Week 9 Report (8/4/16)

Singular and Regular Element Implementation

|  |
| --- |
| C:\Users\ootut\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updated.png |

The progress with regular elements is that the new formula is included which dealt with angles that allowed rotation. Also to explain the original formula, when the constant c is greater than 0 it is a separation, less than 0 is an attachment element, and as 0 the constant will make it a regular element.

To the right is an attachment element angled at a 35 degree angle. Problems that you can see with this implementation is that some of the vector pattern gets overdone away from the element origin. This is due to values dealing with weights and some constants. A little tinkering to get the visuals right is all that is needed.

|  |  |
| --- | --- |
| C:\Users\ootut\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Degree1.pngOriginal Formula with assumed angle 0 degrees. | C:\Users\ootut\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Degree2.pngUpdated formula with user input theta. |

The angle is calculated depending on user’s initial click and location of release. However, the vector values between the multiple elements overlap each other due to the weights being too small. This causes difficulty to differentiate where and the number of elements. The new weight formula involves some constant over dx^2+dy^2. Also, I will have to check not only how attachment elements interact with itself but also with singularities and regular elements by further looking into each of the weight values.

There was a problem with the rotation of singular elements. Upon review, we found that the source Jacobian matrix was being manipulated when an angle was set. However since the Jacobian matrix of source is the identity matrix, there should not have been any change in the calculations and so source and sink elements should not be affected by rotation. Therefore the problem has been narrowed down to my matrix multiplication and calculations. After I can get the rotation to work, I will implement user interface that allows the user to rotate the singularity by dragging rotation points.

Texturing

I tested the image based flow visualization demo in C++ with the original code and it worked as it should. I tried tinkering with the code and commenting out parts to see what did what. The initial drawing with the quadstrips is responsible for the warping animation and the second drawing processing with the blending technique and displaylists works with the black and white texturing. I tried to do the same in JOGL but sadly no success. The texture that is stored in the displaylist seems to not work the same. I believe it is the way JOGL works with buffered images and bytebuffers. That means that textures are encoded differently and must be formatted to accommodate to be accepted in OpenGL. Another difference is that JOGL doesn’t have the same direct access that C does with pointers. This makes binding and setting textures a little difficult. I will try to put a little more bit of time into JOGL and see what I can learn. Until then, I will be researching about LIQ and using it to implement streamlines.

|  |
| --- |
| Flow Visualization Texture |