until the point is moved inside the fundamental domain.

5. Implementation examples

In this section, we show a variety of uses of the IIS.

5.1. Geometrical representation of Möbius transformations

Thus far, we have only used a simple circle inversion or sphere inversion. Varieties of images can be rendered using other types of Möbius transformations. As noted in Section 2.2, Möbius transformations are classified into three types. We compose the Möbius transformations with inversions.

We can change the parameters of Möbius transformations by operating on the positions or radii of circles or spheres. This enables us to create intuitive geometrical constructions. We also attach animated examples made with the GLSL in footnotes.

5.1.1. Two-dimensional generators

As described above, the IIS needs a fundamental initial tile and transformations that move a given point to the fundamental tile. In this paper, we call the transformation T.

First, see Figure 16(a), which shows hyperbolic transformations. The three overlapping disks represent the transformation. Let $C1$ be a red disk, let $C2$ be a green disk, and let $C1'$ be a blue disk. $C1$ and $C2$ are the parameters of the transformation. $C1'$ is $C1$ inverted by $C2$.

The fundamental area is changed to a green and blue area. Then, we have to put disks in the area; see Figure 16(b). The white-edged disk is transformed by inversions in $C1$ and $C2$.[10] T is composed of inversions in $C1$ and $C2$.

When the disks $C1$ and $C2$ are kissing at one point, as in Figure 17(a), the transformation becomes parabolic, as shown in Figure 17(b).[11]

When $C1$ and $C2$ are crossing, as in Figure 18(a), the transformation becomes elliptic. The orbit of the disks is rotated around the line composed of the crossing points $C1$ and $C2$; see Figure 18(b). Note that the angles crossing $C1$ and $C2$ should be rational angles; otherwise, the orbits of the disks will overlap each other.

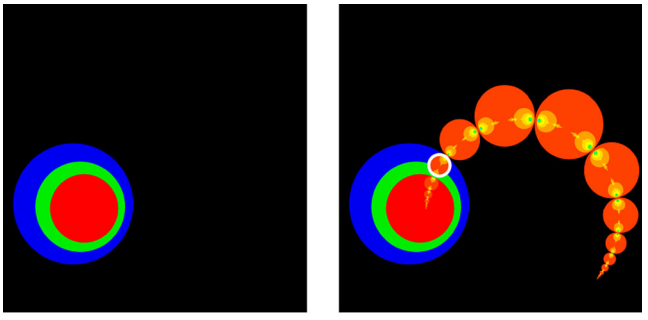


Figure 16. Hyperbolic. (a) Generator and (b) Orbit.

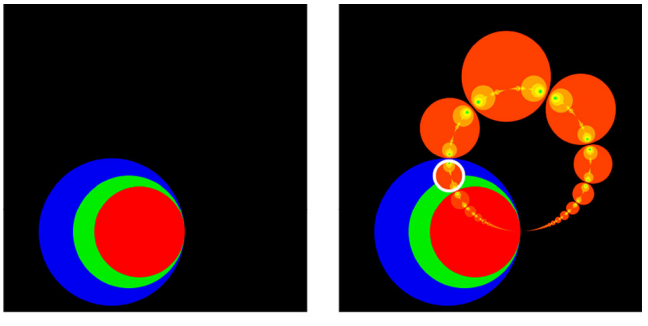


Figure 17. Parabolic. (a) Generator and (b) Orbit.

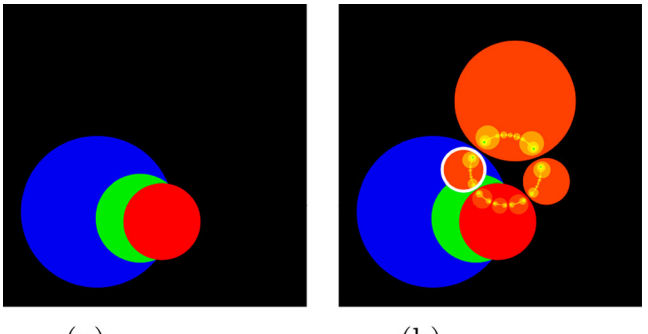


Figure 18. Elliptic. (a) Generator and (b) Orbit.

Figure 19 shows the loxodromic transformation. It is composed of four disks, a line, and a pink control point. The orbit is a spiral, as in Figure 19(b). The torsion is operated by the control point.[12] T is composed of inversions in $C1$, $C2$, the circumference of the yellow disk, and the reflection through the white line.

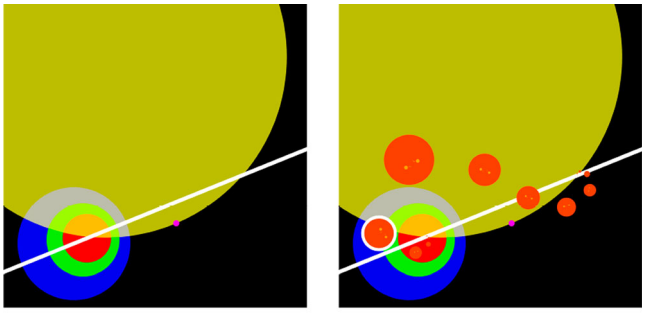


Figure 19. Loxodromic. (a) Generator and (b) Orbit.

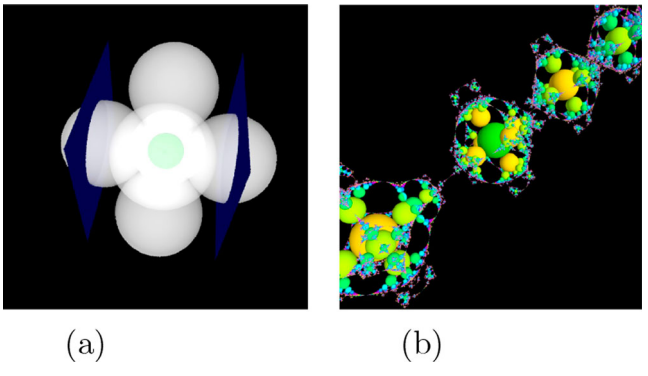


Figure 20. Parabolic (parallel translation). (a) Generator and (b) Orbit.

