MSIM 441/541 & ECE 406/506

Computer Graphics & Visualization

Programming Assignment One

Input Analysis Using Histograms

Joshua Rich

Computer Science

Jrich006@odu.edu

10/4/18

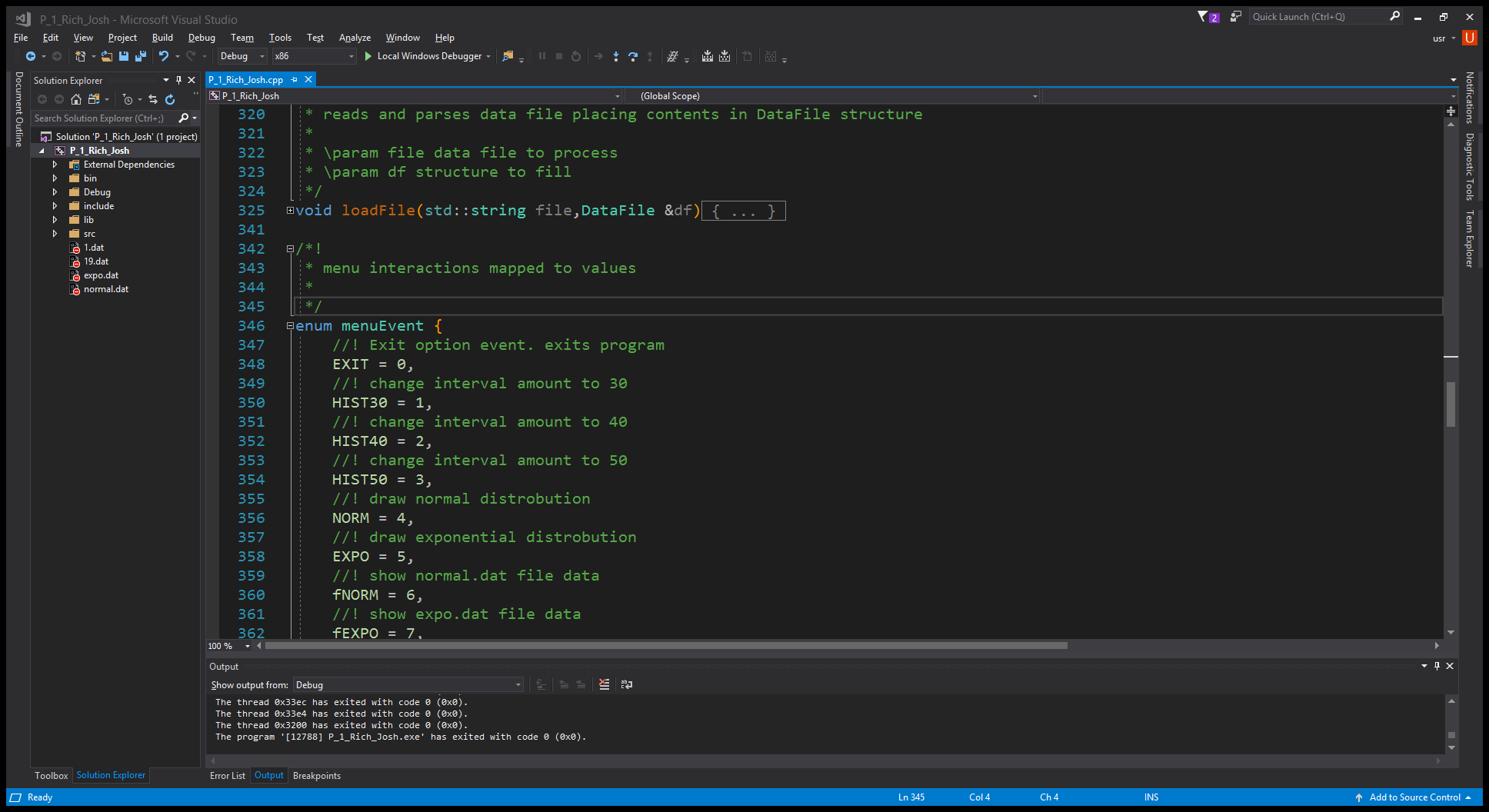
# Introduction:

Mathematics in itself is built from abstract constructs. The idea of equations, variables, and graphs can be represent in computer graphics to help the abstract nature of mathematics be more tangible. Like many mathematical subjects statistics works very well with being visualized. Some concepts of statistics like distributions and probability density functions can be difficult to understand while comparing real data. This is why a simulation is needed to remove the nebulous abstraction and parallel it with reality.

