Interactive Coral Generator

Huyson Lam
University of California, Berkeley
cs285: Procedural Design, Solid Modeling, and Rapid Prototyping

Project Description

The goal of this project was to procedurally generate a water tight coral structure which could be easily prototyped. The particular type of coral which was to be modeled was those with a branching, or dendritic, structure. The most important elements to the dendritic structure are the branching structure as well as the irregular pathing of each of the branches.

Figure 1. Two pictures of coral with the desired dendritic shape

This project can be divided up into two main parts. The first part is creating the dendritic shape. This is the creation of the underlying shape of the coral. The second part is converting this underling shape to geometry. The geometry is necessary for both being able to see the coral as well as exporting it to an STL file suitable for fabrication.

Creating the Dendrite

The algorithm I used for this project was presented in "Procedural Natural Phenomena from Least-Cost Paths in a Weighted Graphs" by Ling Xu and David Mould. The algorithm is as follows:

1. Create a regular lattice with weighted Edges
2. Choose a root or roots in order from which the dendrite grows
3. Populate all nodes with their least-cost paths from the root (or roots)
4. Choose the endpoints of the dendrite
5. Find the least-cost path from each endpoint to the closest root

The regular lattice was represented as a Graph using adjacency lists. The edges of the lattice were weighted using the function 1 + rand[0,1] *10. This allowed for the irregular path each

individual branch had. Changing the weight of the random (the 10) allowed for control of how erratic the paths would look as seen in Figure 2.

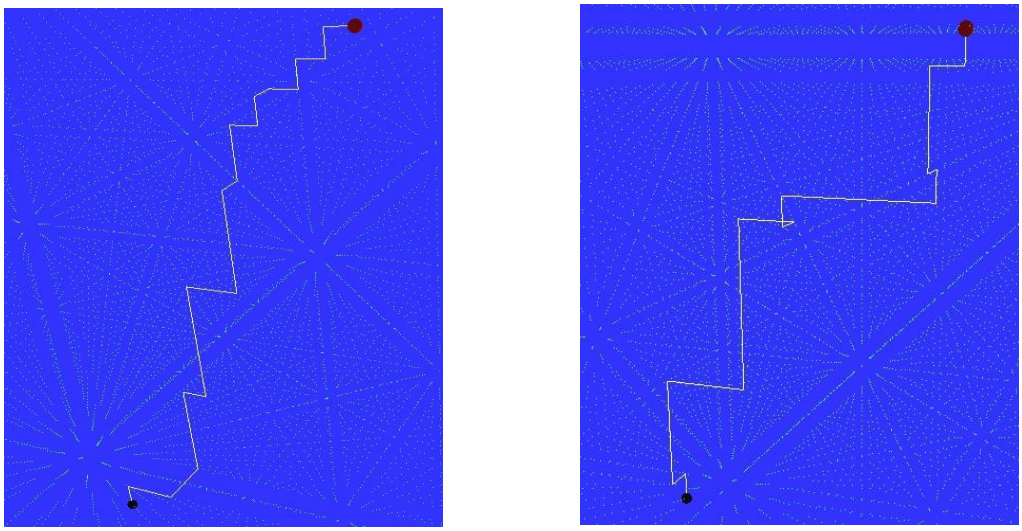

Figure 2: Both dendrites were created with one root and 1 endpoint Left: A dendrite with 10 weight on random. Right: A dendrite with weight 2 on random

Upon selecting a root, every node is populated with its least cost path to the root (or roots). This is done using breadth first search on each of the nodes. The roots are shown as black spheres. The next step in the algorithm is to select the endpoints. The endpoints are shown as white spheres. With both the roots and endpoints chosen, a greedy algorithm is used to find the last-cost path from each endpoint to the nearest root.