For more details of the implementation and other generators, read '2D Generators' of Nakamura and Ahara (2017) or Section 4.3.1 of Nakamura (2018).

5.1.2. Three-dimensional generators

In this section, we show Möbius transformations act in three-dimensions. A parallel translation is as shown in Figure 20. The blue plates are a part of the hyperplane. The plane applies a plane reflection. The two facing hyperplanes generate a parallel translation. A compound parabolic transformation adds a rotation around the translation axis, as in Figure 21.[13] The torsion of two hyperplanes represents the amount of rotation.

There are elliptic, loxodromic, and parabolic transformations similar to two-dimensional transformations. They are also composed of three spheres, as shown in Figure 22.[14]

In a three-dimensional Möbius transformation, the simple generators are similar to those of two-dimensional Möbius transformations. However, some compound loxodromic transformations have three-dimensional torsion, as in Figure 23.[15]

For more details of the implementation and other generators, see '3D Generators' of Nakamura and Ahara (2017) or Section 4.3.2 of Nakamura (2018). Figures 16–23 are modified images in Nakamura and Ahara (2017).

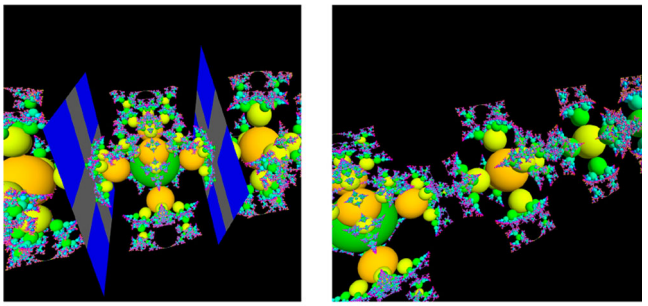


Figure 21. Compound parabolic. (a) Generator and (b) Orbit.

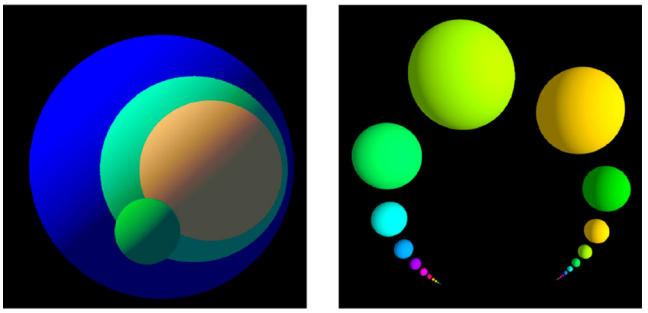


Figure 22. Loxodromic. (a) Generator and (b) Orbit.

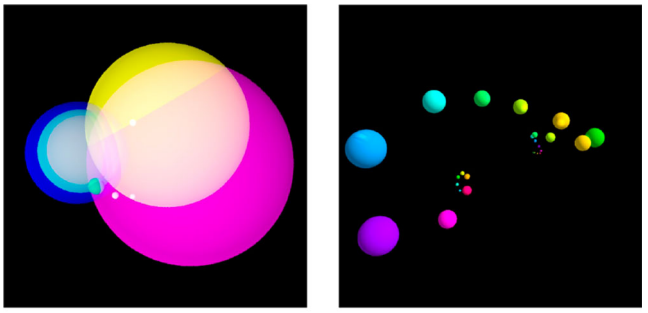
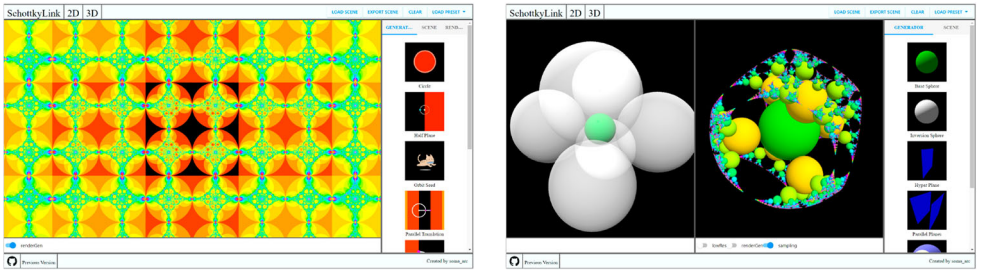


Figure 23. Compound loxodromic. (a) Generator and (b) Orbit.

5.1.3. Schottky link

The author is developing software called *Schottky Link*,[16] which visualizes two-dimensional circle inversion fractals and three-dimensional sphere inversion fractals quickly and intuitively.

The software has a two-dimensional mode, as shown in Figure 24(a), and a three-dimensional mode, as shown in Figure 24(b).



(a) (b)

Figure 24. Schottky link. (a) Two-dimensional mode and (b) Three-dimensional mode.

(a) (b)

Figure 25. Edge of the limit set of the circle inversion fractal.

In two-dimensional mode, we simply add a disk and generators selected from the bar on the right side. In three-dimensional mode, the software has two main panels. The left panel shows the generators of the group and the green seed of the orbit. The right panel displays the orbit of the green sphere. When we operate the generators in the left panel, the orbits of the spheres are deformed according to the generators.

Thanks to the IIS, we can manipulate many generators and render the images in real-time. It is easy to explore inversion fractals with Schottky Link.

