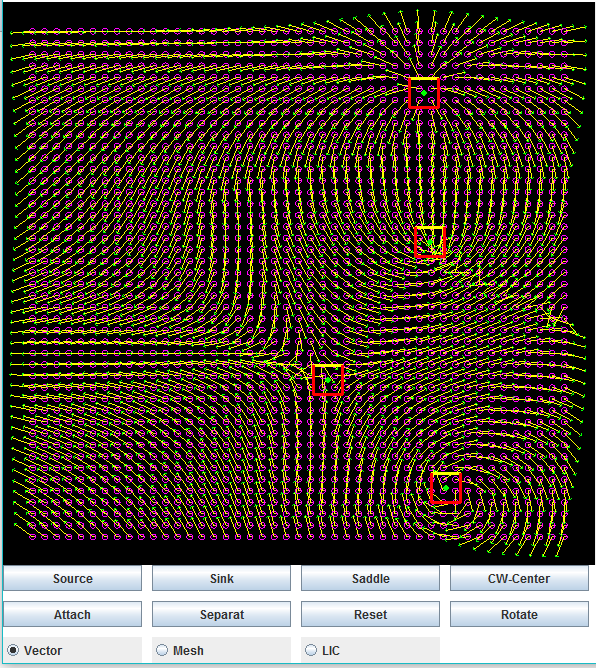
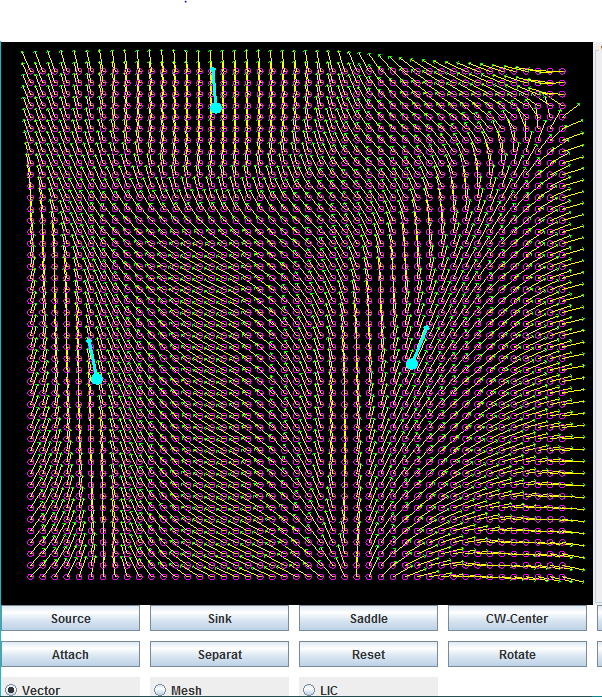
Week Report 12

Program Structure Summary

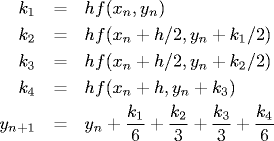
The purpose of the program was to simulate flow behaviors starting with a correlated vector field. The program uses Java Swing to provide user interface while Java OpenGL is used for a canvas to provide visualization features. The field is mapped out first by sample points. Given a designated amount value, the canvas’ height and width will be divided to generate the desired sample points. The sample points are represented by circles which were drawn in OpenGL by stacking multiple triangles on each other. Using the location of the sample points, a triangle mesh could be created with the vertices located at each sample point.