One of the most common kinds of statistical data structures is a histogram. Given a set of data and a number of intervals a histogram can be created by counting the amount of data points that lie within those intervals. As the histogram is a representation of empirical data, it is discrete and can only represent the input data. However histograms may fallow a specific pattern that can be comparable to a distribution. The normal distribution can be related to data sets that have a semi-regular pattern. The normal distribution can show the average or mean of the data set and the standard of deviation. The exponential distribution can be related to data sets that have a declining trend. The slope of that trend can be characterized in this distribution.

For this assignment, multiple data sets were given that, when made into a histogram, fallow the pattern of either a normal distribution or an exponential distribution. The program created for the assignment takes in the data sets, creates a histogram of the data, and visualizes the histogram along with a visualization of a normal or exponential distribution. The user can then manipulate the variables to the equations of either distribution to relate it to the histogram. Certain variables can be changed to better the simulation like, the interval size, the parameter step, or which distribution is shown.

# Program Design:

The program is designed and organized around input, computation, visualization, and interaction in that order. The window was configured to have a single RGB framebuffer because the only times the display will be updated is after interaction. There are two types of interactions, menu events and keyboard events.

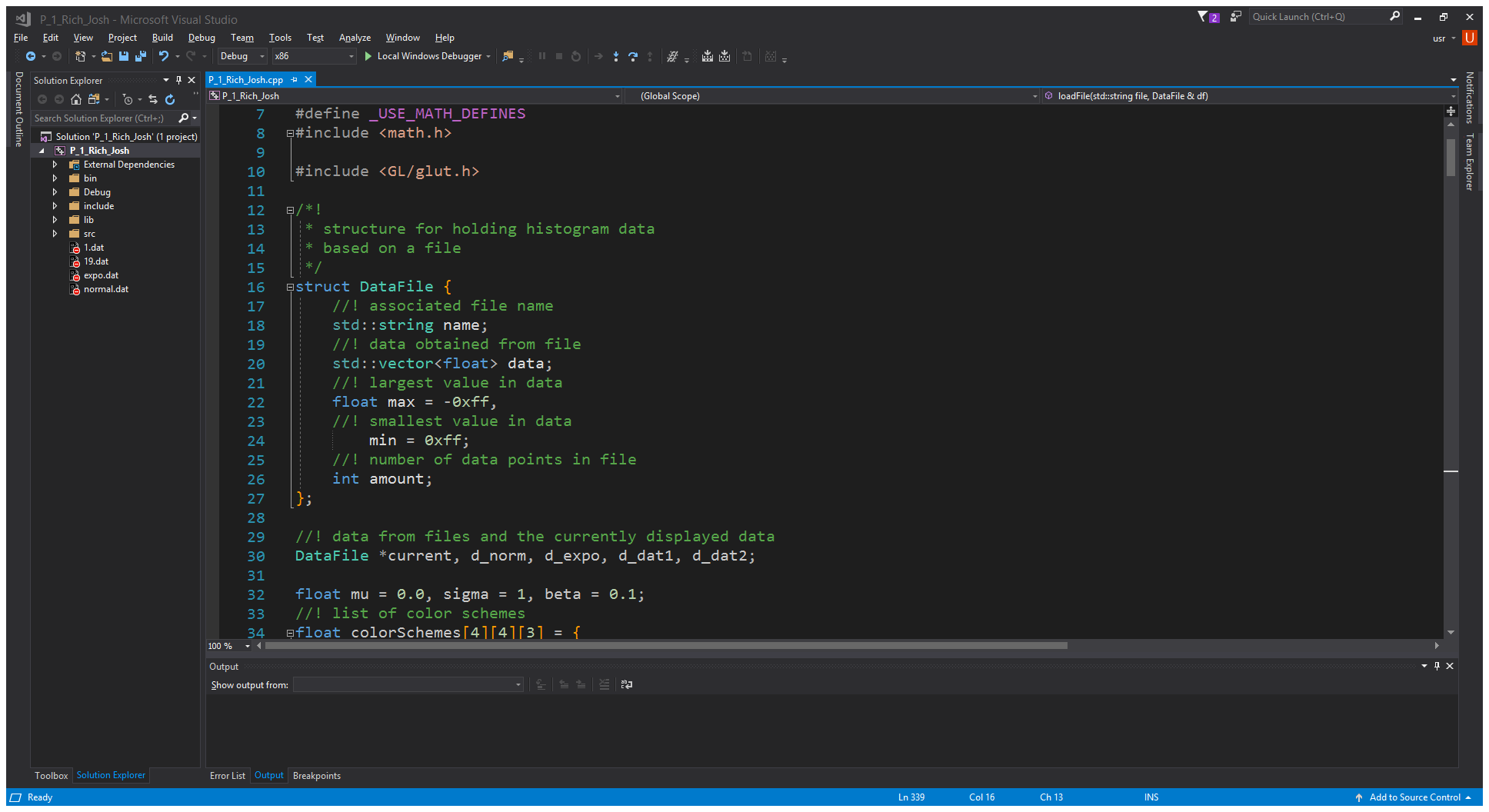
Menu events are facilitated by GLUT’s menu system. GLUT registers events with a function callback that is provided (seen on line 387). From this function an event value is compared with a menuEvent which are mapped to indexed values (see fig. 1). These menuEvents and the menu listener allows the user to change certain variables of the program like the number of intervals, the shown data file, shown distribution, the distribution variable step, color scheme, and also allows the user to exit the program.