However, Schottky Link cannot visualize all of the Kleinian groups. For example, the IIS cannot visualize a group that includes two or more loxodromic or parabolic generators because we cannot know that the transformations will move the point into the fundamental tile. Additionally, the IIS cannot render only the limit set.

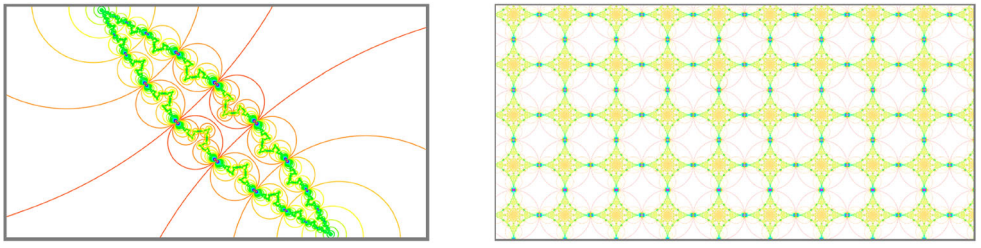
5.2. Simple visualization techniques

In this subsection, we show some simple techniques to render artistic images.

5.2.1. Render internal area

In two-dimensional circle inversion fractals, when all of the circles touch each other, the limit set divides the plane into two parts, as shown in Figure 25(a). The image is generated by four inversions of the circles.

Circle inversion preserves the interior or exterior part of the limit set. Thus, after applying the IIS, we fill a pixel when the transformed point is moved into an inner part of the



(a) (b)

Figure 26. Circumferences of the circle inversion fractal.

(a) (b)

Figure 27. The orbit of the cube.

fundamental tile, that is, the inner black area. Then, we obtain only the inner part of the circle inversion fractals. This is shown in Figure 25(b).

5.2.2. Rendering circles

In two dimensions, we can render, not the orbit of disks, but rather the orbit of circles using a Jacobian, as shown in Figure 26.[17] We can estimate the distances between the circumferences of the disks and the point on the initial disks. When we apply circle inversions, we multiply and accumulate the Jacobian of the inversions. When the transformed point is moved to the initial disks, we divide the computed distance between the circumference of the initial disk and the transformed point by the accumulated Jacobian. We then obtain the distance between the circumference of the initial circle and the final point.

5.2.3. Other seed of an orbit

For three-dimensional sphere inversion fractals, we cannot use a seed sphere, but we can use another object expressed by distance functions. For example, we can visualize the orbit of the cube; see Figure 27. The transformed cubes are distorted by sphere inversions.

5.3. Sphairahedra and three-dimensional fractals

Kazushi Ahara and Yoshiaki Araki invented a geometrical concept called *sphairahedron* in 2003 (Ahara & Araki, 2003). Sphairahedra are polyhedra that have spherical faces instead of planar faces. We can visualize three-dimensional tilings using inversions in the spheres that form the faces of the sphairahedron. We can also use IIS to render the tilings. Figure 28

Figure 28. Cube-type sphairahedron.

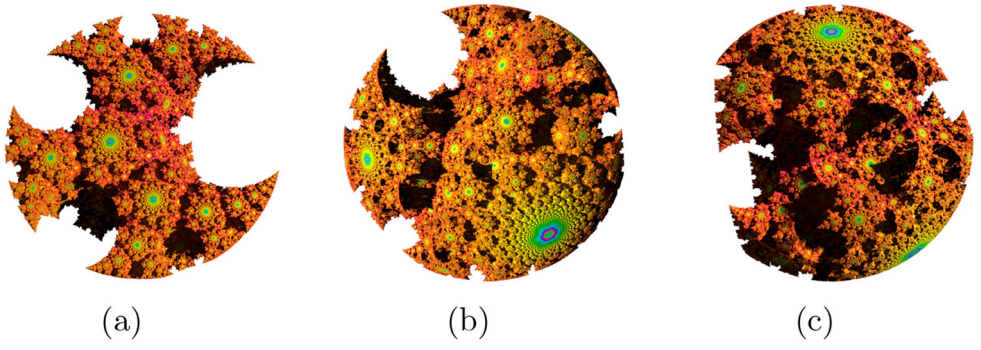


Figure 29. Images of a quasi-sphere rendered in different viewpoints.

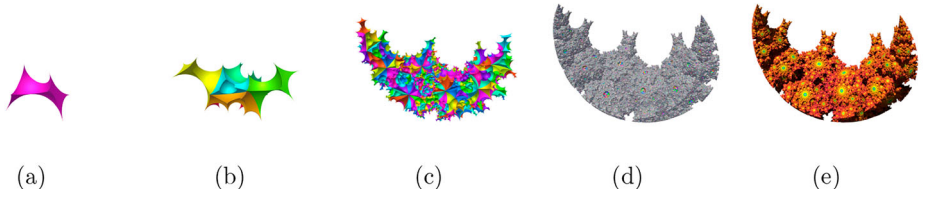


Figure 30. Tiling of the finite sphairahedron. (a) Step1. (b) Step2. (c) Step5. (d) Step20 and (e) Final image.

shows a cube-type sphairahedron, and Figure 29 shows the result of a three-dimensional tiling of a cube-type sphairahedron from three perspectives.

We also call the sphairahedron, whose all of the vertexes are finite, a *finite sphairahedron*. The union of all the tiles generated by the tiling is homeomorphic to a ball. Thus, its boundary is called a *quasi-sphere*. Quasi-sphere is the limit set of the Kleinian group. Figure 30 shows the tiling process for a finite sphairahedron.

Figure 31(a) shows a sphairahedron, one of whose vertexes is at infinity; this is called an *infinite sphairahedron*. The tiling process is shown in Figure 31. It generates the fractal terrain shown in Figure 31(e).

