**Programming Project Report**

Student Blake Williams

Student 010974718

**Problem Statement:**

The goal of this programming assignment was to utilize OpenGL to render a penny using 2 provided .txt files. These files contained the depth values and color values of the penny. Then the penny would be rendered to show the color depth file, depth values using lines, and Phong shading. The inputs of this program are to allow users to rotate the penny to see the depth of the penny, using various keys to rotate around the xyz axis. The main error handling was if users click other buttons other than the set keys, luckily Open GL simply ignores all keypresses that are not already coded into the function.

**Design:**

The design of the program was as follows. First get the skeleton up, including main, keyboard functionality, init, and the #inlcudes. Then focus on the depth read in and display. Next focus on the color display, and finally the phong display. As each display required other functions they were to be added alongside development of that function. Pros of this was a much more structured code. There were no noticeable cons to this approach.

**Implementation:**

First was the readDepthFile() function. This took in a filename, and looped through all the entries and stored it into a array called depthData[width][height] where the width and height were 500. If the numbers are read normally then the image will be rotated in a weird way, so to fix this another nested loop mirrors the data round the mid-vertical axis. The outer loop iterates through each row, while the inner loop iterates through the first half of the columns. This swaps the position [i][j] to its counterpart on the opposite side of the row by doing 499-j. This corrects the image to make it appear upright on initial display. This process will be repeated in the readColorFile. Next the depthDisplay function was added which clears the window when called and iterates through the depthData array in steps of 5. It calculates the depth of the penny, then offsets it with /20 to make it better for visualization. Then uses the nested loop to define a quadrilateral with 4 vertecies and uses it to create a wireframe based off the penny height which was calculated previously. Then the loop continues through the entire array until a competed wireframe is output. Next is the read color file which operates similarly to the readDepthFile(), the key difference being the addition of 3 arrays instead of 1 array reading in data. Other than that it is the same as readDepthFile(). Same with the colorDisplay() function, is almost identical to the depthDisplay() function with the only difference being the addition of a glColor3f() function call which adds color to the penny. Finally, the phong display. It has the same initialization as the other displays, but the key difference is the introduction of init\_material(), init\_surface, init\_normals(). Init\_material() configure the material properties such as ambient, diffuse, and specular reflections are configured here. Init\_surface and init\_normals are used to calculate the 3D positions of (Px,Py,Pz) and (Nx,Ny,Nz). They determine how light reflects off surfaces. Finally, the function iterates over the surface, drawing polygons that represent the surface sections. Each vertex is set to the normal vector with glNormal3f and specifies vertex position with glVertex3f. These are the main functions that are utilized in this program. Another being the keyboard() function which allows for user input to rotate, and change the mode of display for the penny.

**Testing:**

Testing the program was as follows. First when implementing the depthDisplay() initially the image would be rotated upside down. By changing the [i][j] values I was able to determine how to flip the image around via changing these variables in the input. Once each of the functions were implemented. I would run the program and rotate each display mode to make sure it properly displayed, if there was something wrong with a display, then I would know exactly where to look as each display modes are divided into separate functions. This testing continued until a proper display function was created for each of the 3 modes.

**Conclusions:**

Overall, the project was a success as each of the 3 modes were properly displayed, and were able to be rotated about the xyz axis based off user input. Next time I would attempt to make this program utilize classes instead of a single file. This project took about 7hrs to complete with 30minutes dedicated to the report.