Figure 1

Key events are handled by providing GLUT two keyboard callbacks; keyboardDown (line 255) and keyboardUp (line 278). To allow for the user to hold a key continuously, the key listener was set as an idle function checking control variables. The control variables are changed when a key is held or released in a callback allowing to register a continuous press. Up and down keys increment or decrement the sigma or beta variable, depending on distribution being displayed, and the right and left keys increment or decrement the mu variable. On the press of any directional key the histogram is recalculated to accommodate the update and the window is redisplayed.

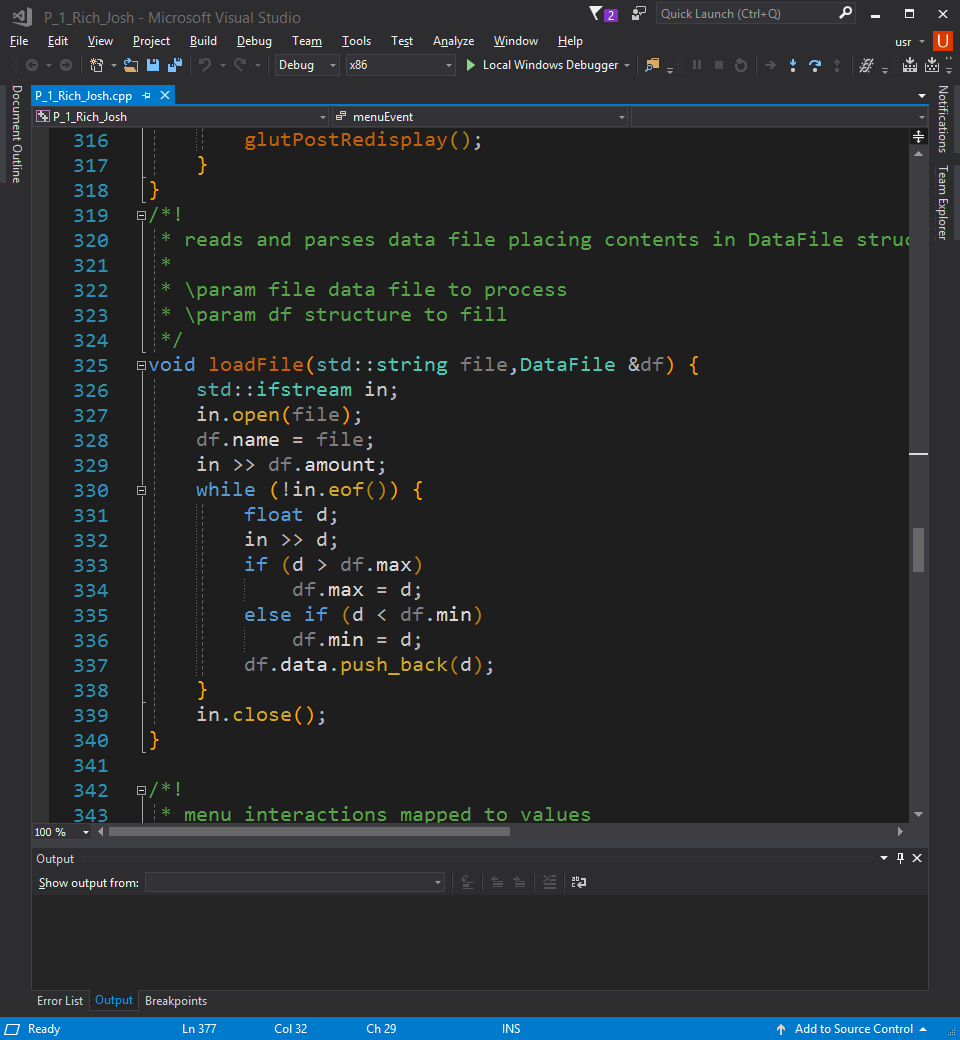
The calculation of the histogram is done in the function calcHistogram (see line 61). The algorithm is not the most efficient means of creating a histogram, however it is a straight forward means. It starts with a list of which the length is the amount of intervals set. This list represents the bars, or bins, of the histogram. The interval step is computed by finding the range of the current data set and dividing by the interval amount. For each data value the algorithm checks if it lies in a interval of times to times plus , where is a value from the minimum data point to the maximum incrimented by (seen in lines 65-74). This algorithm will break out of the loop when the correct interval is found to avoid needless looping. The outcome of this function is the list’s values are the bars of the histogram.

The visualization is straight forward. A display callback is provided to GLUT that calls other display functions. The axis, histogram, and distrobution is drawn respectivly with the colors defined by the selected color scheme. Color schemes are a two dimensional array of thee component color vectors. The first dimension is the list of colors in a color scheme. The second dimension is each of the colors applied to the histogram, distrobution, axis, and background in that order. The set color scheme will also effect the text labels. Text is drawn using GLUT’s glutBitmanCharacter function with the built in font GLUT\_BITMAP\_8\_BY\_13.