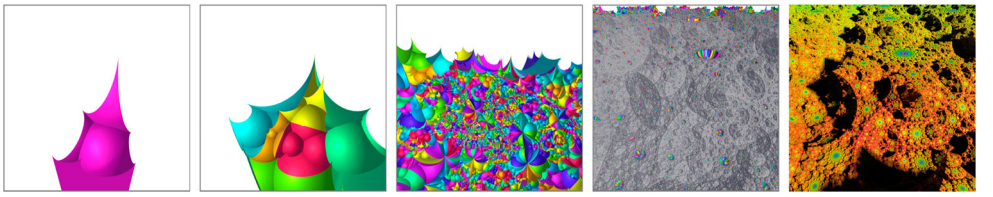


Figure 31. Tiling of the infinite sphairahedron. (a) Step1. (b) Step2. (c) Step5. (d) Step20 and (e) Final image.

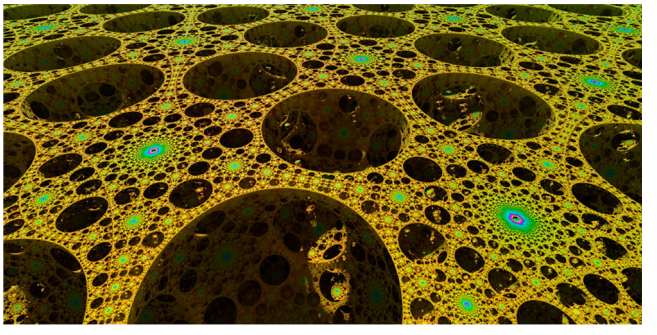


Figure 32. Bridges.

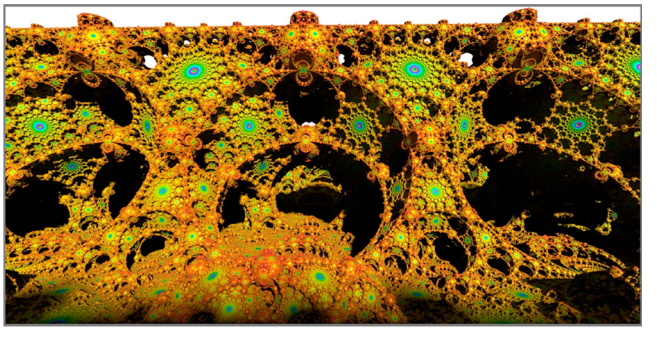


Figure 33. Tunnels.

A finite sphairahedron is generated by applying a sphere inversion to an infinite sphairahedron. We call the sphere an *inversion sphere*. The shape of the finite sphairahedron changes according to the position and radius of the inversion sphere.

A sphairahedron has two properties, called *ideality* and *rationality*. An ideal rational sphairahedron generates the Kleinian group. Therefore, we compute the parameter space of the ideal rational sphairahedron. The parameter spaces of various sphairahedra are shown in Nakamura et al. (2020). Mathematically, we are not interested in outside of the parameter space. However, from an artistic perspective, the quasi-sphere outside of the parameter space is also interesting. It can make fractal bridges or tunnels in the resulting fractals, as shown in Figures 32 and 33.

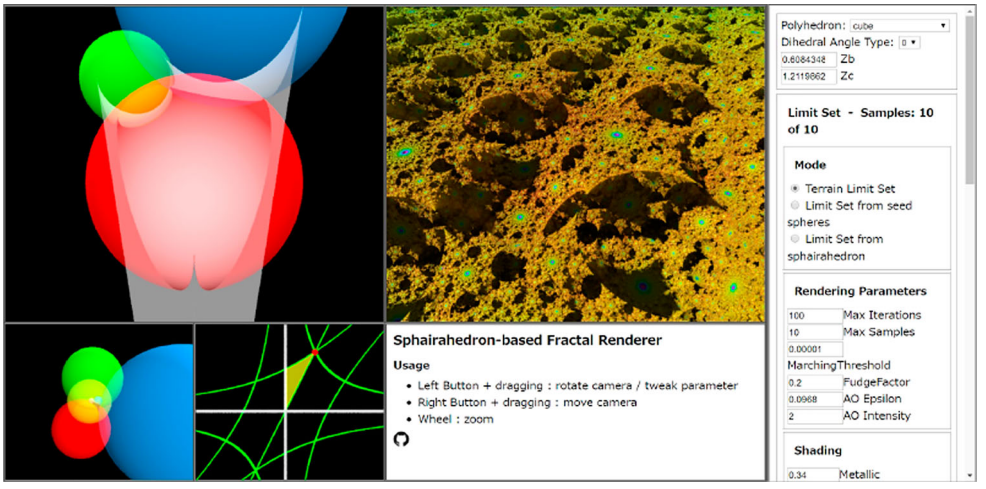


Figure 34. Sphairahedron-based fractal renderer.

The author is developing the renderer for sphairahedra and its fractals.[18] It is shown in Figure 34. There are two large panels at the top of the screen and two small panels at the bottom. The small panel at the bottom right shows the parameter space. The red point represents the current parameter of the sphairahedra. We can see the deformation of the quasi-sphere and sphairahedron at the same time. We can control various parameters with the side panel.

For more details about the fractals generated by the sphairahedron, read (Nakamura & Ahara, 2018; Nakamura et al., 2020). More pictures and gif animation files of the fractals are summarized on the author's webpage.[19] In Nakamura et al. (2020), we discussed sphairahedra and the fractals generated by them from a mathematical viewpoint. We experimented with the sphairahedron-based fractal renderer, and we found mathematical problems using the software.

5.3.1. Colouring of quasi-spheres

In this section, we discuss the colouring of the fractals. Colouring affects the appearances of fractals. Three-dimensional fractals have infinitely small surfaces. Thus, we cannot determine the colour of a pixel as one colour. Therefore, we take samples within a pixel, and we fill the pixel with the average colour.

