**M2 CTA Reflection**

Giselle Martinez-Sanchez

Colorado State University Global

Course Code: CSC405-1

Bingdong Li

05/28/2025

**Reflection**

Building the Sierpinski Gasket in WebGL was an unexpectedly deep and layered experience, both technically and conceptually. At first glance, recursive triangle subdivision seems like a straightforward graphics problem. But in practice, breaking it down into manageable steps with clean rendering logic took several iterations, especially when shifting from a traditional recursive approach to a more visual, level-by-level rendering model.

One of the biggest technical shifts was thinking in terms of a draw queue instead of recursively pushing triangles into a buffer all at once. This queue-based approach was essential not only for debugging but also for visually tracing how the fractal evolves with each depth level. By queuing gl.drawArrays calls, I was able to animate the construction of each triangle, giving the process a narrative quality that mirrors how recursion actually builds the gasket.

Understanding how to consistently sort triangle points by coordinate (notice how each triangle is rendered in the order of top, bottom left, and bottom right) was another key step. Although not strictly required by the assignment or diagram logic, it felt visually appropriate to render the sub-triangles in a specific order. Without this consistency, subdivisions could break the intended geometry or appear disjointed. To me, it wasn’t just about rendering lines but about preserving the mathematical structure and process while drawing in real time.

As a JavaScript developer with no prior WebGL experience, even small missteps in buffer updates or viewport scaling often resulted in invisible or distorted output. Getting the canvas to scale correctly with device pixel ratios, using normalized coordinates, and clearing the buffer between frames were all subtle but critical pieces of the puzzle. Learning the syntax for WebGL operations and understanding its state machine model was a key part of the learning curve.

What began as a visual experiment quickly became a masterclass in debugging invisible graphics, thinking iteratively, and building rendering logic that is both correct and maintainable. The process demanded not just technical precision, but also design sense and a deep respect for how small details compound in graphics programming.