James Guan

Tool Review

D3.js is a decent tool to use. It simplifies for loops and shortens the code. Another positive is that it allows you to use javascript so you can easily make the visualization compatible with a web browser. D3 using SVG makes it a bit slower because each pixel draw uses an HTML element versus if I had used a canvas, the whole picture generated would be on HTML element. So instead, I found a way to use canvas with D3 while maintaining the use of handy D3 tools. D3.js can also be very annoying to use because they change their function names drastically with their recent update to D3.js version 4, so trying to learn through tutorials online means that the functions you call may not be named the same. This makes your old javascript D3 code deprecated fast. I suppose you would make a new visualization for new data, but it still quite a hassle to relearn the syntax and workings of D3 with each new iteration. The changes from library to library also changes in syntax for the same functions previously so there's a lack of backwards compatibility support.

With Open GL, 3+ requires creation of meshes and shaders just to draw basic 2D shapes which is a huge hassle to get running. Instead, I found a library called SDL2 that takes OpenGL and simplifies the rendering of the shapes while still using OpenGL and similar functions. One problem with OpenGL is that it is very platform dependent because you have to compile it for each operating system. D3 is more flexible in this regard. However, D3 allows you to generate visualizations on your machine so you can probably do some more CPU intensive visualizations. OpenGL also has a lot of support with many different libraries keeping it up to date. It is very useful for 3D modeling of data because of OpenGL's capabilities and hardware support. Making it a better tool than D3 performance wise.

As for the Visualizations themselves, I chose to do a single blue color to show the data with luminousity. However, you don't really see much, so I made another data with two colors that contained

red and green. I did these horizontally so they fill up the screen more in OpenGL. The data definitely has more values to it so you can differientiate more of the data when two colors are added. However, I realized that if someone that was colorblind between red and green were to look at my red and green visualization, they would not be able to differientiate the data.

As for the d3 generated visualizations, I chose to do one in white canvas and red to black because some people might notice darker colors better over a white canvas. The grayscalish looking one is attempting to display less color and just use brightness to convey.