Noah Pettinato
pettinan@oregonstate.edu
Final Project: Interactive Fractal Shader

In this final project, I developed an interactive real-time GLSL shader demo that generates a Mandelbrot-like fractal pattern. The project utilizes the GLMAN framework with accompanying .glib, .vert, and .frag files to render intricate visuals that can be manipulated through user-adjustable parameters.

The implementation involves several uniform variables that are crucial in controlling the fractal's appearance:

- **uZoom:** Controls the zoom factor, allowing users to explore different levels of fractal detail by scaling the (s,t) coordinates.
- **uIntensity:** Adjusts the overall brightness of the fractal, affecting its visual intensity.
- **uRedMix, uGreenMix, uBlueMix:** Determine the contribution of red, green, and blue in the color blending, enabling dynamic color customization.
- **uIterationFactor:** Modulates the maximum number of iterations used in the fractal calculation, balancing detail and performance.
- **uCenterX, uCenterY:** Define the center of the view in the complex plane, enabling users to pan across the fractal and explore different regions interactively.

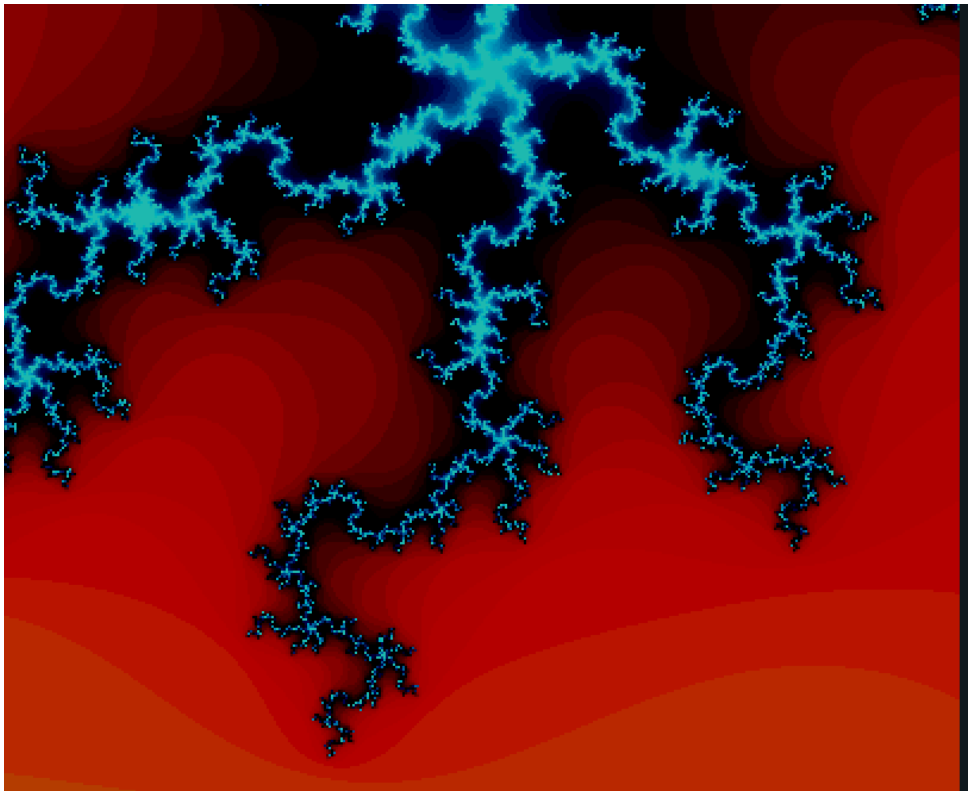
The fractal is computed in the fragment shader using an iterative approach. For each fragment, the shader determines whether the corresponding point in the complex plane belongs to the Mandelbrot set by iterating the function $z_{n+1} = z_n^2$+c. The escape time algorithm is employed, where the number of iterations before the magnitude of z exceeds 2 is used to determine the color of the fragment.

The color of each fragment is mapped based on the normalized iteration count, using a smooth gradient created with sine and cosine functions applied to the iteration value. This base color is then blended with the user-defined red, green, and blue mixes (controlled by uRedMix, uGreenMix, and uBlueMix) to achieve the final color, providing a vibrant and customizable visual output.

The use of uniform variables enables real-time interaction. Users can adjust the zoom level to delve into the fractal's intricate details, modify the intensity for visual impact, tweak the color mixes for artistic expression, and control the iteration depth to balance between detail and computational efficiency.

The project was implemented using GLMAN's .glib file to set up the rendering context and define the uniforms, a simple vertex shader to pass through the (s,t) coordinates, and a fragment shader that performs the fractal computations and color mappings. The combination of these elements results in a visually stunning and interactive fractal display that showcases the power of shader programming for real-time graphics.

Hey everyone, welcome to my interactive Mandelbrot fractal shader demo! This project lets you explore the Mandelbrot set in real-time, zooming into intricate details, panning across the fractal, tweaking colors and brightness, and adjusting iteration depth—all powered by GLSL shaders in the GLMAN framework.



Kaltura Capture Video Link: https://media.oregonstate.edu/media/t/1_iy8o61fe