Brandon Colbert, Jacob Strickland

Professor Shuangshuang Jin

CPSC 4780

6 December 2020

Marched Fractals

Our project combines two algorithms, Fractal Flames and Marching Cubes, to create meshes of fractals generated from parameters. While typical fractals have geometry emphasizing repeated patterns, Fractal Flames uses a “chaotic attractor” to create geometry with common features (Draves).

In 2 dimensions, Fractal Flames is able generate a flame-like, organic image by randomly selecting from a list of parameterized functions. We apply this algorithm in 3 dimensions to create a point cloud with color information. Once the colored point cloud has been created, we move to the visualization process, which requires the creation of a mesh. Marching Cubes is commonly used to acquire a polygonal mesh from a scalar field. In our project, this scalar field is a 3d grid containing points which are either on or off. The randomly distributed points from the Fractal Flames algorithm are then matched to the nearest grid point. This process activates the matched grid points into an on state and inscribes color alongside it.

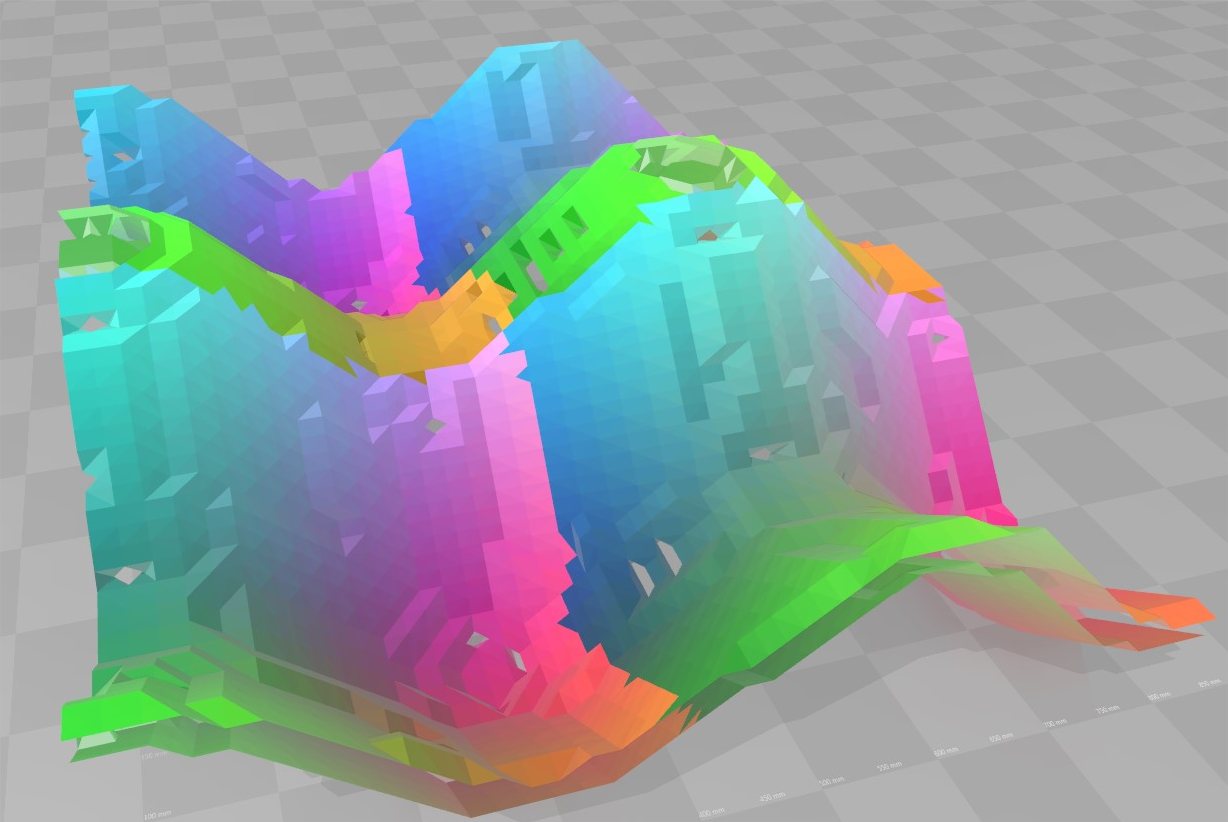
The Fractal Flames implementation creates a user-defined number of threads, which each run for a user-defined number of iterations. Each one of these threads starts with a random point and calculates a resultant point with a randomly selected function (albeit the randomness is dependent upon user-defined weights) at the start, storing that point and its associated color into the two outbuffers. On each subsequent iteration, that resultant point is passed back into another randomly selected function, once again storing the new position and associated color into the outbuffers. This continues until all threads have completed their iterations.