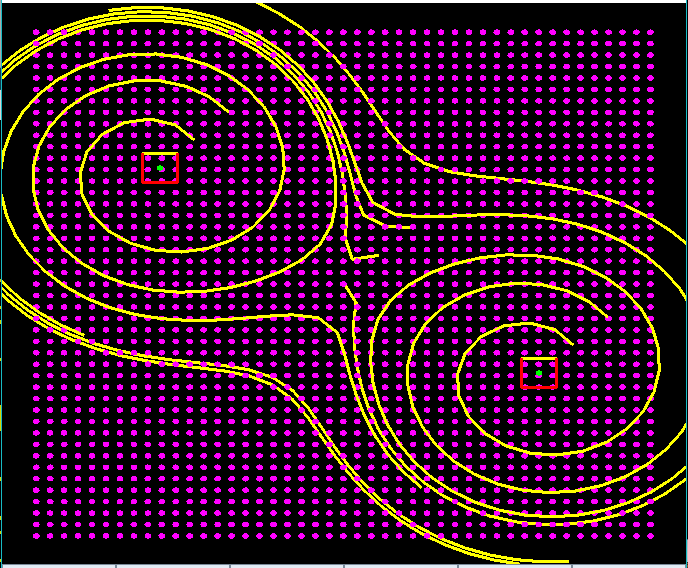
Singular elements and regular elements are represented with their own class. Singular elements have a jacobian matrix that hold different values that determine the flow behavior of the element when calculated with the corresponding field. The different types of singular elements are source, sink, saddle, center, and clockwise-center. Singular elements can also have variables stored to change its rotation angle and scalar effect on the field. User interface was also implemented to easier manipulate those values. Regular elements had different flow behaviors than singular elements because they were calculated differently which fundamentally consisted of X,Y location and vector values determining direction of the element. Each element is calculated against every sample point by comparing the distance between the element and point. Then the vector value that is calculated is stored for the sample point and since there will be multiple elements interacting the vectors values will be summed together to create this interaction. Singular and regular elements are each stored in their own list and so rendering all the elements would just require iterating those lists and plotting the elements. The interaction between the each of the elements are determined through weight values that are each calculated differently.

Regular Element Interaction

Singular Element Interaction

Runge Kutta Fourth Order Formula



To represent the flow of the field we use vector arrows, streamlines, and line integral convolution. When drawing the arrows, OpenGL can render the arrows using starting and ending position. However, to reduce visual clutter all vectors are normalized by a specific length value to evenly create a visual field representation. Streamlines are also drawn onto the field using the vector values of each point. At a determined point around the center of the element we perform RK4 integration to create an accurate estimate of a slope for the field and then take a step and repeat till our streamline is finished drawing. LIC allows to better visually represent the flow of the field and it is down by creating a white noise and convoluting it with a streamline. This convolution is done by recording the pixel values across the streamline, summing up each of their color values and dividing the total by the number of pixels. The final color should be gray and show a contrast between the pixels located around the streamline. The final LIC image should be stored in a 2D array so that it can be used by OpenGL to paste the image on the field.

Euler Streamline Representation

White Noise convoluted with multiple streamlines

This is my brief summary of how I interpret the program and how it works.

Overall Experience

I am grateful for not only working on technical skills such understanding graphics with OpenGL but also developing my ability to evaluate my effectiveness in learning new concepts and how to work on myself to manage my skill set for anything I decide to work on. Although this research experience has shaken me off my feet at a couple of moments, it has definitely brought me a better understanding of my skill set and how to improve on it. The workload you had given me was comfortable with room to experiment and let my mind grasp the concepts of the research. Difficulty only arose when working with a new feature or mathematical concept that was specific enough for me to see in the code to implement. I believe developing on my ability to draft my understandings and the software design can correct for this ineffectuality. The basics of OpenGL were easy to get started in but the fundamentals kept expanding as new features were introduced which took time to grasp. You made a very valid point when you suggested if I lacked the type of background to be able to match the competency I want to have. Coming from working in a backend environment, I was very inexperienced working with graphics. Learning something new takes some time regardless and if I had substantive career experience such as the one I got from this position, I am sure I can work to become more effective with such wisdom. I also thoroughly understand the value of having forethought in designs and understanding before starting the programming which very much fundamental in any area dealing with software and projects.

Moving Forward

I have really much appreciated this learning experience and challenge you provided for me. Being placed in a learning environment helped me realize that working on a project is quite different than how it is presented in the classroom and that the experience gained this summer will be indispensable in preparing me in the career I choose to go. The experience allowed an opportunity to apply the teachings I received in classrooms. The opportunity heighted my awareness of what competencies I may still lack so hopefully I can work on them while still in college. Although this was your first time mentoring under the undergraduate research program which may have been a lot of work for you, I want to thank for your time and patience to give me an experience and quality knowledge I would not have gained in a classroom alone.