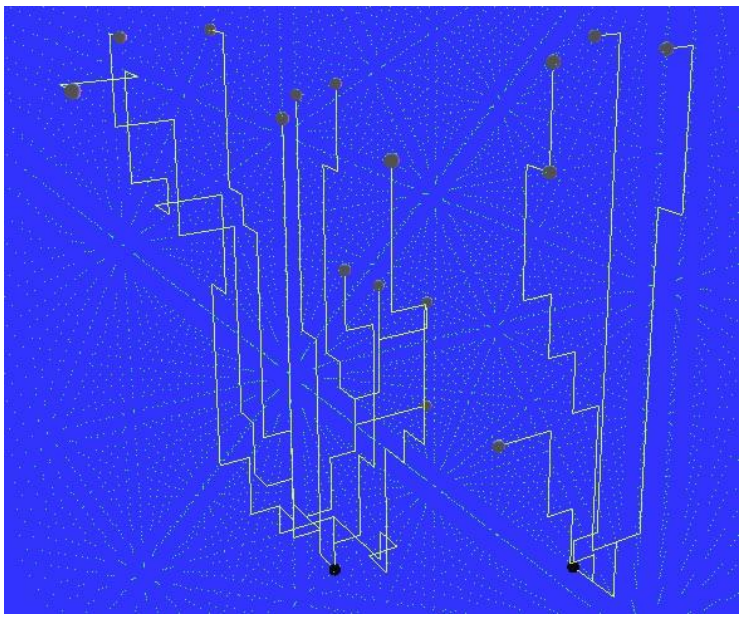

Figure 3: Dendrite with 2 roots and 16 roots

Creating the Geometry for Fabrication

The following steps were used to create the geometry of the coral:

1. Solve for the isocontours around the dendrite
2. Use marching cubes to create the geometry from the isocontour
3. Convert the information from marching cubes to an STL file

Solving for the isocontours is necessary for the marching cubes algorithm. It also allows for variable thickness of the dendrite.

