**Program One:** **The Sierpinski Gasket**

Jordon Paynter

Department of Computer Science, Colorado State University Global

CSC 405: Graphics and Visualization

Professor Jennifer Marquez

November 26, 2023

Contents

[**Reflection** 3](#_Toc152166152)

[**Introduction** 3](#_Toc152166153)

[**Understanding WebGL Primitives and Attributes** 3](#_Toc152166154)

[**Modifying Primitives and Attributes for Visual Variations** 4](#_Toc152166155)

[**Exploring Fractal Geometry through WebGL** 4](#_Toc152166156)

[**Conclusion** 4](#_Toc152166157)

[**Images of the Sierpinski Gasket** 6](#_Toc152166158)

[Initial **Sierpinski Gasket** 6](#_Toc152166159)

[Color alteration **Sierpinski Gasket** 7](#_Toc152166160)

[Starting point alteration **Sierpinski Gasket** 8](#_Toc152166161)

[Different recursive depth of **Sierpinski Gasket** 9](#_Toc152166162)

[Different vertices and color of **Sierpinski Gasket** 10](#_Toc152166163)

[Different depth of **Sierpinski Gasket** from the view perspective 11](#_Toc152166164)

[Different color shader of **Sierpinski Gasket** 12](#_Toc152166165)

[Different opacity level of shader for the **Sierpinski Gasket** 13](#_Toc152166166)

[**REFERENCES** 14](#_Toc152166167)

## **Reflection**

### **Introduction**

The Sierpinski Gasket, a fractal known for its intricate and repetitive pattern, offers a fascinating subject for computer graphics exploration, particularly within WebGL. This paper delves into the specifics of rendering this fractal using WebGL, emphasizing the roles of primitives and attributes. By analyzing the provided code, we aim to uncover how these components contribute to the fractal's visualization and explore the effects of their manipulation.

### **Understanding WebGL Primitives and Attributes**

In the context of WebGL, primitives and attributes form the core components of any rendering process. The primitives determine the basic shapes that WebGL will render, while attributes provide detailed data about these shapes, such as their coordinates or colors. In the provided code, the primary primitive used is **gl.TRIANGLES**, as highlighted by the **gl.drawArrays(gl.TRIANGLES, 0, 3)** call. This choice perfectly aligns with the Sierpinski Gasket's structure, which comprises equilateral triangles.(Shreiner et al., 2020) The **gl.TRIANGLES** primitive instructs WebGL to interpret sets of three vertices as individual triangles, effectively forming the fractal's fundamental geometric shape.

Attributes in the code play a crucial role in defining the vertices of these triangles. The **coordinates** attribute, declared in the vertex shader as **attribute vec2 coordinates;**, is essential in determining the position of each vertex. This attribute is linked to the vertex data via the **gl.vertexAttribPointer(coordinates, 2, gl.FLOAT, false, 0, 0)** function. (tutorialspoint, n.d.) The recursive function **renderSierpinski** is responsible for calculating the coordinates for smaller triangles at each recursion level, which is crucial for producing the fractal's detailed pattern.

### **Modifying Primitives and Attributes for Visual Variations**

Experimenting with primitives and attributes can significantly alter the fractal's appearance. It is altering the primitive from **gl.TRIANGLES** to either **gl.LINES** or **gl.POINTS** would change how the fractal is visually represented, showcasing its edges or vertices. This would offer a new perspective on the fractal's structure and accentuate different aspects of its geometry, such as its linear connections or nodal points. (Romualdo, 2020)

Manipulating attributes, particularly by introducing new ones like color, can yield diverse visual effects. Applying a color attribute in the vertex shader and modifying it in the fragment shader would allow different fractal parts to be rendered in various colors. It could be achieved by defining a new color attribute in the vertex shader, which would then be passed to the fragment shader. Such a modification would produce a multicolored fractal, enhancing its visual appeal and potentially offering new insights into its recursive construction. (WebGL fundamentals, n.d.)

### **Exploring Fractal Geometry through WebGL**

The process of rendering the Sierpinski Gasket in WebGL is a valuable learning tool for understanding fractal geometry and computer graphics. By manipulating primitives and attributes, one can better appreciate the fractal's structure and underlying mathematical principles. This exploration goes beyond mere visual representation, offering an educational experience in the complexities of fractal patterns and the capabilities of WebGL.

### **Conclusion**

The rendering of the Sierpinski Gasket using WebGL highlights the powerful capabilities of this technology in visualizing complex mathematical patterns. Through carefully manipulating primitives and attributes, various fractal aspects can be explored and presented in new and innovative ways. This serves as an educational resource for understanding fractal geometry and opens up creative avenues for graphical representation in computer graphics. The flexibility and robustness of WebGL make it an invaluable tool for visualizing mathematical concepts and creating intricate graphical designs, showcasing the intersection of mathematics, art, and technology.

# **Images of the Sierpinski Gasket**

**Figure 1a.**

Initial **Sierpinski Gasket**

A screenshot of a computer screen

Description automatically generated

**Figure 1b.**

Color alteration **Sierpinski Gasket**

A screenshot of a computer screen

Description automatically generated

**Figure 1c.**

Starting point alteration **Sierpinski Gasket**

A screenshot of a computer screen

Description automatically generated

**Figure 1d.**

Different recursive depth of **Sierpinski Gasket**

A screenshot of a computer screen

Description automatically generated

**Figure 1e.**

## Different vertices and color of **Sierpinski Gasket**

A screenshot of a computer screen

Description automatically generated

**Figure 1f.**

## Different depth of **Sierpinski Gasket** from the view perspective

A screenshot of a computer

Description automatically generated

**Figure 1g.**

## Different color shader of **Sierpinski Gasket**

A screenshot of a computer

Description automatically generated

**Figure 1h.**

## Different opacity level of shader for the **Sierpinski Gasket**

**A screenshot of a computer

Description automatically generated**

## **REFERENCES**

Romualdo, G. (2020, November 22). *How to generate the Sierpinski Triangle in vanilla javascript with HTML5 canvas*. DEV Community. https://dev.to/xtrp/how-to-generate-the-sierpinski-triangle-in-vanilla-javascript-with-html5-canvas-239d

Shreiner, D. (2020). Chapter 2.4 Primitives and Attributes. In *Interactive Computer Graphics* (8th ed.). essay, Pearson.

tutorialspoint. (n.d.). *WebGL - drawing a Triangle*. Online Tutorials, Courses, and eBooks Library. https://www.tutorialspoint.com/webgl/webgl\_drawing\_a\_triangle.htm

webglfundamentals.org. (n.d.). *WebGL fundamentals*. WebGL Fundamentals. https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html