**Programming Project 4 Report**

E. Mitchell Norris

010782482

For project 4 I was asked to build a program that took in two input files. One containing the RGB color values of an image and one that contains a depth map (the Z values) of the same image. The program then takes this information and builds a 3D model in openGL of that image.

**Design/Implementation**

Starting in the init function, I decided to define the Ortho as -2 to 2, -2 to 2, and -4 to 4. The 4x4 X and Y dimensions is fine but the Z ortho had to be expanded to prevent clipping when rotating the 3D model. GL\_DEPTH\_TEST is enabled, and the shade model is set to GL\_SMOOTH. I also initialize 3 lights taken from balls.cpp. I then call a void function I defined as data\_extractor which takes in a Surface object as it’s only parameter. The Surface object is defined as 9 250x250 arrays, 3 arrays for RGB color, 3 arrays for XYZ positions and 3 arrays for XYZ surface normals. Data\_extractor open 2 text files, penny-image and penny-depth, penny-image must hold RGB values for each pixel of an image and penny-depth must hold the Z values for each pixel in a depth map. In order to get all the information from the files, I loop over the entire penny-image file and if any 2 of the RGB values is over 245 then all RGB values are set to 0 else they are scaled to a value between 0 and 1 and saved to a position in the Surface array defined by the X and Y value. Since the depth map is the same size as the image the Z values out of the penny-depth file are saved to the Z portion of the Surface class along with the color information. I also use this time to find the average color values which I use later to define the material of the object. At the end of the data extraction I call the surface\_normals function defined in the balls.cpp source code to find the surface normals of each polygon.

In my display function I call 3 rotation functions one for each X, Y and Z rotation which will rotate the object when the user pushes defined key-bindings. I also have 2 display\_surface functions which take in a Surface objects as a parameter and will display the information saved in the Surface class on the screen as ether black and white line loops or colored polygons.