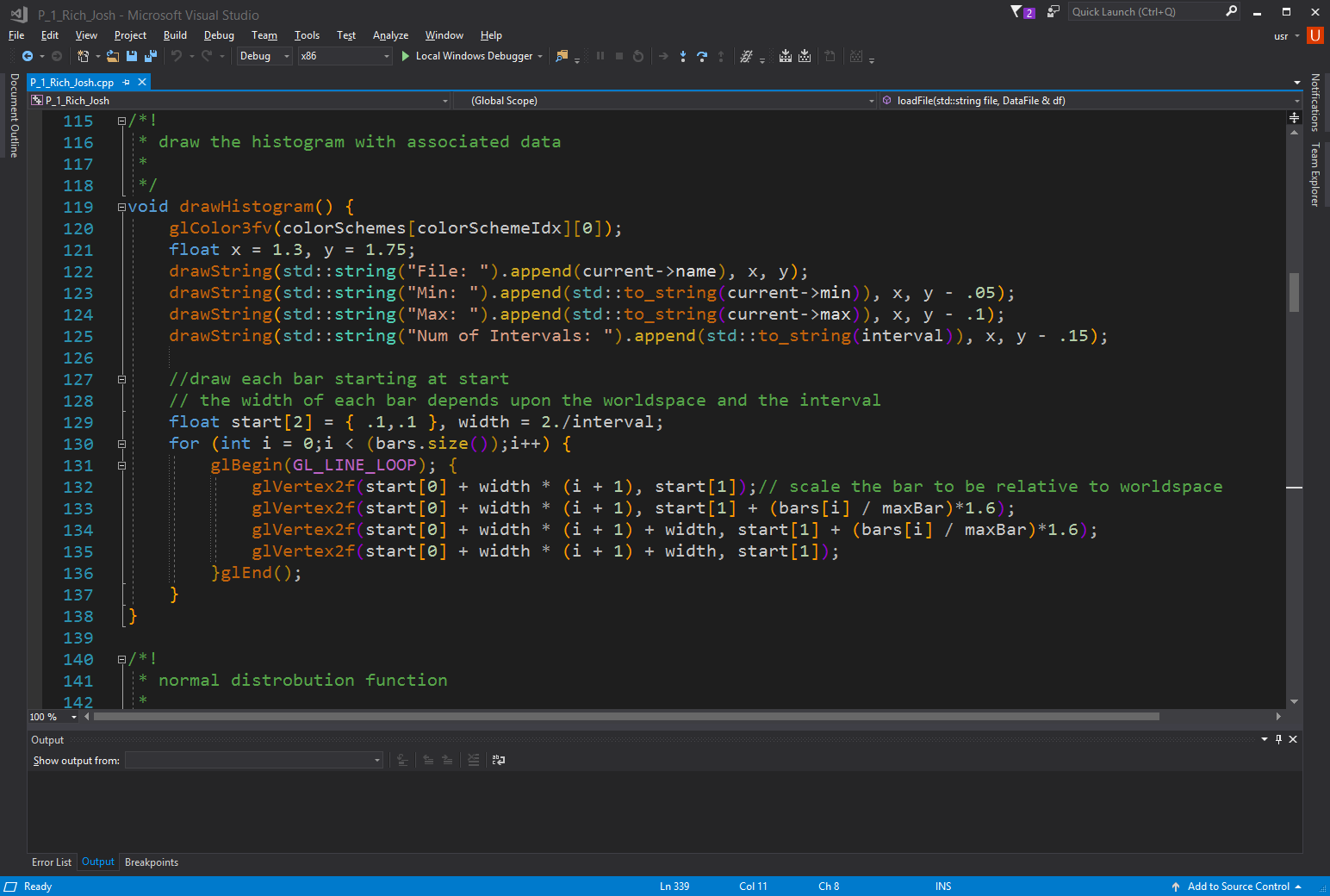
Each data file is loaded using a simple file reading algorithm that while taking in the input data finds the maximum and minimum values. Each data file is organized into a DataFile struct (see fig. 2). In a DataFile the file name, max and min, number of data points, and a vector of data points. To activly switch between DataFiles without needing to read in the file again, all of the used files are loaded at the start of the program and a pointer to one of the files is set as the one currently being displayed. When the menuEvent fNORM, fEXPO, fDAT1, or fDAT2 is called by use of the menu system the program will switch the current to a pointer to the selected DataFile and recalculate the histogram.