Quasi-spheres have fractal surfaces. In this case, we refer to the colour wheel according to the number of inversions of the IIS. We use large steps on the colour wheel so that each tile will be coloured clearly, as shown in Figure 31(b). When we have many iterations, the tiles seem to be grey. In one pixel, there are many sampling points and tiles with various colours. Thus, the many colours are averaged, and the result is grey, as shown in Figure 31(d).

When we use the colour wheel with small steps, the sampled colours are similar, and they are averaged in the pixels. Therefore, the surfaces of the quasi-sphere are also similar colours, as shown in Figure 31(e).

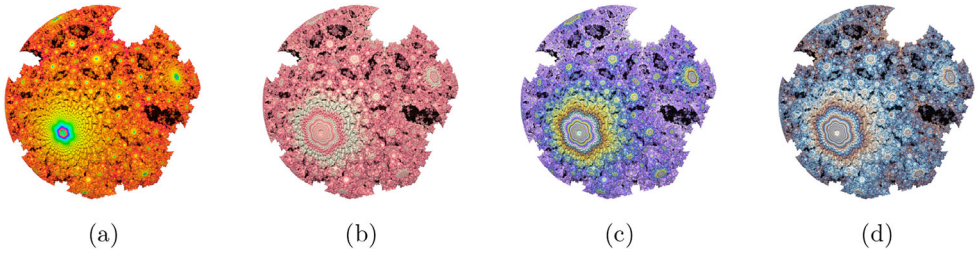


Figure 35. Quasi-spheres coloured with different colour palettes.

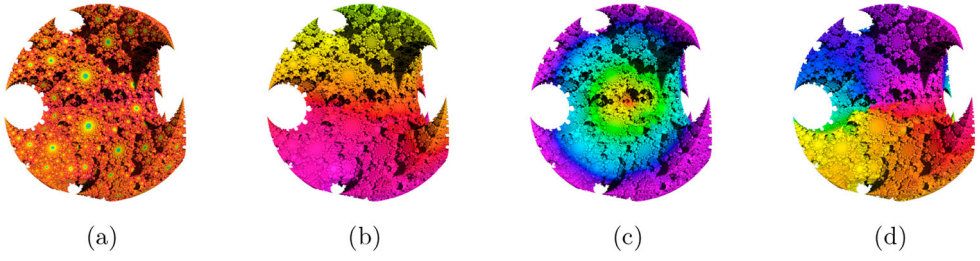


Figure 36. Quasi-spheres coloured by different parameters. (a) Number of inversions. (b) Y-coordinates. (c) Distance from origin and (d) Argument of y-coordinates and x-coordinates.

When Ahara and Araki rendered quasi-spheres, they coloured them according to the coordinates of the surfaces of the quasi-sphere.[20] In contrast, we colour the quasi-sphere according to the number of inversions applied to the first tile. This method emphasizes the surfaces that gather infinite numbers of tiles, but it has slow convergence. This phenomenon is often seen around parabolic fixed points.

We used a simple HSV colour wheel, but there is an algorithm to generate colour palettes. For more details, read the blog post by Inigo Quilez.[21] In Figure 35, we show four different colour palettes.

Colouring also depends on the types of parameters, for example, the number of iterations of inversions, the coordinates of a point, the distance between the origin and the point, and the argument of the point. These are shown in Figure 36.

In addition to these colouring methods, there are many ways to use the parameters originating from the IIS.

5.3.2. Three-dimensional printing

Three-dimensional printing is a useful way to materialize mathematical objects. Yoshiaki Araki tried to materialize quasi-spheres in 2006 (Araki, 2006). Recently, many people have worked on materializing mathematical objects using three-dimensional printers. For example, Jeremie Brunet has made many beautiful three-dimensional printed fractals.[22] Henry Segerman published a book about three-dimensional printed mathematical objects (Segerman, 2016).

We can also materialize three-dimensional fractals rendered by the IIS. The IIS generates voxel data; therefore, we have to convert voxel data into mesh data. For more details

(a) (b)

Figure 37. Monochrome printing by the MakerBot Replicator Z18 with PLA resin.

(a) (b)

Figure 38. Full-colour printing by the DMM.make 3D PRINT. (a) Plaster and (b) Plastic.

about generating data for three-dimensional printing using the IIS, see Section 4.4.8 of Nakamura (2018).

Figure 37 is made with a monochrome three-dimensional printer. We used 'MakerBot Replicator Z18' and PLA resin. We can print the model with a home-use three-dimensional printer; however, full-coloured three-dimensional printing is not popular yet. The author places orders with a three-dimensional printing service. Figure 38 shows full-coloured plaster and plastic models.

The author has published three-dimensional models at Sketchfab.[23] Note that soma_arc is the screen name of the author. These kinds of mathematical objects are interesting models for applications such as art and studying objects in doing mathematics. They allow us to observe the objects from another point of view, and we may find underlying symmetrical properties by physically touching them.

5.4. Shader artworks

The author has published many artistic works with GLSL. They are published at *Shadertoy* with the source code.[24] Shadertoy is a website used to build and share shaders.

As mentioned in the background section, the author has submitted artworks to the GLSL Graphics Compo at TDF. This category is a competition in real-time graphics rendered by the GLSL. All of the author's artworks are related to Kleinian groups and rendered using the IIS. They are looped animations. Additionally, the author has submitted PC 4K graphics

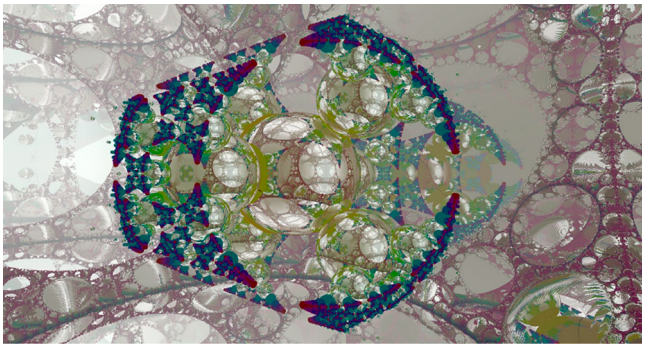


Figure 39. Indra's bubbles.

