**Programming Project Report**

Clayton Warstler

010971514

**Problem Statement:**

The goal of this programming project is to create an interactive OpenGL program that displays geometric models of objects that have been scanned with an RGBD camera. With RGBD cameras color information stored in a text file and depth information stored in a separate file. We want to represent a penny with OpenGL. The inputs for this program are the keys 1,2,3,x,y,z,t,r to represent axis and modes. The output for this program is a graphical representation of a penny in whatever mode the user is in. There wasn’t any real error handling, as the program just moves the penny around.

**Design:**

For this programming project, I decided to use Dr. Gauch’s surface4 as starting code. I didn’t really have any major data structures or algorithms for this program other than OpenGL. I did have to change the way the program did lighting and materials. I tried to implement what was provided, but it just broke everything. I found online a way to implement lighting and materials using the following code:  
  
 GLfloat ambientLight[] = {0.2f, 0.2f, 0.2f, 1.0f};

GLfloat diffuseLight[] = {0.8f, 0.8f, 0.8, 1.0f};

GLfloat specularLight[] = {1.0f, 1.0f, 1.0f, 1.0f};

GLfloat position[] = {1.5f, 1.0f, 4.0f, 1.0f};

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambientLight);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuseLight);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, specularLight);

glLightfv(GL\_LIGHT0, GL\_POSITION, position);

GLfloat mat\_ambient[] = { 128.0f/255.0f, 85.0f/255.0f, 66.0f/255.0f, 1.0f };

GLfloat mat\_diffuse[] = { 128.0f/255.0f, 85.0f/255.0f, 66.0f/255.0f, 1.0f };

GLfloat mat\_specular[] = { 128.0f/255.0f, 85.0f/255.0f, 66.0f/255.0f, 1.0f };

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

The code above create the materials and lighting, but for some reason did not break it, unlike what was provided by surface4.

**Implementation:**

To start this assignment, I started with the line loop display. This was a simple four glVertex3f functions looping over all 500x500 points. After that one, I moved on to the RGB version. This version was a little more difficult, as I had to figure out how I needed to save the colors, but it was basically the same as the depth information. After saving that, I just copied the line loop code and added a glColor3f function to it. Phong Shading was the most difficult of the three displays. For this one, I had:

glBegin(GL\_POLYGON);

glNormal3f(Nx[i][j], Ny[i][j], Nz[i][j]);

glVertex3f(Px[i][j], Py[i][j], Pz[i][j] \* 0.001);

glNormal3f(Nx[i + 1][j], Ny[i + 1][j], Nz[i + 1][j]);

glVertex3f(Px[i + 1][j], Py[i + 1][j], Pz[i + 1][j] \* 0.001);

glNormal3f(Nx[i + 1][j + 1], Ny[i + 1][j + 1], Nz[i + 1][j + 1]);

glVertex3f(Px[i + 1][j + 1], Py[i + 1][j + 1], Pz[i + 1][j + 1] \* 0.001);

glNormal3f(Nx[i][j + 1], Ny[i][j + 1], Nz[i][j + 1]);

glVertex3f(Px[i][j + 1], Py[i][j + 1], Pz[i][j + 1] \* 0.001);

glEnd();

This was simple, as it was already provided by the surface4 program. After, I had to figure out how I was to save the lighting and materials, but I used the code block above to do that.

**Testing:**

For testing, the only inputs that were allowed were 1, 2, 3, x, y, z, r, and t. At the end of the assignment, everything was working as expected.

A screen shot of a computer

Description automatically generated

A close up of a coin

Description automatically generatedA computer screen shot of a coin

Description automatically generated

**Conclusions:**

Overall, this project was a success. The program displays three different versions of the penny, a line loop display, RGB display, and a Phong Shading display. This project probably took about three days to complete, with a lot of time spent trying to figure out why something wouldn’t work.