Figure

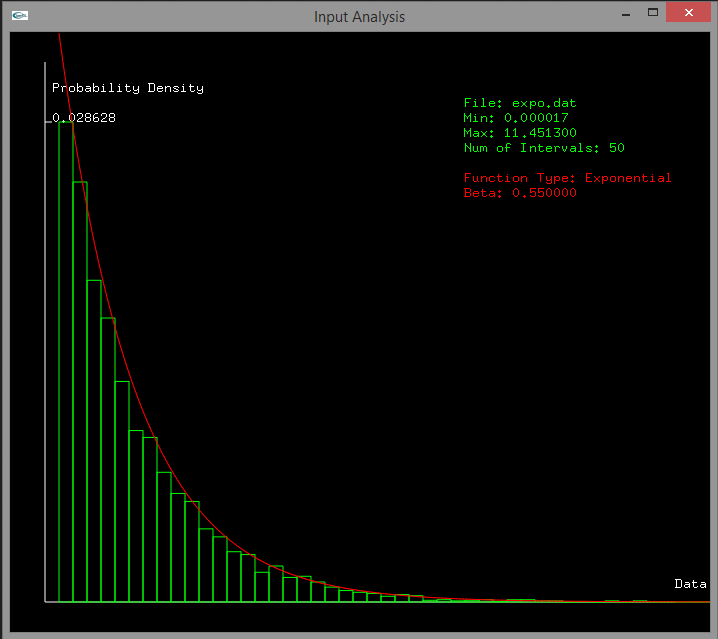
# Results:

The solution to each task was made to be straight forward to the task provided.

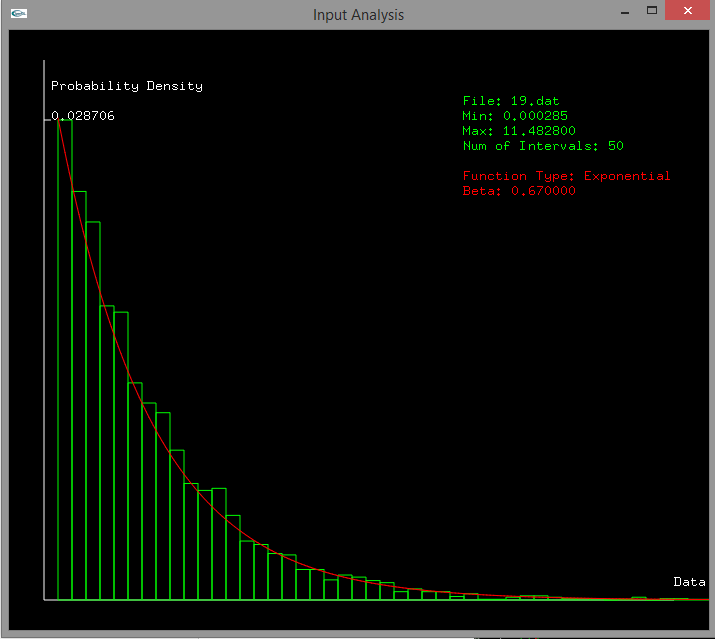
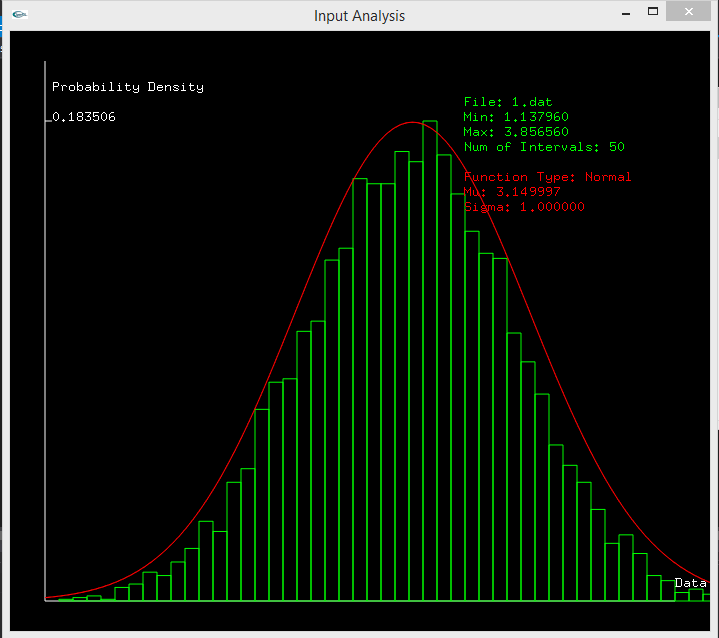
* 1. For reading in the data files, I used a std::ifstream and opened the file. I passed a refine to a DataFile and filled its members with the appropriate values. Pulling in the amount of data first into the DataFile’s amount member. Then until the end of the file reading in each data point, finding if it is max or min so far, and adding it to the vector stored in DataFile. At the end of the function the files is closed and the referenced DataFile’s member functions are filled.
  2. For the histogram visualization, the projection is set to be a coordinate system from zero to one in the x and y axis. This was to allow for the equations to be more easily mathematically accurate. The histogram is generated by the algorithm explained in the prior section and displayed by positioning each bar as a fraction of the width of the graph and the amount of intervals. The height is calculated backwards by multiplying the expected max height by the value of the bar divided by the maximum bar.