Figure 40. Pseudo-Kleinian rendered with Fragmentarium.

to the Combined Graphics Compo. This is a competition for 4 KB executable files that generate one image.

5.4.1. Indra's bubbles

'Indra's Bubbles'[25] received the GLSL Graphics Compo 2nd place at TDF 2016. Figure 39 shows a scene from the work.

The themes of the work are three-dimensional fractals and reflections. A specular object at the centre is based on the orbit of the Kleinian groups shown in Figure 14(b). The number of the orbit of the spheres increases, and finally, they become one object. The fractal at the centre is a reflective material and reflects the surrounding objects. Mirror reflections by spheres are similar to sphere inversions. The reflections of spheres make beautiful patterns.

The Kleinian group-like fractal surrounding the spheres is called a *pseudo-Kleinian*. It is a famous kind of fractal similar to the limit set of the Kleinian groups shown in Figure 40. The image is rendered with a renderer called Fragmentarium.[26] The source code is contained in Fragmentarium.

A pseudo-Kleinian uses a sphere fold operation similar to sphere inversion. Therefore, sphere folds cause Kleinian group-like shapes. For more details about sphere folds, see the blog post by Christensen.[27]

The rendering algorithm enables us to render the fractal quickly, like the IIS, but the pseudo-Kleinian is not a Kleinian group that mathematicians study. We apply a mirror

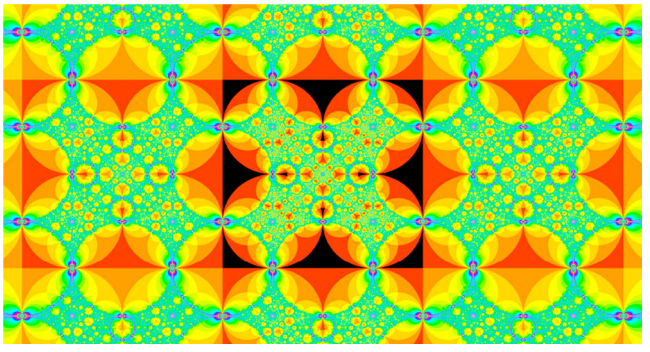


Figure 41. Schottky waltz.

material to the ceiling, floor, and balls of the pseudo-Kleinian so that all of the space seems to expand infinitely.

Usually, a mirror reflection takes much time to render, but the IIS and sphere tracing enable us to render the work in real-time.

5.4.2. Schottky waltz

'Schottky Waltz'[28] received the GLSL Graphics Compo 1st place at TDF 2017. The main theme is circular motion. The movie uses disks and their inversions on the circumference. The fractals are based on extended Kleinian groups.

The rotated and enlarged images make a large change to the graphics. Additionally, the circle inversion fractals are deformed drastically according to the inversions of circles. When the circles overlap each other, we obtain complicated and beautiful fractals; note, however, that they are not actual Kleinian groups.

Figure 41 shows a pattern generated by inversions in five circles and reflections by four half-planes. Thanks to the IIS, we can render enlarged images in real-time.

When the author made this artwork, we conducted an experiment using Schottky Link to find the parameters that generate nice-looking fractals.

5.4.3. Morph

'Morph'[29] was published at TDF 2018; see Figure 42. The themes of the work are the continuous deformation of fractals. We render the disks using Jos Leys's rendering algorithm[30] for Maskit groups, as shown in Figure 43, and we apply inversions to the disks using the IIS. Then, we obtain Figure 44.

The Maskit group is a subgroup of the Kleinian group. The union of the circumferences of the disks shown in Figure 43 is the limit set of the Maskit group. The shapes of the fractal also change dramatically according to the parameters.

Although the IIS uses circle and sphere inversions, Jos Leys used Möbius transformations algebraically. He observed the orbits of the Maskit parametrization group and discovered an algorithm to visualize Maskit groups.

Jos's method for rendering Maskit groups enables images of fractals to undergo continuous deformation. Therefore, the shape of the fractals does not collapse. Additionally, we apply the IIS and obtain flame-like patterns.

The first appearance of Figures 43 and 44 is in Nakamura (2018).

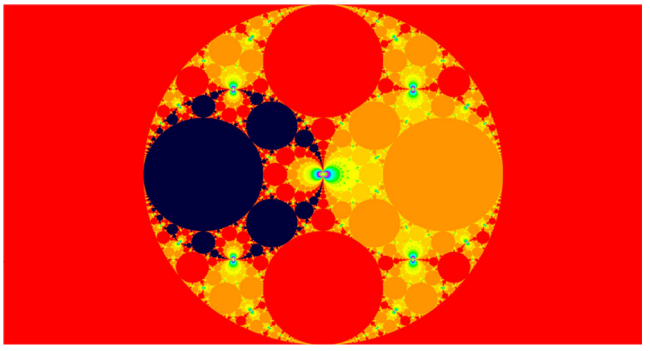


Figure 42. Morph.