Since I adapted the fractal flame algorithm I used in this project from a 2D implementation, it was a bit difficult to define the colors in the same way, resulting in a less than ideal color calculation algorithm. The original implementation took advantage of pixels being hit multiple times, but since this is a 3D application, I didn’t really have the luxury of doing that..

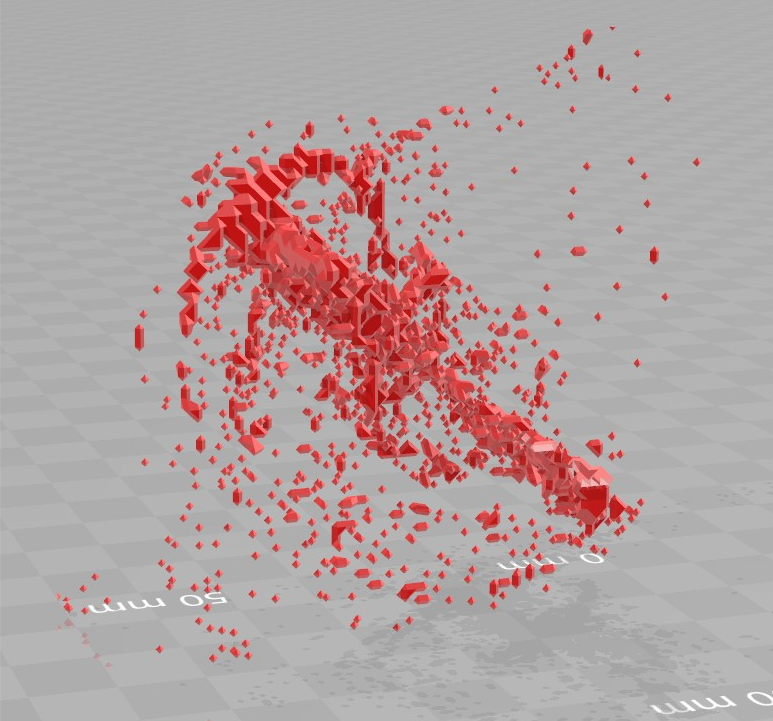
The Marching Cubes implementation converts unorganized vertices into a grid and grid points into cubes, or cells. Faces are calculated for each of these cells in parallel. These faces represent the surface of the defined shape. A list of triangles and edges are used to compute the individual faces inside a cell by evaluating the on/off state of each cell corner vertex. The final product is a mesh with vertices, triangles dictating the connection between vertices, and unique colors per face.

On my machine, creating the second object under samples in README, a visualization of “sin(x) + cos(z)”, took on average 0.02 seconds to complete with CPU and 0.07 seconds to complete with OpenCL. In order to parallelize the Marching Cubes algorithm, the information must for a cell must be transferred to the device. In my implementation, the cell information was transferred on the host and immediately run on the device to get the faces. A more optimal way to accomplish this would be passing the entire grid to the device as the global problem size and having the size of a cell as the local problem size. Due to indexing errors, however, achieving this optimal method was highly challenging. With the optimal implementation, OpenCL compute time would be much faster than CPU.

The following images are were created by marching:



sin(x) + cos(z)



Fractal Flames OpenCL

Works Cited

Bourke, Paul. “Polygonising a Scalar Field.” *Polygonising a Scalar Field (Marching Cubes)*, May 1994, paulbourke.net/geometry/polygonise/.

Coordinate Systems, Department of Mathematics, Oregon State University, sites.science.oregonstate.edu/math/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/coord/coord.html.

Draves, Scott, and Erik Reckase. “The Fractal Flame Algorithm.” 2003, https://flam3.com/flame\_draves.pdf.

Lague, Sebastian. “Coding Adventure: Marching Cubes.” *YouTube*, YouTube, 6 May 2019, [www.youtube.com/watch?v=M3iI2l0ltbE](http://www.youtube.com/watch?v=M3iI2l0ltbE).

*Parametric Surfaces*, Department of Mathematics, Oregon State University, sites.science.oregonstate.edu/math/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/parsurf/parsurf.html.

Shu, Yujie. “3D Fractal Flame Wisps.” *Clemson University*, Clemson University, 2013, pp. 1–51.