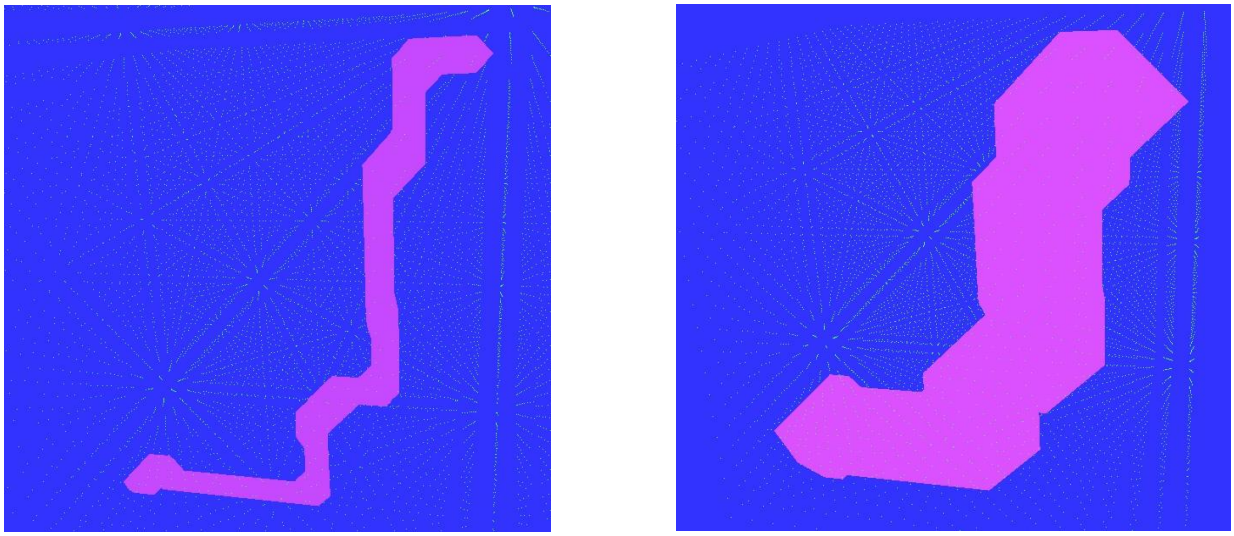

Figure 4: Left: Coral with thickness 1. Right: the same coral with thickness 3

Marching cubes is an algorithm published by Lorenson and Cline in 1987. It is used for extracting a polygonal mesh from an isosurface. For each cube, represented as eight of our nodes forming the corners of the cube, the appropriate triangles on that cube are chosen. Each of the triangles on that cube is then combined to create the geometry which you see above. With that information it is fairly straightforward to create an STL file.

A Problem

The main issue I encountered with this approach was that the coral, which now had thickness, had to be contained in the original lattice. Any edges or nodes near the edge would not create geometry and would simply be cut off. In order to solve this issue, I enlarged the lattice with a "border" which roots and endpoints would not be allowed. This space was reserved strictly for giving the coral geometry.

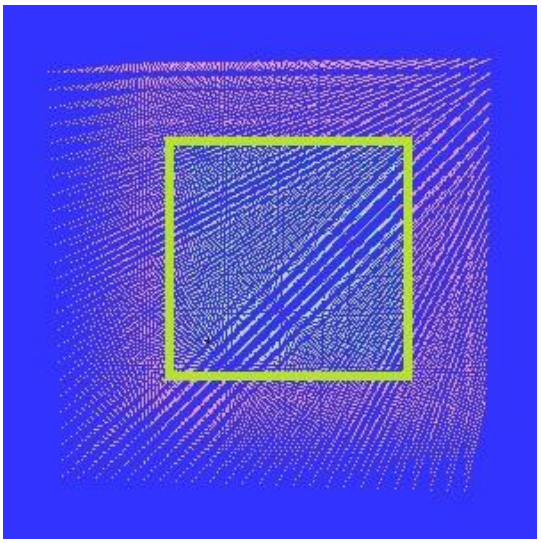

Figure 5: The central region which is shown by the green square is the workable region, which roots and endpoints are allowed. The red region are the borders which are reserved for the coral's geometry.

User Interface

Another import aspect of this project is the usability of it and the ease in which a user can create coral. The first thing that needed to be added was a way to select roots and endpoints. This was accomplished using a red sphere as a cursor. The user could move it around as he saw fit using and adding roots and edges with ease. Once the roots and edges are added, the coral can be generated. While viewing the generated coral, the geometry can be turned off in order to see the underlying structure. From there, the user may continue to add or remove roots and endpoints as he sees fit. Once finished, the structure can be saved as an STL file for fabrication.

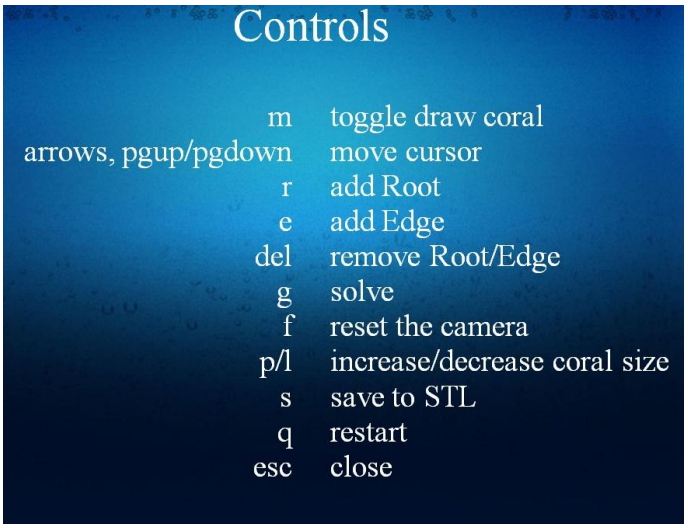

Figure 6: How to control the coral Generator

Results

Using the program, I was able to create shapes with the same look and feel as real coral.

