**Title:**

**High-Performance Julia Set Visualization using OpenMP and Python GUI**

**Introduction:**

The Julia Set is a fractal pattern generated from complex numbers and is widely studied in mathematics and computer graphics. However, rendering detailed Julia Sets at high resolution can be computationally expensive.

To solve this, our project combines the power of:

* **OpenMP (C)** for fast, parallel computation of Julia sets.
* **Python** for building a user-friendly GUI that allows real-time control and image saving.

**Results and Improvements After Parallelization:**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Before (Sequential)** | **After (OpenMP Parallel)** |
| Execution Time | High | Significantly Reduced (4x–10x faster) |
| GUI Responsiveness | Poor | Smooth and Responsive |
| Image Quality | Low (had to use fewer iterations) | High-resolution, detailed |

**Methodology:**

1. **C Code** is written to compute Julia set using OpenMP to parallelize pixel-wise computations.
2. This compiled code is used in **Python via ctypes**.
3. **Python GUI (Tkinter)** accepts user input for real & imaginary values.
4. Computed image is rendered and saved as PNG using **Pillow (PIL)**.
5. The GUI supports input variations to generate **tree-like or mountain-like** Julia patterns.

**Algorithm (Simplified Steps):**

For each pixel (x, y) on the screen:

Convert (x, y) to complex plane coordinate (zx, zy)

iteration = 0

While (zx^2 + zy^2 < 4) and (iteration < MAX\_ITER):

Temp = zx^2 - zy^2 + real\_part

zy = 2\*zx\*zy + imag\_part

zx = Temp

iteration++

Map iteration to color

Parallelism: The loop over each pixel is done in **parallel** using #pragma omp parallel for.

**Example Inputs**

Try these to generate cool fractals:

|  |  |  |
| --- | --- | --- |
| **Real** | **Imaginary** | **Description** |
| -0.7 | 0.27015 | Classic swirling fractal |
| 0.355 | 0.355 | Symmetric fractal |
| -0.4 | 0.6 | Spidery web-like design |

Group Members Contribution:

* Slides (25% each)
* Backend (25% each)
* Frontend (25% each)
* Documentation (25% each)