David Buckley

Professor Tao Ju

Geometric Computing for Biomedicine

12/11/22

Final Project Report

Required Features:

* Dual Contouring – implement dual contouring in both 2d and 3d
  + Successfully managed to implement dual contouring in 2d but did not manage to implement it 3d.
  + One minor bug in dual contouring: if a line would be drawn off an edge it is connected back to the first vertex.
    - Fixed by changing placeholder value in index array from 0 to -1.
* Marching Cubes/Squares – implement these algorithms for 2d and 3d contouring
  + Successfully managed to implement marching squares but not cubes.
* GUI
  + Successfully implemented GUI in html.
    - Radio buttons for algorithm/image selection
    - Slider for iso value selection
* WebGL visualizations of voxel/pixel images
  + Not visualized using WebGL, but pixel images are placed underneath outline.
  + No display of voxel images.
* A sufficient collection of sample data
  + The Cryo image, a brain MRI, and an X-Ray of a breast lesion.
  + Could potentially find more sample data.
  + Could also implement a way for users to upload files.
    - Needs a more robust backend component to the webpage
    - Users could copy file paths from their personal computer into a text field.

Wishlist Features:

* Speed up algorithms
  + Not implemented
* Additional fairing options
  + Not implemented

GUI Development:

Relatively straightforward development process. All HTML components easily added. Radio Buttons grouped by element name to allow for only one selection in relative categories; checked attribute allows easy access of radio button value via JavaScript. Slider value attribute can easily get selected iso value. Add event listeners to each element to fire on changes or clicks and run the reload function.

Dual Contouring/ Marching Squares:

Implemented almost the same as in Mathematica. Data structure slightly different used objects to represent vertexes and edges.

Reload:

Reprocesses the selected image with the updated parameters and displays it. Calls main function and selected algorithm function.

Main Function:

Most rendering code comes from boiler plate template on <https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html>. Altered primitive type to lines to draw the curve. There are other options for line rendering but “LINES” renders from coordinates in a similar way to Mathematica, so it was the best option. “TRIANGLES” would be primitive type for 3d. Read image data of selected image and calculate the vertices and use convertToVerts() to calculate the screen coordinates in WebGL format. Also used vertex shader from this page that uses pixel coordinates as opposed to WebGL clip space coordinates as they align with image pixel coordinates vs having to recalculate at each creation of a vertex.

Convert to Verts:

Takes in an edge array and a vertex array of objects and iterates through edge array to construct the WebGL 1 dimensional coordinate array.

Read Me: