Recursive Symmetric Patterns in Sierpinski Circlets: A Computational and Visual Framework

Leo Borcherding

May 1, 2025

Abstract

This paper presents a novel interactive framework for visualizing and exploring recursive geometric structures based on the Sierpinski gasket principle applied to circular arrangements. We introduce the concept of a "Sierpinski circlet" - a recursive pattern that combines circular symmetry with fractal self-similarity. Our visualization system provides both 2D and 3D representations with adjustable parameters for symmetry count, iteration depth, and visual styling. We explore three primary pattern types: single circlets, overlapping dual circlets, and flower of life arrangements. The 3D extension maps these patterns onto polyhedra, creating immersive spherical visualizations. The mathematical foundations of these structures are analyzed, including their symmetry groups, recursive generation algorithms, and geometric properties.

1 Introduction

Fractal geometric patterns appear throughout nature and have been extensively studied in mathematical literature. The Sierpinski triangle (or gasket) represents one of the most well-known fractal structures, characterized by its recursive self-similarity and triangular symmetry. In this work, we extend this concept by introducing circular symmetry to create what we term a "Sierpinski circlet."

Our framework allows for dynamic exploration of these patterns with the following key features:

- Variable n-fold symmetry (adjustable from 3 to 9 vertices)
- Iterative recursive depth (1 to 7 levels)
- Multiple visualization modes (2D canvas and 3D WebGL)
- Pattern variations including single patterns, overlapping dual patterns, and flower of life arrangements
- Real-time parameter adjustment and visual customization

This paper details both the mathematical foundations of these structures and the implementation of an interactive visualization tool for exploring their properties.

2 Mathematical Framework

2.1 Basic Definitions

We define a Sierpinski circlet $\mathcal{C}_k(n)$ as a recursive geometric structure with k-fold symmetry and n levels of recursion. The base structure ($n = 0$) consists of a circle with radius r_0. For each subsequent iteration, we generate k smaller circles positioned along the vertices of a regular k-gon inscribed within the previous circle.

Formally, we can define the generation process as follows:

[Sierpinski Circlet] A Sierpinski circlet $\mathcal{C}_k(n)$ with k-fold symmetry and n levels of recursion is defined recursively as:

$$\mathcal{C}_k(0) = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = r_0^2\} \tag{1}$$

$$\mathcal{C}_k(n) = \mathcal{C}_k(n-1) \cup \bigcup_{i=0}^{k-1} \phi_i(\mathcal{C}_k(n-1)) \tag{2}$$

where ϕ_i is a transformation that scales by $\frac{1}{2}$ and translates to position $\mathbf{p}_i$.

2.2 Coordinate Transformations

For a circle of radius r centered at origin $(0, 0)$, the positions of the k smaller circles at the next iteration are given by:

$$\mathbf{p}_i = \frac{r}{2} \begin{pmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{pmatrix} \tag{3}$$

where $\theta_i = \frac{2\pi i}{k} - \frac{\pi}{2}$ for $i \in \{0, 1, ..., k-1\}$.

The radius of each smaller circle is:

$$r_{n+1} = \frac{r_n}{2} \tag{4}$$

This creates a recursive pattern where each new level contains k times as many circles as the previous level, with radii decreasing by a factor of 2.

2.3 Triangular Elements

In addition to circles, our visualization incorporates triangular elements that follow the Sierpinski triangle pattern. At each level n, we create a regular k-gon inscribed in each circle. These polygons form the basis for the recursive pattern, with smaller instances placed at each vertex.

The vertices of the k-gon inscribed in a circle of radius r are given by:

$$\mathbf{v}_i = r \begin{pmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{pmatrix} \tag{5}$$

where $\theta_i = \frac{2\pi i}{k} - \frac{\pi}{2}$ for $i \in \{0, 1, ..., k-1\}$.

2.4 Overlay Patterns

For the dual circlet visualization, we create two identical Sierpinski circlets with a phase shift between them. If $\mathcal{C}_k(n)$ represents the base circlet, then the overlaid circlet $\mathcal{C}'_k(n)$ is defined by applying a rotation of π radians (180 degrees):

$$\mathcal{C}'_k(n) = \mathbf{R}_\pi(\mathcal{C}_k(n)) \tag{6}$$

where $\mathbf{R}_\pi$ is the rotation matrix:

$$\mathbf{R}_\pi = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7}$$

2.5 Flower of Life Pattern

The Flower of Life pattern extends the single circlet by positioning identical circlets in a hexagonal arrangement. Given a base circlet $\mathcal{C}_k(n)$ with radius r, we position 7 instances (one central and six surrounding) at coordinates:

$$\mathbf{p}_0 = (0, 0) \quad \text{(central position)} \tag{8}$$

$$\mathbf{p}_i = r\sqrt{3} \begin{pmatrix} \cos(\frac{2\pi(i-1)}{6}) \\ \sin(\frac{2\pi(i-1)}{6}) \end{pmatrix} \quad \text{for } i \in \{1, 2, ..., 6\} \tag{9}$$

The distance $r\sqrt{3}$ ensures proper overlap between adjacent circlets.