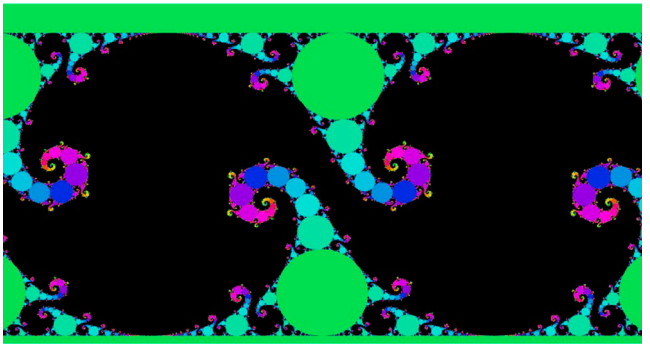


Figure 43. The limit set of the Kleinian groups with Maskit parametrization.

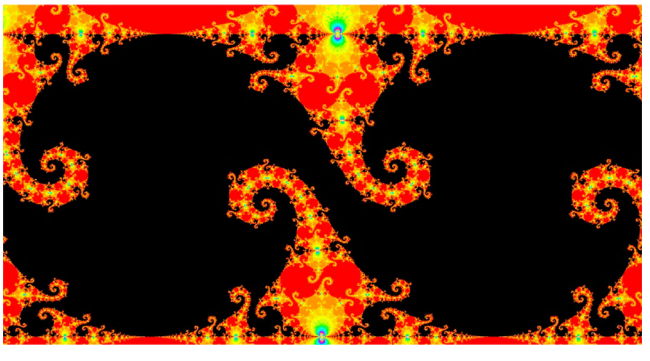


Figure 44. The limit set after applying inversions of the circles.

5.4.4. TokyoDemoFesTessellation

'TokyoDemoFesTessellation'[31] received the Combined Graphics Compo 2nd place at TDF 2017. It is shown in Figure 45 and is rendered with the GLSL and IIS.

This work is a simple tessellation pattern using T, D, and F. It is not related to Kleinian groups but uses the basic IIS idea. In other words, we prepare the original tile and use transformations to move tiles to the original tile.

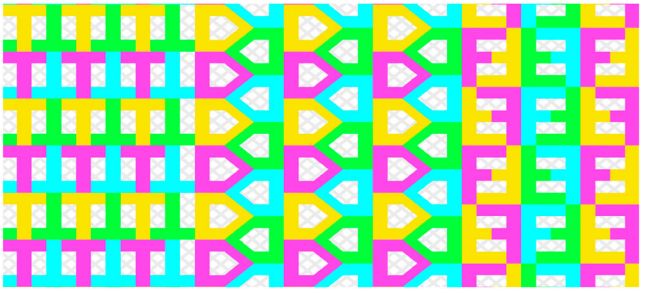


Figure 45. TokyoDemoFesTessellation.

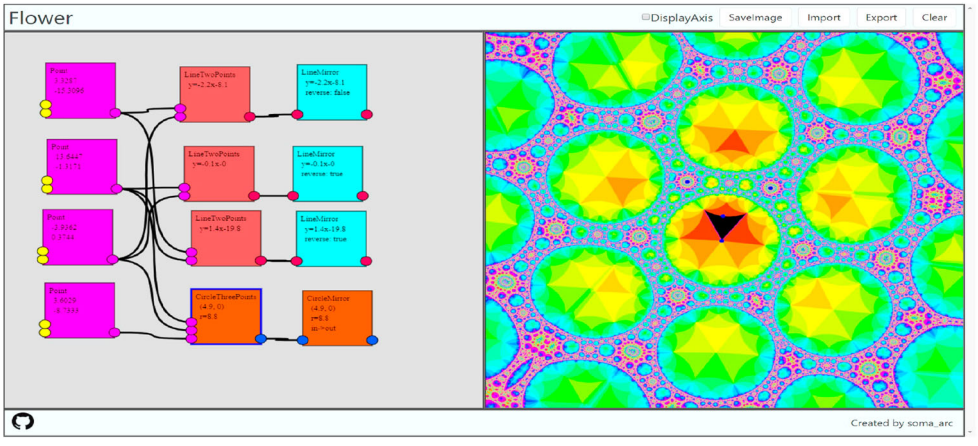


Figure 46. Flower.

5.5. Flower: a flow-based programming environment for kaleidoscope patterns

Kaleidoscope patterns are generated by mirror reflections. Therefore, the IIS enables us to render kaleidoscope patterns in real-time. The author has developed a kaleidoscope pattern generator based on flow-based programming called *Flower*.[32] Additionally, we can make geometrical constructions interactively, as in GeoGebra, with Flower.

Flow-based programming is a programming paradigm in which all of the process and data are represented by nodes. We connect the nodes with edges, and the data are sent from the output socket to the input socket. For more details, see Nakamura and Ahara (2020). Also, there is an introductory video for Flower on YouTube.[33]

There are three advantages of flow-based programming. First, it is easy to understand the construction procedure. Second, flow-based programming is flexible. We can insert processes wherever we like easily. Third, flow-based programming has high extensibility. We can create original nodes or scripts easily.

The construction is rendered with a shader and the IIS. Figure 46 shows the screen of the software. The left panel shows a node graph. The right panel shows the rendered constructions generated by the node graph, which are rendered by the IIS. It also enables us to render enlarged images of the fractals in real-time. This software can be used artistically and educationally.

6. Summary

As noted above, the IIS has various applications and can render them in real-time. The algorithm is valuable in mathematics and art. Additionally, we seek not only a computational experiment but also beautiful artistic images. Kleinian groups show rich fractal shapes.

However, there are many Kleinian groups that we cannot visualize using the IIS; for example, we cannot render the limit set of degenerate group with IIS. But, as mentioned in Section 4.3. Bachman et. al succeeded in rendering the limit sets of degenerate groups efficiently.