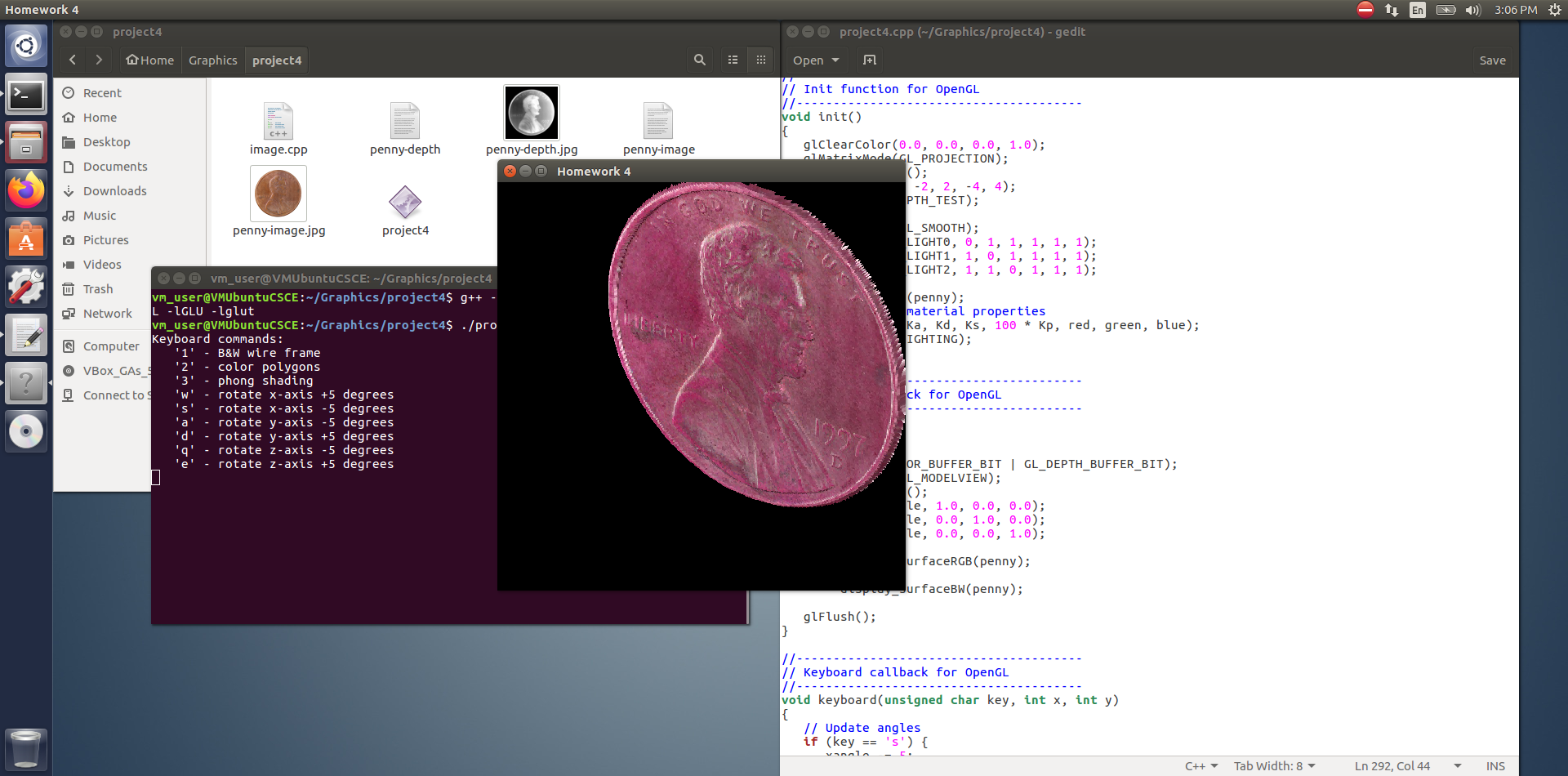
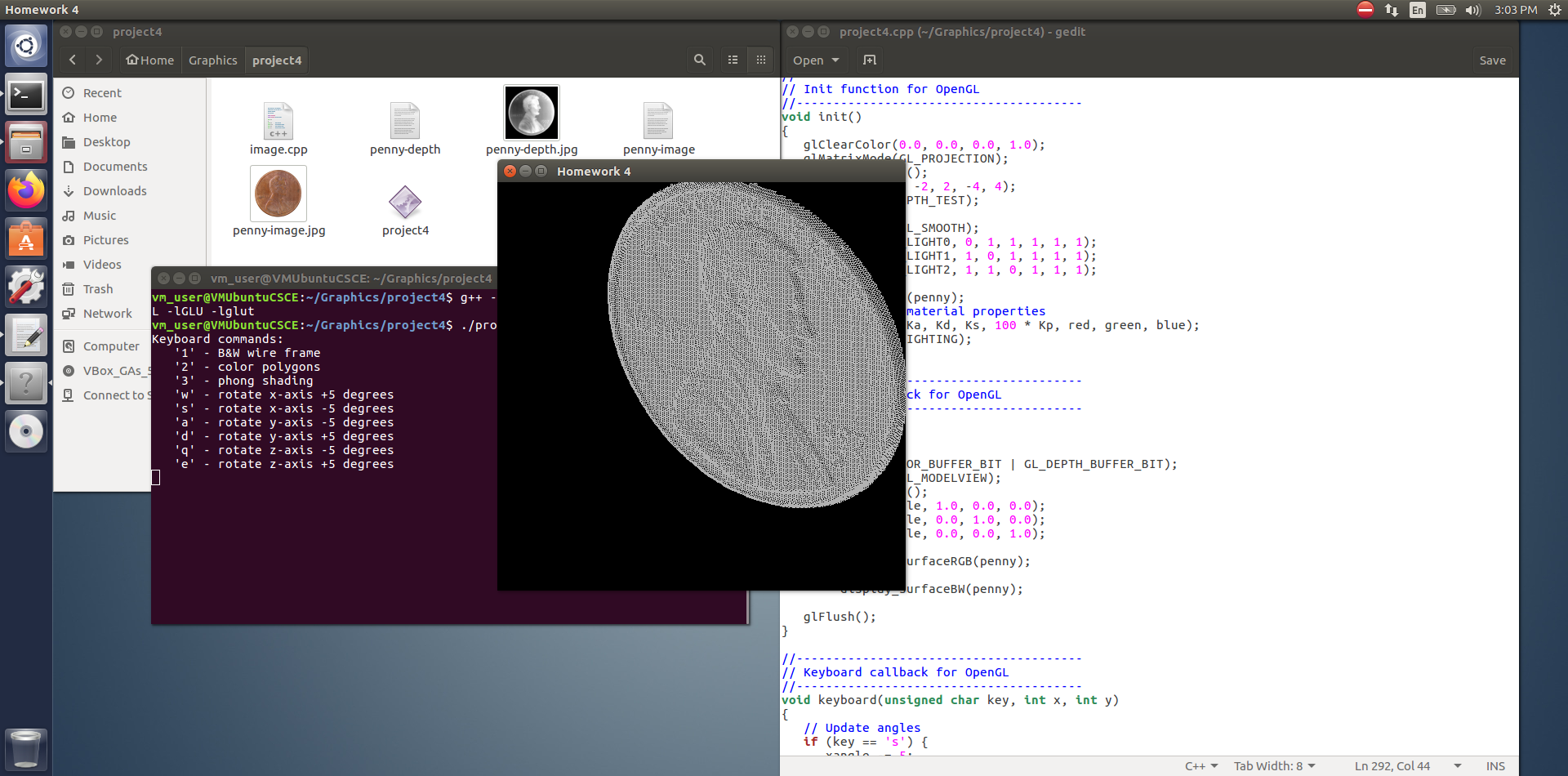
As per the instructions the program opens as black and white line loops. Also pressing 1 will return the 3D model to this state. Furthermore pressing 2 will display the colored polygons version of the 3D model and pressing 3 will display the Phong shaded version. I also implemented some key-bindings for rotation. Use W and S to rotate around the X axis, Q and E to rotate around the Z axis and A and D will rotate the model around the Y axis.