3 3D Extension

3.1 Basic 3D Circlet

To extend the 2D circlet to 3D, we project the structure onto the xy-plane and render it using appropriate 3D primitives. Circles are represented as torus geometries with a small cross-sectional radius, and polygons are rendered as flat mesh objects.

The generation process follows the same recursive logic as the 2D case, but with 3D coordinates:

$$\mathbf{p}_i = \frac{r}{2} \begin{pmatrix} \cos(\theta_i) \\ \sin(\theta_i) \\ 0 \end{pmatrix} \tag{10}$$

3.2 3D Flower of Life

The 3D Flower of Life extends the 2D arrangement into three dimensions, using the same hexagonal positioning but with 3D transformation matrices:

$$\mathbf{p}_i = r\sqrt{3} \begin{pmatrix} \cos(\frac{2\pi(i-1)}{6}) \\ \sin(\frac{2\pi(i-1)}{6}) \\ 0 \end{pmatrix} \quad \text{for } i \in \{1, 2, ..., 6\} \tag{11}$$

3.3 Skybox Mapping

The skybox implementation maps circlet patterns onto the faces of a regular deltahedron. Our implementation supports three polyhedra:

1. Tetrahedron (4 triangular faces)
2. Octahedron (8 triangular faces)
3. Icosahedron (20 triangular faces)

For each face, we position a circlet pattern centered on the face and oriented along the face normal vector. Given a face with vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, we calculate:

$$\mathbf{c} = \frac{\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3}{3} \quad \text{(face center)} \tag{12}$$

$$\mathbf{n} = \frac{(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)}{|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)|} \quad \text{(face normal)} \tag{13}$$

The pattern is then positioned at the face center and oriented to align with the face's coordinate system.

4 Visualization Implementation

4.1 2D Canvas Rendering

For 2D visualizations, we use the HTML5 Canvas API with the following algorithm:

Algorithm 1 DrawSierpinskiCircle(x, y, $radius$, $iterations$, $symmetry$)

if $iterations \leq 0$ **then**
  return
end if
Draw circle at (x, y) with radius $radius$
Draw polygon with $symmetry$ sides
if $iterations > 1$ **then**
  $newRadius \leftarrow radius/2$
  for $i \leftarrow 0$ to $symmetry - 1$ **do**
    $angle \leftarrow 2\pi i/symmetry - \pi/2$
    $newX \leftarrow x + (radius/2) \times \cos(angle)$
    $newY \leftarrow y + (radius/2) \times \sin(angle)$
    DrawSierpinskiCircle($newX$, $newY$, $newRadius$, $iterations - 1$, $symmetry$)
  end for
end if

4.2 3D WebGL Rendering

For 3D visualizations, we utilize the Three.js library to create and manipulate 3D objects. The implementation follows the same recursive logic but constructs 3D geometries:

Algorithm 2 CreateSierpinskiCircle3D(*radius*, *iterations*, *symmetry*)

Create empty 3D group object *pattern*
Create torus geometry with radius *radius*
Add torus to *pattern*
if *showTriangles* AND *iterations* > 0 **then**
  Create polygon mesh with *symmetry* sides
  Add polygon to *pattern*
  if *iterations* > 1 **then**
    $childRadius \leftarrow radius/2$
    for $i \leftarrow 0$ to $symmetry - 1$ **do**
      $angle \leftarrow 2\pi i/symmetry - \pi/2$
      $x \leftarrow (radius/2) \times \cos(angle)$
      $y \leftarrow (radius/2) \times \sin(angle)$
      $child \leftarrow$ CreateSierpinskiCircle3D($childRadius$, $iterations - 1$, $symmetry$)
      Position *child* at $(x, y, 0)$
      Add *child* to *pattern*
    end for
  end if
end if
return *pattern*

4.3 User Interface and Interaction

The visualization system includes a comprehensive user interface allowing real-time adjustment of the following parameters:

- Visualization mode (2D/3D)
- Pattern type (single circlet, double circlet, flower of life)
- Symmetry count (3-9)
- Iteration depth (1-7)
- Line width and opacity
- Color selection for circles, triangles, and overlays
- Rotation speed and axis selection (for 3D)
- Zoom and pan functionality
- Export options (PNG, GIF, MP4)

The interface provides instant visual feedback, allowing users to explore the parameter space and discover interesting pattern configurations.

5 Geometric Analysis

5.1 Symmetry Properties

The Sierpinski circlet exhibits multiple levels of symmetry:

- Rotational symmetry of order k (where k is the symmetry parameter)
- Reflection symmetry across k axes
- Scale symmetry between recursive levels

The combined symmetry group for a circlet with k-fold symmetry is isomorphic to the dihedral group D_k.