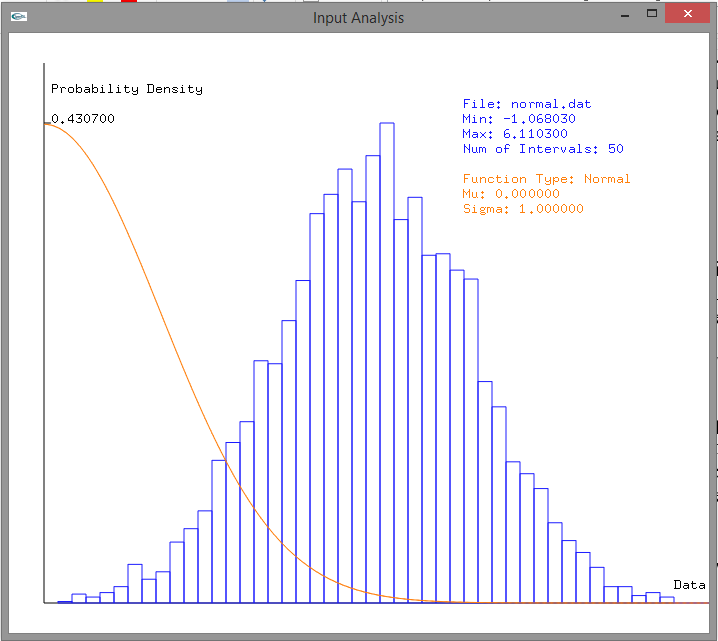
* 1. The probability density functions are visualized with GL\_LINE\_STRIP and graphed across the x axis. For the normal distrobution, the x and y axis needed to be scaled to an appropriate size to accommodate the mathmatical range of the function. A normal distrobution lies between -3.5 and 3.5 on the x axis, this is seen on any z-table, and 0 to .5 on the y axis. The function glScalef was used to accommodate the difference in scale. The same also needs to be don’t to the exponential distrobution. The exponential distrobution lies between 0 and 5 on the x axis and 0 and 1.6 on the y axis, when . to plot either distrobution a floating point value x is incremented by a small value to some upper limit. Both the lower and upper limits are varied to accomidate the size of the graph. For the file expo.dat the was determined to be around .55, however the given value for the file expo.dat is 1.25. after some research the slope that is similar to the one seen in expo.dat is shown to have where , so according to the chart below the line that is similar to expo.dat is .[[1]](#footnote-1) this shows that my representation is acrurate.