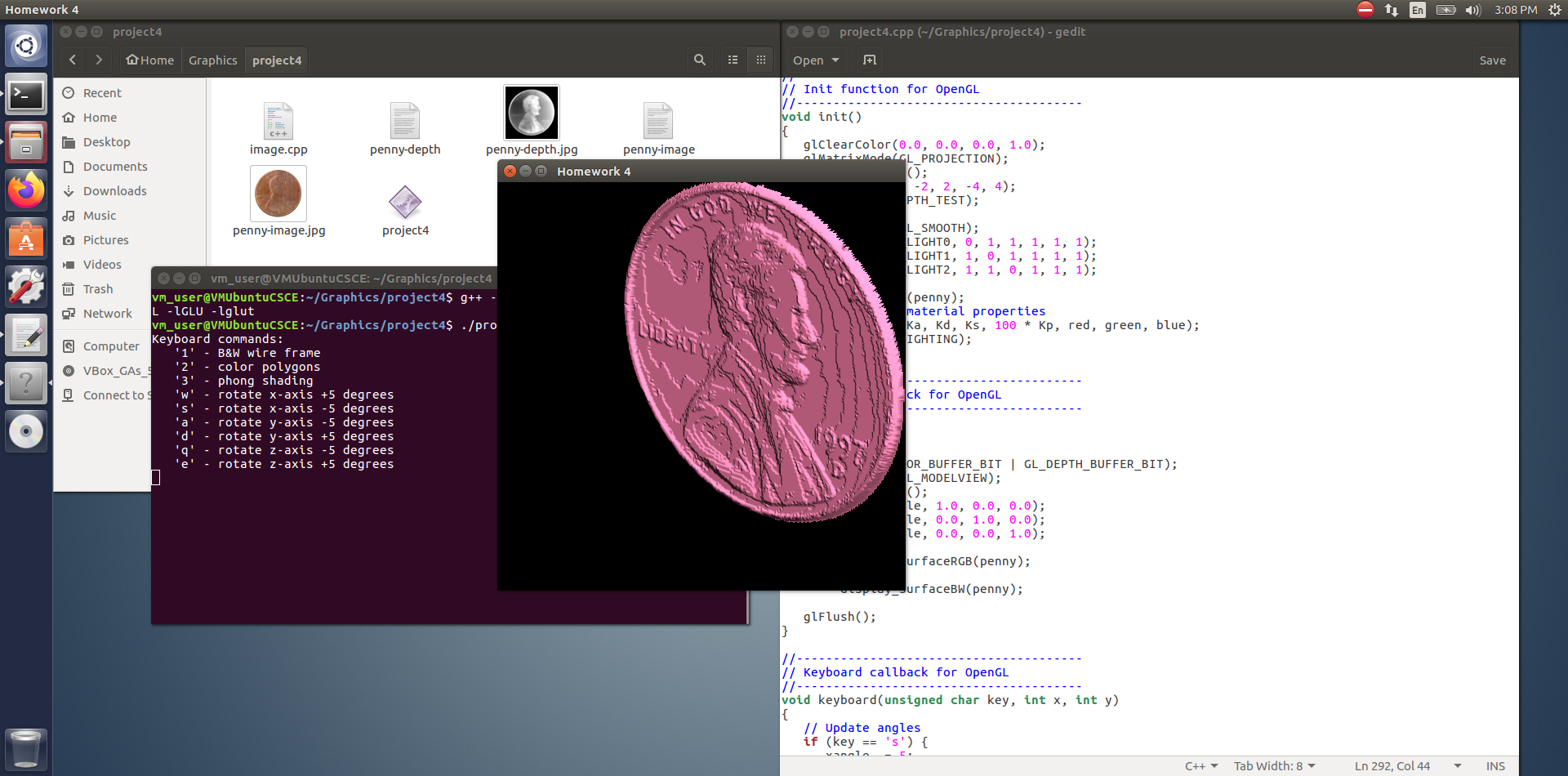
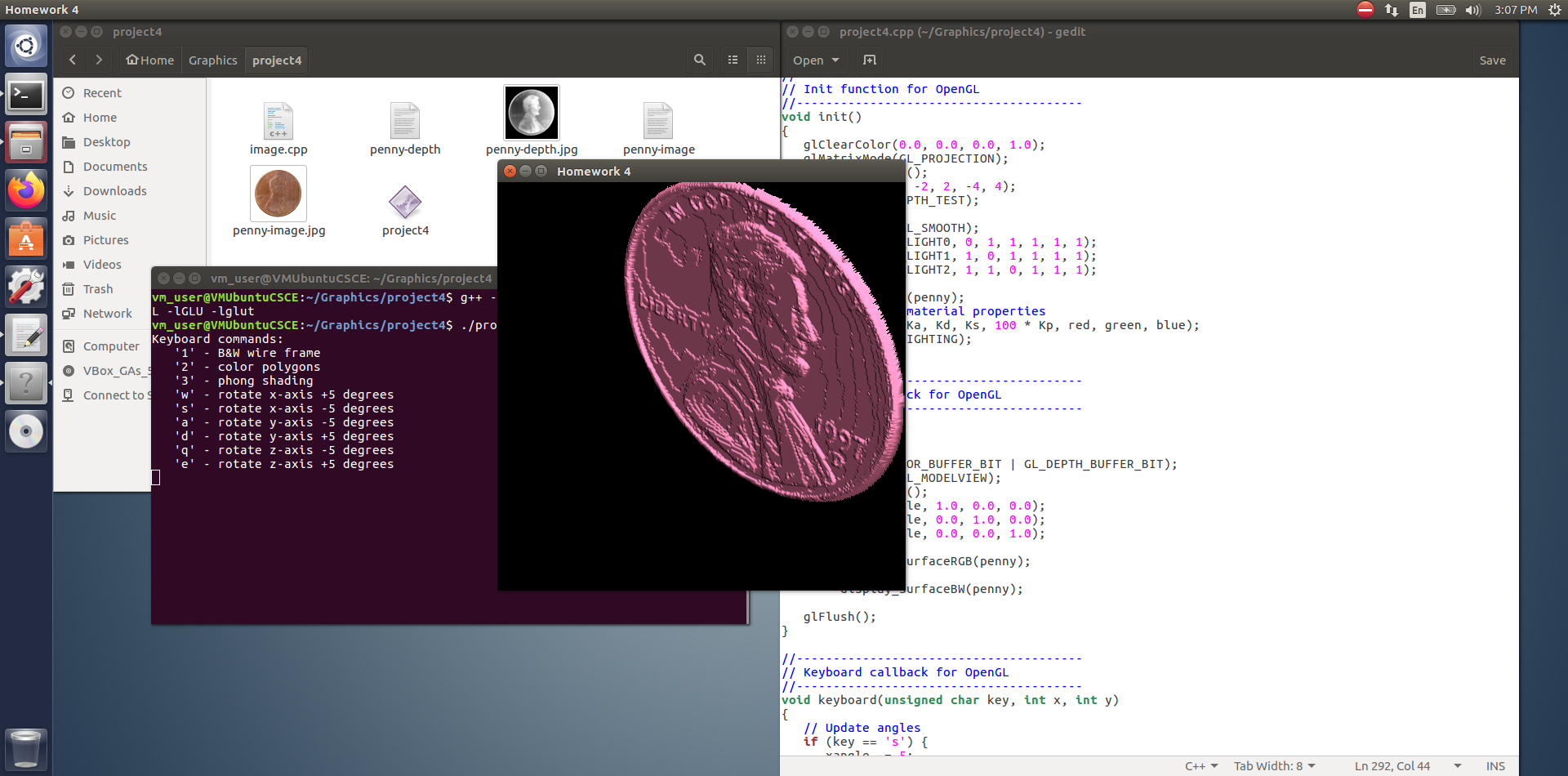
**Testing/Conclusion**

Since I implemented this program modularly, testing was done at each stage. I first verified that I could make a 2D image of the penny, then set the program to make it 3D, set up keyboard calls then Phong shading. I also tried different resolutions and material/lighting settings until I was happy with how everything looked. I then fixed up all the math, so the rotations worked properly and finally made the keyboard calls accurate to the instructions. At this point I’ve noticed that when running this program on my window OS, the model is cut in half with a funky triangle being formed between the cutting edge and the 0,0,0 origin but when ran on Linux everything looks correct. I have no idea why the program acts differently on different OSs.

I found this project very fun and engaging. I worked on it every day for about an hour everyday till it was done. I learned that setting up Phong shading is very much an art form and not a science. Values pertaining to the lighting and materials need to be adjusted a lot to look right. I also appreciated getting more practice with rotation and scaling and can see improvement on how I handle these problems. I think I spent about 6-7 hours on this project total.

**Pictures**





P.S. I just made it to where you must manually enter the names of the text files you want to use to use to create the model. On my final test I used penny-image.txt and penny-depth.txt.