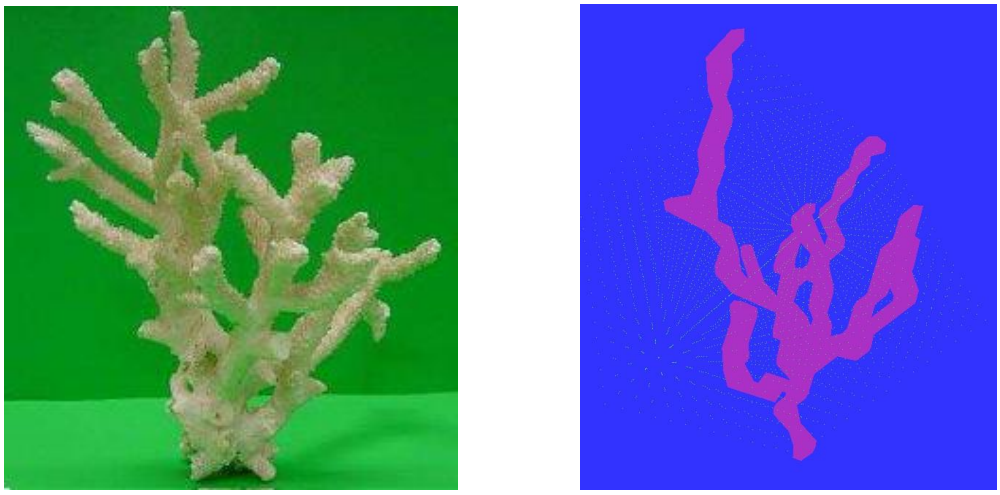

Figure 7: Left: Picture of real coral. Right: Coral generated using the coral generator

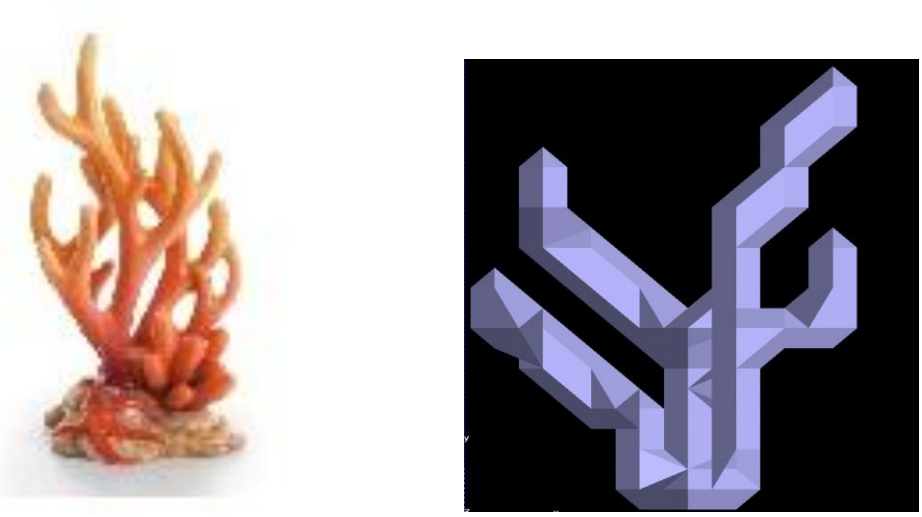

Figure 8: Left: Model of Coral that may be found for sale Right: STL render in Quickslice

Future Work

There are a few more additions I think would be great for the future of this project. The first is path refinement. This will allow for much more natural looking dendrites and more realistic shapes. Another addition is the addition of better lighting texturing. Currently the coral looks very two-dimensional making it hard to see its structure without moving around the viewport. By improving the lighting and adding textures, the interface will both be more aesthetically pleasing. The user will also have an easier time visualizing the coral being created.