Our ultimate goal is to find methods, regardless of IIS, to visualize all of the Kleinian groups in real-time. In particular, four-dimensional Kleinian groups have many unsolved problems and few visualized images. For instance, Sakugawa's families of Kleinian groups, the sphairahedron, and quasi-spheres have many unsolved problems. Moreover, there are still fractals among the Kleinian groups that have not been seen. The IIS will play an important role in their visualization and in finding clues to unravel the unsolved problems of four-dimensional Kleinian groups.

Notes

1. http://www.fractalforums.com/.
2. http://tokyodemofest.jp/2018/?lang=en.
3. https://www.youtube.com/watch?v=iRkZcTg1JWU.
4. https://thebookofshaders.com/.
5. https://cindyjs.org/gallery/main/Kleinian/.
6. http://bulatov.org/math/1101/.
7. https://github.com/mtwoodard/hypVR-Ray.
8. http://www.3-dimensional.space/hyp.html.
9. https://www.shadertoy.com/view/MddfD7.
10. https://www.shadertoy.com/view/MsScWW.
11. https://www.shadertoy.com/view/XsBcDD.
12. https://www.shadertoy.com/view/lsSyDW.
13. https://www.shadertoy.com/view/lsjyzK.
14. https://www.shadertoy.com/view/ldByDW.
15. https://www.shadertoy.com/view/MdjyRV.
16. https://schottky.jp.
17. https://www.shadertoy.com/view/4sSfzD.
18. https://soma-arc.net/SphairahedronExperiment/.
19. https://sphairahedron.net.
20. https://www.youtube.com/watch?v=3lcO9zRCv-4.
21. https://www.iquilezles.org/www/articles/palettes/palettes.htm.
22. https://www.shapeways.com/shops/3dfractals.
23. https://sketchfab.com/soma_arc.
24. https://www.shadertoy.com/user/soma_arc.
25. https://www.shadertoy.com/view/XsGGWG.
26. http://syntopia.github.io/Fragmentarium/.
27. http://blog.hvidtfeldts.net/index.php/2011/11/distance-estimated-3d-fractals-vi-the-mandelbox/.
28. https://www.shadertoy.com/view/XslyzH.
29. https://www.shadertoy.com/view/MlGfDG.
30. http://www.josleys.com/article_show.php?id=221.

31. https://www.shadertoy.com/view/MsscR4.
32. https://soma-arc.net/Flower/.
33. https://youtu.be/FWp-eF5gz5o.

Acknowledgments

We would like to thank professor Kazushi Ahara, the author's academic advisor. He provided discussion from various perspectives and worthy comments. We wish to thank Dr. Yoshiaki Araki for his valuable advice on this paper. We are also grateful to thank the referees for introducing the related literature and worthy comments.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The author is supported in part by a grant of the Research Associate position by Meiji University.

ORCID

Kento Nakamura http://orcid.org/0000-0002-5628-4925

References

Ahara, K., & Araki, Y. (2003, July). Sphairahedral approach to parameterize visible three dimensional quasi-Fuchsian fractals. In *Proceedings of computer graphics international* (pp. 226–229), Los Alamitos, CA, USA. IEEE Computer Society.

Araki, Y. (2006). Materializing 3D quasi-Fuchsian fractals. *Forma*, 21(1), 19–27.

Bachman, D., Goerner, M., & Schleimer, S. (2020). *Cohomology fractals, Cannon-Thurston maps, and the geodesic flow*. arXiv:2010.05840.

von Gagern, M., & Richter-Gebert, J. (2009). Hyperbolization of euclidean ornaments. *Electronic Journal of Combinatorics*, 16(2), Article number R12. https://doi.org10.37236/78

Hart, J. C. (1996). Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10), 527–545. https://doi.org/10.1007/s003710050084

Marden, A. (2016). *Hyperbolic manifolds: An introduction in 2 and 3 dimensions*. Cambridge University Press.

Montag, A. (2014). *Interactive image sequences converging to fractals* [Bachelor thesis, Technical University of Munich].

Mumford, D., Series, C., & Wright, D. (2002). *Indra's pearls: The vision of Felix Klein*. Cambridge University Press.

Nakamura, K. (2018). *Iterated inversion system: An efficient algorithm to visualize Kleinian groups based on inversions* [Master thesis, Meiji University]. Retrieved May 23, 2020 from https://soma-arc.net/paper/masterThesis.pdf.

Nakamura, K., & Ahara, K. (2016). A new algorithm for rendering kissing Schottky groups. In *Proceedings of bridges 2016: Mathematics, music, art, architecture, education, culture* (pp. 367–370). Tessellations Publishing.

Nakamura, K., & Ahara, K. (2017). A geometrical representation and visualization of Möbius transformation groups. In *Proceedings of bridges 2017: Mathematics, music, art, architecture, education, culture* (pp. 159–166). Tessellations Publishing.

Nakamura, K., & Ahara, K. (2018). Sphairahedra and three-dimensional fractals. In *Proceedings of bridges 2018: Mathematics, music, art, architecture, education, culture* (pp. 171–178). Tessellations Publishing.

Nakamura, K., & Ahara, K. (2020). A flow-based programming environment for geometrical construction. In *Mathematical software – ICMS 2020*, *Lecture notes in computer science: Vol. 12097* (pp. 426–431). Springer.

Nakamura, K., Araki, Y., & Ahara, K. (2020). *Polyhedra with spherical faces and four-dimensional Kleinian groups*. Preprint. https://soma-arc.net/paper/expMath2020preprint.pdf

Nelson, R., & Segerman, H. (2015). *Visualizing hyperbolic honeycombs*. arXiv:1511.02851.

Sakugawa, K. (2010). On limit sets of 4-dimensional Kleinian groups with 3 generators. *Tokyo Journal of Mathematics*, 33(1), 165–182. https://doi.org/10.3836/tjm/1279719585

Segerman, H. (2016). *Visualizing mathematics with 3D printing*. JHU Press.