5.2 Fractal Dimension

The Sierpinski circlet exhibits fractal properties similar to the Sierpinski triangle. The fractal dimension can be calculated using the box-counting method:

$$D = \frac{\log(k)}{\log(2)} \tag{14}$$

where k is the symmetry parameter. For the standard case where $k = 3$, this gives $D = \frac{\log(3)}{\log(2)} \approx 1.585$, matching the Sierpinski triangle's dimension.

5.3 Pattern Interaction in Overlays

When two circlet patterns are overlaid with a phase difference, they create interference-like visual effects. These interactions follow from the superposition principle, where the visual impact of overlapping regions creates interesting moiré patterns.

The combined opacity α in overlapping regions follows:

$$\alpha_{total} = 1 - (1 - \alpha_1)(1 - \alpha_2) \tag{15}$$

where α_1 and α_2 are the individual opacity values.

6 Implementation Details

6.1 Performance Considerations

The recursive nature of the Sierpinski circlet leads to exponential growth in the number of elements as iteration depth increases. For a circlet with k-fold symmetry and n levels of recursion, the total number of circles rendered is:

$$N_{circles} = \sum_{i=0}^{n} k^i = \frac{k^{n+1} - 1}{k - 1} \tag{16}$$

To maintain interactive performance, we implement several optimizations:

- Limiting maximum iteration depth to 7

- Using hardware-accelerated WebGL rendering for 3D modes
- Implementing visibility culling for off-screen elements
- Adjusting detail level based on zoom factor

6.2 Color Management and Styling

The visualization supports custom color selection for different elements:

- Circle color (primary outlines)
- Triangle color (polygon fills)
- Overlay color (for dual circlet mode)

Additionally, we implement both light and dark themes for the interface, with appropriate contrast adjustments for all elements.

6.3 Animation and Export

The system provides several options for animated visualizations:

- Real-time rotation along selectable axes (X, Y, Z)
- Adjustable rotation speed
- Export functionality for static images (PNG)
- Animation export as GIF or MP4 with customizable duration

The animation capture process uses the CCapture.js library for high-quality output at 60 frames per second.

7 Results and Discussion

7.1 Visual Patterns and Emergent Properties

The Sierpinski circlet visualization reveals several interesting properties:

- At higher iteration levels, the pattern approaches the classical Sierpinski gasket in the limit
- Varying the symmetry parameter creates diverse visual structures
- The overlay of two phase-shifted patterns generates complex interference patterns
- The Flower of Life arrangement demonstrates how individual circlets can combine to form larger coherent structures

7.2 3D Extensions

The 3D implementations provide additional perspectives on these structures:

- The basic 3D circlet allows exploration of the structure from different viewing angles
- The 3D Flower of Life reveals how multiple circlets interact in three-dimensional space
- The skybox mapping onto polyhedra creates immersive environments where the patterns surround the viewer

The icosahedral mapping is particularly interesting, as it represents the highest level of symmetry available in regular deltahedra.

7.3 Potential Applications

The visualization framework and the mathematical structures it explores have potential applications in:

- Educational tools for teaching fractal geometry and symmetry
- Generative art and computational aesthetics
- Design inspiration for architecture and product design
- Visualization of mathematical concepts related to group theory and topology

8 Conclusion

This paper has introduced the Sierpinski circlet as a novel geometric structure combining circular symmetry with fractal recursion. We have presented a comprehensive visualization framework for exploring these patterns in both 2D and 3D contexts, with customizable parameters and interactive controls.

The mathematical analysis provides a formal foundation for understanding these structures, while the implementation details offer practical insights for creating similar visualizations. The results demonstrate the rich visual complexity that emerges from relatively simple recursive rules, highlighting the beauty of mathematical patterns.

8.1 Future Work

Several directions for future exploration include:

- Extension to higher-dimensional spaces (4D and beyond)
- Exploration of alternative recursion rules and non-uniform scaling factors
- Investigation of dynamic systems where pattern parameters evolve over time
- Application to physical systems such as acoustic resonance or electromagnetic field visualization
- Integration with virtual reality environments for immersive exploration

References

[1] Sierpiński, W. (1915). "Sur une courbe dont tout point est un point de ramification". Comptes Rendus de l'Academie des Sciences, 160, 302–305.

[2] Mandelbrot, B. B. (1982). "The Fractal Geometry of Nature". W. H. Freeman and Company.

[3] Cabello, R. et al. (2010-2025). "Three.js". `https://threejs.org/`.

[4] Bello, J. (2016). "CCapture.js". `https://github.com/spite/ccapture.js/`.

[5] Melchizedek, D. (1999). "The Ancient Secret of the Flower of Life". Light Technology Publishing.

[6] Falconer, K. (2003). "Fractal Geometry: Mathematical Foundations and Applications". John Wiley Sons.

[7] Armstrong, M. A. (1988). "Groups and Symmetry". Springer-Verlag.

[8] World Wide Web Consortium (W3C). (2014). "HTML Canvas 2D Context". `https://www.w3.org/TR/2dcontext/`.

[9] Khronos Group. (2011). "WebGL Specification". `https://www.khronos.org/webgl/`.