**Computer Graphics: Project 4 Report**

Matthew S. Hartstein

010567629

Problem Statement:

The goal of this programming project is to create an OpenGL program that uses Phong Shading to display geometric objects that have been scanned with an RGBD camera. Color information (RGB) will be stored in one JPEG image and depth information (D) will be stored in a second JPEG image. The provided text files were used to extract RGB and depth values for my program.

Normal inputs to my program involve keyboard and mouse inputs that are given by each user. Ultimately, my program outputs a “Main Menu” after successfully extracting both text files (depth and image). Most importantly, my program outputs/displays a Penny model that the user can interact with by pressing various buttons located on the keyboard.

Plenty of error handling was used throughout this project to ensure a high quality, finished product. Simple “cout” statements were used to test variable values, as well as writing to 2 separate log files, which contain the extracted information from the two original text files, but in a more organized manner. These log files were mainly used to ensure that my program was correctly extracting each value (or triplet of values) correctly.

Design:

The design decisions that I made throughout this project heavily revolved around the sample code that Dr. Gauch provides on his personal website; “surface4.cpp” was extremely helpful with Phong Shading, Lighting, and with the initial set-up of my program. I used several 2D arrays to store the depth and RGB values from both text files; these 2D arrays were extensively used throughout my program to ensure correctness. Numerous functions and OpenGL callback functions have also been implemented to provide the functionality of my program. As a side note, I originally went with a class to store my penny object, but quickly realized this wasn’t needed since only one object is needed to be displayed. Overall, I am very satisfied with my design decisions because my code is extremely organized, easy to understand, and contains comments throughout its entirety.

As for the algorithms that were used, these were mainly given to us in the “surface4.cpp” file located on Dr. Gauch’s website. Scaling, converting, and transposing/rotating the values stored in each of the created 2D arrays was paramount to this assignment, which, at first, was extremely hard to understand. Over time, I was able to figure this out and successfully implemented the proper code that displays a 3D model from a picture of a penny. Although I was able to successfully utilize the depth values, I was not able to figure out how to incorporate RGB values.

Implementation:

My implementation process was fairly straightforward. I started with Task #1, which involved Model Creation. For this section, I created a polygonal model to represent the penny. This was accomplished by modeling the top surface of the penny with a 2D surface, D(x, y), where the ‘D’ is the depth and (x, y) are the sample locations in the JPEG image.

Instead of directly reading the pixel values from each image, Dr. Gauch provided an alternative route by simply allowing us to extract this information from two text files: “penny-depth.txt” and “penny-image.txt”. Both of these files are included in my project’s directory folder and should be located properly when the program runs. After storing this information in a polygonal mesh, I scaled each depth value into an appropriate range in order for the display function to work as expected.

For Task #2, I extended my OpenGL program to display the model surface. Then, in order to view the surface from different angles, I extended my program again by adding code to the display callback, which calls “glRotatef(xangle, 1, 0, 0)”, “glRotatef(yangle, 0, 1, 0)”, and “glRotatef(zangle, 0, 0, 1)”. Then, I added code to the keyboard callback to increase/decrease the global variables that store values associated with “xangle”, “yangle”, and “zangle”, when certain keys are pressed.

For Task #3, I extended my program yet again to read the color image, which gets stored into three 2D arrays: “red[SIZE][SIZE]”, “green[SIZE][SIZE]”, “blue[SIZE][SIZE]”. As expected, each RGB value ranges from 0 to 255, which needed to be converted in order to work properly. Thankfully, Dr. Gauch helped me understand the conversion process and why it was mandatory for displaying purposes. Next, I needed to extend my program to calculate and save the average RGB color of the penny’s pixels, which unfortunately, I was not able to accomplish. If I had a second chance, I would immediately figure this out, but I simply ran out of time to finish implementing it.

Moving on to Task #4, I was instructed to display the penny with the stored RGB values. As previously mentioned, I was not able to complete this step but spent many hours trying to figure this out. I believe the problem revolved around having to convert these values into a range from [0..1] in order to display properly.

For Task #5, I implemented a “Phong Display” that displays my penny’s surface by using Phong Shading. In order to do this, I added code to my program to calculate and save the surface normal at each (u, v) location on the surface. Next, I included “#include shading.cpp” in my program to provide shading to my 3D model. Then, I modified the “init()” callback function to define my light positions and colors. Also, I specified the material properties to display the polygons using Phong Shading. Overall, by adjusting the Phong Shading parameters via keystrokes, my program is able to dynamically change the view of my penny, which makes it look more-or-less realistic, depending on the values you assign to it.

For the final task – Task #6 – I implemented a keyboard callback to allow user to switch between display callback functions. Although the RGB option doesn’t work, the depth values work as expected, as well as the Phong Shading and lighting that is provided in the “shading.cpp” source file. My program implements this functionality as a “Main Menu”, which the user can interact with until they decide to exit the program. As mentioned, “surface4.cpp” was used as referencing material for this project. Overall, it took me roughly 35 hours to complete this program.

Testing:

As mentioned earlier, “cout” statements were used to verify correct variable values contains within 2D array structures, as well as global variables that periodically get updated. This method was used to ensure proper functionality of my program. Normal inputs involve keyboard strokes (and an extra mouse feature), which are both given by the user. A special case was used when designing my surface grid, which is hidden, to only display the 3D model of the penny. I’m not sure if this is what the instructor preferred, but the example pictures didn’t show grid lines, therefore, I decided to blend them into the background (black on black).

Although some aspects don’t work as expected, I believe the majority of my program meets the requirements associated with our problem statement. I was able to extract the data from both text files and - not only saved them to 2D arrays - but then used these values throughout my program to display the model. Due to the ongoing virus situation, office hours were moved/postponed, emails were overflowed, and the transition to online lectures/office hours has been extremely frustrating, to be completely honest. Despite these challenges, I’m extremely happy with the overall result of my program! Located below, I’ve included numerous screenshots of my program:

A close up of a coin

Description automatically generated

**Figure #1**

**A close up of a coin

Description automatically generated**

**Figure #2**

**A screenshot of a cell phone

Description automatically generated**

**Figure #3**

**A close up of a coin

Description automatically generated**

**Figure #4**

**A close up of a coin

Description automatically generated**

**Figure #5**

Conclusion:

Overall, I am extremely satisfied with the progress that was made throughout this strenuous situation with the ongoing virus/classes moving online, as well as virtual office hours. Despite these challenges, I was able to complete most of this assignment, which makes me feel very accomplished. But that doesn’t discard the hard work that was put in to making this program, as well as learning new OpenGL methodologies via Google searches. Therefore, I would most definitely classify this project as a success. Overall, I spent roughly 30-35 hours working on this project and learned a lot of new and useful information that I will apply to future projects that deal with C++/OpenGL.