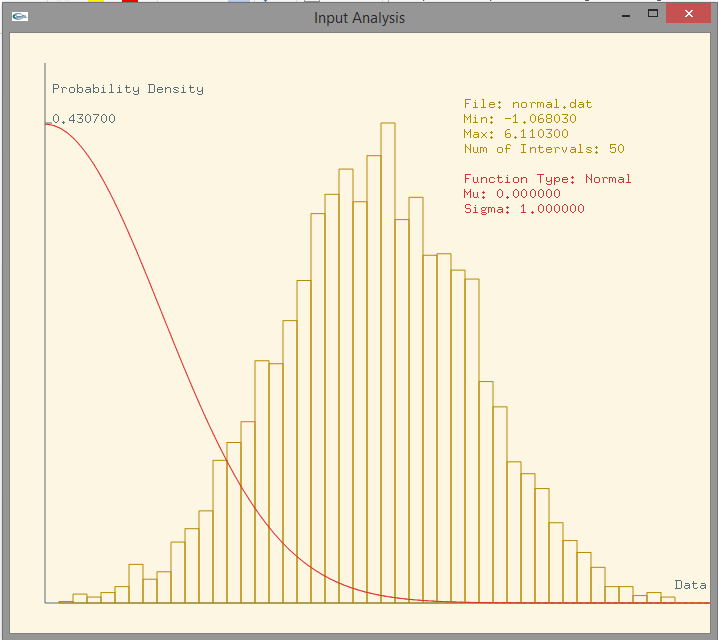
* 1. The resualt of reading in the additional data files was similar to the main files. For my specific project I used the files 1.dat and 19.dat. the file 1.dat is the normal distrodution with the found variables and . The exponenital distrobution is in 19.dat and its variable is .



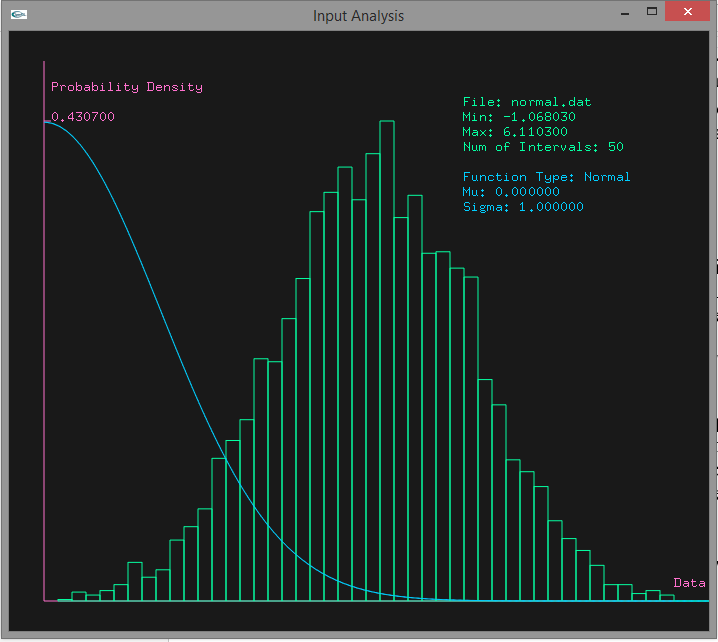
* 1. As an additional feature the user is given the ability to change the color scheme of the program. There is a submenu called color scheme which contains defualt, light, solarized, and vaporwave. My personal prefrence is to the vaporwave color scheme, as it has a high contrast and interesting colors. The defualt was left in as the main color scheme so to look like the examples as much as possible.



Light Color Scheme



Solorized Color Scheme



Vaporwave Color Scheme

# Conclusion and Discussion:

This project was fairly straight forward and the most stressful part to me is this report. I had trouble at first figuring out how to make the histogram algorithm, however I still found it interesting trying to implement the mathematics. The algorithm itself is fairly inneffiecent because it iterates through most of the intervals for every data point. So what I could’ve done to avoid the extra loops and if statement calls is calculated the index of the interval. This could be done by. I have not tested this fully and don’t know why there is a disparity of 6 between the theoretical and actual interval index. Another problem with this solution is that even when adding six the index is still off by one. With more playing around this method might work and remove the need to iterate over 30 or so intervals checking if they fit. In this project I also had some fun with doxygen. Although I had heard about it, I had never used it before and I am surprised by its usefulness. It took me about two days to finish the programming side of this project, with about three hours each day. I spent a lot of time tweaking the positions and looks of things, as well as testing the histogram algorithm. I think through all of the theoretical topics we touch during lecture, if we were to discuss a recommended organization and structure to graphics projects it would help in the more practical side of openGL.

1. Found here: <https://en.wikipedia.org/wiki/Exponential_distribution> [↑](#footnote-ref-1)